

DESIGN, ANALYSIS AND EXPERIMENTAL VERIFICATION OF  
A SMART POWER SOCKET

by

Rohit Seshadri

A thesis submitted to the faculty of  
The University of North Carolina at Charlotte  
in partial fulfillment of the requirements  
for the degree of Master of Science in  
Electrical Engineering

Charlotte

2016

Approved by:

---

Dr. Madhav Manjrekar

---

Dr. Arindam Mukherjee

---

Dr. Valentina Cecchi



## ABSTRACT

ROHIT SESHADRI. Design, analysis and experimental verification of a smart power socket. (Under the direction of Dr. MADHAV MANJREKAR)

Demand for smart homes and smart devices has witnessed a consistent rise in the past decade. Home automation in residential buildings provides residents additional comfort of controlling their devices at the touch of a button. The concept of automated homes dates back to early 1930s but it has grown in consumer acceptance in this internet age courtesy of better sensors and seamless connectivity. Although smart devices and smart home technologies offer great convince to users, they tend to be expensive and are have to be first replaced with existing devices which requires high investments. There are alternatives to make existing appliances smart by adding adapters but these adapters too tend to be expensive.

Proposed smart socket ecosystem uses an existing home power distribution network for communications. This power socket uses power lines to transfer control commands and measurement data. The hardware consists of a low cost microcontroller to measure and control power characteristics of electric appliances. A Raspberry Pi like credit card sized computer acts as a server and serves as a communication hub for the entire socket ecosystem. This system utilizes PHP and Python scripts along with network socket communications to transfer commands and data from between server and power socket. Each Socket transfers power characteristics periodically to the server. Smart power socket system is capable of detecting an appliance whenever it is connected by utilizing K-Nearest Neighbor classification and regression algorithm. An Android application designed in

Android Studio acts as an interface to control and monitor the power socket. This application provides the user options to view power characteristics, schedule when to turn on connected appliance, learn and detect appliances connected to the socket.

## ACKNOWLEDGEMENTS

I would like to express gratitude to my family Seshadri Srinivasan, Sujatha Seshadri, Sujit Seshadri and Tulasi Burle for their continued motivation and belief in me.

I would like to thank my advisor, Dr. Madhav Manjrekar for giving me inspiration to work on this thesis. His guidance and encouragement helped me complete this thesis to the best of my ability.

It would be my duty to extend my gratitude to the committee members Dr. Valentina Cecchi and Dr. Arindam Mukherjee for taking time to be on my committee and for their valuable assistance during the course of my academics.

I thank EPIC (Energy Production and Infrastructure Center) for its generous support.

I would like to thank Rahul Agarwal, Monica BN, Anish Singam, Sandeep Reddy, Lokesh, Vijay, Rajashekar Reddy and my lab mates for motivating me during the course of this thesis.

## TABLE OF CONTENTS

CHAPTER 1: REVIEW	1
1. Introduction	1
1.1. Wired Technologies	2
1.1.1. X10 Home Automation	3
1.1.2. Universal Powerline Bus (UPB)	3
1.1.3. Home Plug Alliance	4
1.1.4. Home Plug Green PHY	4
1.2. Wireless Technologies	5
1.2.1. ZigBee Standard	6
1.2.2. Z-Wave	7
1.2.3. 6loWPAN	7
1.3. Commercial Review	7
1.3.1. The Energy Detective (TED)	8
1.3.2. Kill A Watt	9
1.3.3. Belkin WEMO	11
1.3.4. Samsung Smart Things	13
CHAPTER 2: SMART SOCKET DESCRIPTION	15
2. Introduction	15

2.1. Power Line Network Setup	17
2.2. Micro-controller Selection	17
2.2.1. Arduino	18
2.2.2. Intel Galileo	19
2.2.3. Raspberry Pi	20
CHAPTER 3: ARCHITECTURE AND COMPONENTS OF SMART SOCKET	24
3.1. Initial Setup and Proof of Concept	24
3.1.1. ADC Selection for Raspberry Pi	24
3.1.2. Voltage Sensing Circuit	27
3.1.3. Current Sensing Circuit	29
3.1.4. Webpage Design	33
3.1.5. Server and Database Installation	36
3.2. Arduino as Energy Meter:	38
3.2.1. Arduino Setup	38
3.2.2. Hardware Interface	41
3.2.3. Voltage Sensing Circuit	41
3.2.4. Current Sensing Circuit	42
3.2.5. EmonLib Calculations	46
3.2.6. Arduino Interface with Raspberry Pi:	47

CHAPTER 4: LOAD MONITORING AND RECOGNITION ALGORITHMS	51
4.1. Non-Intrusive Appliance Load Monitoring (NALM)	52
4.1.1. Steady State Signatures	52
4.1.2. Harmonic Frequency Signatures	53
4.1.3. Transient Signature	54
4.1.4. NALM Algorithm	54
4.2. Intrusive Appliance Load Monitoring	55
4.3. Load Recognition Algorithm	55
4.3.1. Bayes Filter Approach	56
4.3.2. k-NN Algorithm	57
CHAPTER 5: ANDROID APPLICATION DEVELOPMENT	66
5.1. Android Application Development	66
5.2. Smart Socket Android Application	67
5.2.1. Login Activity	68
5.2.2. Sockets Activity	70
5.2.3. Energy Monitor Activity	71
5.2.4. Schedule Activity	75
5.2.5. Graph Activity	80
5.2.6. Detect Activity	86



5.2.7. Learn Activity	89
CHAPTER 6: COMMUNICATION AND CONTROL DESIGN	92
6.1. Socket Communication	92
6.1.1. Smart Socket Server	93
6.1.2. Power Socket	96
6.2. Android Application Flow	99
CHAPTER 7: LABORATORY SETUP	102
7.1. Smart Socket System Design	102
7.2. Proof of Concept Setup	103
7.3. Alpha Prototype	104
7.4. Final Thesis Design	106
CHAPTER 8: CONCLUSIONS AND FUTURE SCOPE	109
REFERENCES	111

## LIST OF FIGURES

FIGURE: 1.1 A user interface image of the energy detective (TED) pro home	9
FIGURE 1.2: A Kill-A-Watt P4400 energy saver	11
FIGURE 1.3: User interface of Belkin wemo insight switch	13
FIGURE 1.4: Samsung smart things home monitoring kit	14
FIGURE 2.1: Top view of Arduino uno microcontroller	20
FIGURE 2.2: Top view of Intel galileo gen 2 development board	21
FIGURE 2.3: A picture of Raspberry pi 2 model 2	23
FIGURE 3.1: Simplified schematic of a voltage divider circuit	28
FIGURE 3.2: Modified voltage divider circuit	29
FIGURE 3.3: Schematic of a voltage sense circuit used for prototype design	30
FIGURE 3.4: Schematic of a current sense circuit used for prototype design	31
FIGURE 3.5: Schematic of sensing circuits along with raspberry pi	32
FIGURE 3.6: Default apache web server page	35
FIGURE 3.7: Smart socket prototype user interface	36
FIGURE 3.8: A screen shot of the userlist database used by android application	39
FIGURE 3.9: Structure of an Arduino sketch seen in the Arduino IDE	40
FIGURE 3.10: Schematic of voltage sensing circuit used for Arduino	44
FIGURE 3.11: Schematic of current sensing circuit for Arduino microcontroller	45
FIGURE 3.12: A screen capture of Arduino serial data monitoring screen	47
FIGURE 4.1: Representation of a Bayesian network	58
FIGURE 5.1: Screenshot of Smart power socket login screen	69

FIGURE 5.2: Screenshot of smart power socket, socket selection screen	71
FIGURE 5.3: Screenshot of smart power socket menu screen	72
FIGURE 5.4: Screenshot of smart power socket energy monitor screen	73
FIGURE 5.6: A picture of AC phase angle control module	74
FIGURE 5.5: Screenshot of smart power socket schedule screen	76
FIGURE 5.6: A screen capture of database appliance energy information	81
FIGURE 5.7: Screenshot of smart power socket graph screen	85
FIGURE 5.8: Screenshot of smart power socket device detect screen	88
FIGURE 5.9: Screenshot of knn.txt file from smart socket server with the appliance list	89
FIGURE 5.10: Screenshot of smart power socket learn screen	90
FIGURE 5.11: Modified knn.txt after addition of new appliance name	91
FIGURE 6.1: A flow chart showing smart power socket server process architecture	96
FIGURE 6.2: Power socket flowchart	97
FIGURE 6.3: Flowchart showing a sequence of functionalities in the smart socket android application	101
FIGURE 6.4: Smart socket application flow diagram	102
FIGURE 7.1: One line diagram of the entire smart socket system	103
FIGURE 7.2: Proof of concept setup with Raspberry Pi and ADS1115 ADC	105
FIGURE 7.3: Voltage and current readings from ADS1115	106
FIGURE 7.4: Alpha prototype setup using Arduino and Raspberry Pi	107
FIGURE 7.5: Top view of ACS712 current sensor designed by sunrom technologies	108
FIGURE 7.6: A picture of the electric project box used in the final design	109

FIGURE 7.8: Top view of smart power socket 110

FIGURE 7.9: Smart power socket components inside an electrical box 110

## CHAPTER 1: REVIEW

### 1. Introduction

Technology has advanced rapidly in the past few decades. Ten years ago, cellphones were used just to make calls. The rate at which technology is growing is completely dependent on consumer's demand for more features, faster speed and better usability. Energy consumption has also increased tremendously in this period. Between years 1973 and 2013, there was a 66% increase in the total energy usage in the world [1]. Energy consumption is projected to continue to increase drastically in the future. This significant year-to-year rise in energy consumption has increased the need for Energy conservation and efficient energy management solutions. Rise in harmful gases in the atmosphere especially CO<sub>2</sub> can be directly linked to the gases emitted by the power plants. In developed countries, smart meters are being implemented at household levels to help utilities and consumers to manage their energy consumption in a more efficient manner. Smart meters can provide more accurate readings than the regular analog meters and also provide better reliability, faster power restoration, convenience and better control [2]. Smart meters provide facility for variable pricing which can encourage consumers to use appliances that require high power at a time when the cost is less. This gives the opportunity to the utility to divert power to high demand areas during peak hours thereby reducing the peak demand on the generators and increasing their reliability. Although smart meters

provide more details than a regular meter, the information collected is generally sent directly to the utility and benefit for the average consumer is limited.

Home automation provides users with a higher level of control on household appliances. Users can control their appliances remotely or by a means of an automated process. Communication plays an import role in home automation. Different means of communication are used my various manufacturers to achieve a seamless connectivity between the devices. Technologies such as Power Line Communication (PLC), Radio Frequency (RF), Wi-Fi, ZigBee, Bluetooth, etc. have been used in home automation. These technologies can be divided into two groups based on them being a wired or wireless solution. A literature review of some of the most common protocols is given below.

### 1.1. Wired Technologies

Powerline communication technology is widely used wired solution for home automation. In 1838 the first remote electricity supply metering and in 1897 [48] the first patent on power line signaling were proposed in the United Kingdom [49]. The advantage with PLC is that there is no need for additional wiring or investment in new communication equipment. The existing electrical wiring of the homes can be used to achieve communication. PLC modems are used to make communication in power supply networks. Data signal from conventional communication devices, (computer, telephone) is converted by PLC modem in a form that is suitable for transmission over power lines [50]. Some home automation protocols based on PLC that have been introduced to the market in the past few years are explained below.

### 1.1.1. X10 Home Automation

X10 home automation protocol was initially launched in the year 1975 and since then, it has seen a wide range of acceptance by users. X10 utilized power lines as a means of communication between the user and the devices. Later models had an additional feature of using RF protocol in addition to power line communication. X10 protocol functioned by sending short messages to the receiver from a controller pad. Transmissions are synchronized to the zero crossing point of the AC power line. When the AC wave approaches zero crossing, the signal is transmitted [3]. The message contained the device id followed by the control bit (on/off). One of the major drawback of this technology at that time was the slow response time of the signals, frequent loss of signals when more than one control signal was sent at once and the limited functionality that it offered. Another limitation was that only 256 devices can be addressed in a network [4].

### 1.1.2. Universal Powerline Bus (UPB)

This protocol is also based on PLC. Compared to X10, it is about 100~1000 times more reliable, 20 to 40 times faster [5] and can be used in presence of other PLC protocols such as X10, CEbus, etc. as it has better noise isolation. UPB also supports two-way communication and can incorporate up to 64,000 devices compared to only 256 in X10 [5] [51]. UPB uses pulse modulation based methodology to transmit its messages. These messages are transmitted at a higher voltage level than X10 and are also transmitted at a low frequency level (4 to 40Hz). As a result, it can carry more power than high frequency technology like X10 [5].

### 1.1.3. Home Plug Alliance

The Home Plug Alliance was formed by industry leaders Motorola, Intel, LG, Texas Instruments, etc. The standard was developed by three groups addressing key issues such as: in-home connectivity for consumer appliances, broadband over power lines and home automation [20]. IEEE 1901 standard was approved in 2010 and home plug's OFDM modulation technique was included in the standard. The first version of HomePlug known as Home Plug 1.0 was capable of data transfer speeds of 14Mbits/s. The latest versions of home plug devices Home Plug AV2, are capable of reaching data speeds of 1000MB/s. The HomePlug V2 devices can be used for digital home theater entertainment systems, broadband access, smart energy monitoring, mobile data, etc. [21]. Chipset architecture is designed by Qualcomm, Broadcom, etc. Home Plug network adapters are sold by various manufacturers such as Lea Networks, Cisco, TP-Link, Netgear, etc. These adapters are often bought in pairs where one of them is placed near the home router where the ISP connection terminates and the other adapter can be placed on any power outlet in the home to receive broadband connection at the other end. This technology provides a simple solution to dead zone issues of wireless communication as all the rooms of a house are designed to have power outlets and unlike Ethernet, no additional wiring is required and purchase of expensive routers can be avoided.

### 1.1.4. Home Plug Green PHY

Home Plug alliance in 2010 released a new subset of HomePlug AV known as HomePlug Green PHY [22] for integration with smart grid. The new chipsets are small enough to be designed for embedded system applications. The Green PHY specification is designed to share the data over the home network and also to utilities. They consume 75%



lesser energy than the HomePlug AV devices and have a peak data transfer rate of 10Mbits/s. The low transfer rate is suitable for home automation and monitoring applications as they transmit limited quantities of data and do not require higher bandwidth [22]. HomePlug Green PHY technology has obvious advantages versus older, low-speed PLC technologies. Because this technology operates at much higher signal frequencies, it is much more impervious to noise on the powerline, and has no difficulties with phase coupling. Also, as mentioned above, since HomePlug Green PHY is interoperable with HomePlug AV and AV2 and supports both IPv4 and IPv6, it enables a true merging of home automation and home control with high speed digital entertainment home networks [23].

## 1.2. Wireless Technologies

Descriptive of a network or terminal that uses electromagnetic waves (including RF, infrared, laser, visible light—and acoustic energy) rather than wire conductors for telecommunications [52]. Systems which implement wireless communication protocols, generally have sensors, switches, etc. that can receive and transmit information wirelessly. Network systems that are implemented in homes is referred to as Home Area Networks (HANs). HAN technology enables one to remotely connect to and control many automated digital devices throughout the house [6]. A home automation system based on wireless technology requires low energy consumption from the sensors. The network range should encompass the entire house. In a typical smart home integration, most of the solutions offered are based on the IEEE 802.15.4 standard which was created specifically for low rate Wireless Personal Area Networks (WPANs). Although the WPANs range is limited to 10 meters, modifications have been made to suite the requirements of home automation by

modifying certain layers of the standard. Technologies such as ZigBee, 6LoWPAN, Z-Wave, etc. make use of this standard.

### 1.2.1. ZigBee Standard

ZigBee is the language for a wide variety of smart home devices so companies can deliver an integrated ecosystem of home monitoring, energy management, heating and cooling, security and convenience devices [53]. ZigBee is a low cost and low power standard that is maintained and published by the ZigBee alliance. The low cost of this technology has allowed a wide deployment of sensors and control devices and due to the low power requirements, it can be powered by batteries. ZigBee systems utilize a mesh network architecture. This type of architecture has higher flexibility in terms of communication range. Each system consists of a device known as the ZigBee Coordinator (ZC) [54]. This device is the root of the network and stores entire information of all the other devices that are in the network. ZigBee Routers (ZR) act as intermediate routers and help to transmit the message further than the range of the ZC. These routers can also run as applications performing sensing and monitoring operations. The third type of devices that is described in the standard are known as ZigBee End Devices (ZED). The ZED have only limited functionality by which they can communicate with either the ZC or the ZR. They cannot perform additional routing functions like the ZR [4]. The average distance between two devices can be as large as 30 meters. If the ZC has to send a message to a ZED that is not in its range, the ZC can send the message to its nearest ZR which then passes the message to the next ZR until the ZR that is in range of the ZED is reached. The number of hops is limited to 30 when the network follows mesh routing, 10 when it follows tree routing and 5 when it uses source routing [7].

### 1.2.2. Z-Wave

Z-wave is a proprietary wireless communication standard that is used to control and monitor applications by using RF based technology [8]. Like the ZigBee standard, Z-Wave supports mesh networks but it achieves this without the need of a coordinator. The Z-Wave appliances operate in the sub 1Ghz frequency band. The data transfer speeds in Z-Wave is in the range of 100kbits/s [8]. It utilizes frequency-shift keying radio and the throughput is 40Kb/sec [9]. Z-Wave devices operate in the ISM bands and as a result, these devices are not subjected to the interference by the 802.11 and WPAN devices. The various devices that are available under this standard are classified into two kinds; controllers and slaves. The controllers are similar to the ZC. They hold all the messages and routing information. The slaves act as the ZED and execute the commands it receives from the controllers. Z-Wave follows source routing and allows a maximum of 4 hops for data transmission between nodes.

### 1.2.3. 6LoWPAN

This is a low power wireless communication protocol that was designed to be applied to small devices that can be connected to the internet. The standard is based on IEEE 802.15.4 protocol [55]. Changes were made to the methods in which the IPv4 and IPv6 packets were delivered. IPv6 is also used by some of the smart meters as a result, a better integration could be possible between the home automation products and the smart meters.

## 1.3. Commercial Review

The technologies and protocols mentioned in the previous sections have found wide range of acceptance and many adopters. Some of the popular adopters of these protocols are explained in the following section. A brief review of some of the most popular energy

monitoring and smart home appliances are covered. There are different kinds of solutions that are available in the market. Some of them require additional sensors and equipment to be mounted at the main breaker panel of the house and some of them require the purchase of adapters in order to expand the range of functionality of existing power outlets.

### 1.3.1. The Energy Detective (TED)

TED is an energy management product that helps in measuring the energy consumed by individual residences in real time [<http://www.theenergydetective.com/>]. TED uses energy monitoring sensors that are mounted inside the main breaker cabinet. The sensors are mainly clamp on current sensors that help in measuring the power consumed on each circuit of the home. These sensors relay the data to a Measuring Terminal Unit (MTU). The sensors use PLC to transmit the data to the MTU. Data that is accumulated on the MTU can be viewed by means of an interactive webpage or an interactive device that can be purchased from TED [10]. The software that runs on the MTU that shows HMI to the users is known as footprints. This software is able to show real time power consumption as well as historical data of up to 10 years. MTU data can be accessed from anywhere at home by connecting the MTU to a router and visiting the network address of the router on a web browser. The cost to purchase the basic model (TED 5000-G) is \$199.95 [11]. The package includes a set of Current Transformers (CT) and a MTU. Gateway is enabled with the Footprints software.



FIGURE: 1.1 A user interface image of the energy detective (TED) pro home [11]

TED provides energy monitoring capabilities that were earlier unavailable to consumers. Although consumers can take advantage of these features and save money by conserving energy, TED does not provide any control options to users and because monitoring takes place at the main circuit board level, individual appliance loads cannot be monitored and controlled.

There are other commercially available products that provide individual appliance level monitoring and control. Some of the popular manufacturers are Samsung (Smart Things), Belkin (WEMO), P3 International (Kill A Watt), etc. [12-14].

### 1.3.2. Kill A Watt

This is a power meter that can be used by consumers to measure the power consumed by individual home appliances manufactured by P3 International

[<http://www.p3international.com/>]. This device is an adapter between the appliance and power socket. Adapter consists of an LCD display that is used to show energy consumed by connected device. Kill A Watt device can measure supply voltage, current consumed by connected device, energy consumed (kWh), power factor of the connected appliance and apparent power. Kill A Watt meter can measure power readings as low as 0.1Watt. The most important use of this device is to measure standby power consumption (Vampire power) of electronic devices such as Televisions (TV), Home Theater systems, power electronic devices, etc. A basic version of Kill A Watt (P4400) has 5 analog buttons on the front which change the type of information that is displayed on the LCD [12]. P4400 is powered directly from the mains supply and total energy consumed is shown until this adapter is connected to the mains. Once adapter is disconnected, it resets the measurements. This is one of the major drawbacks of this device as there is no history of energy consumption data that is available to the users. Future versions have the capability to be powered by a battery and has the ability to store the measured information [56]. Like TED, the Kill A Watt is a measuring device but it can be used to monitor individual load energy consumption. Certain versions are available that can prevent standby energy consumption of TV's by completely switching it off.

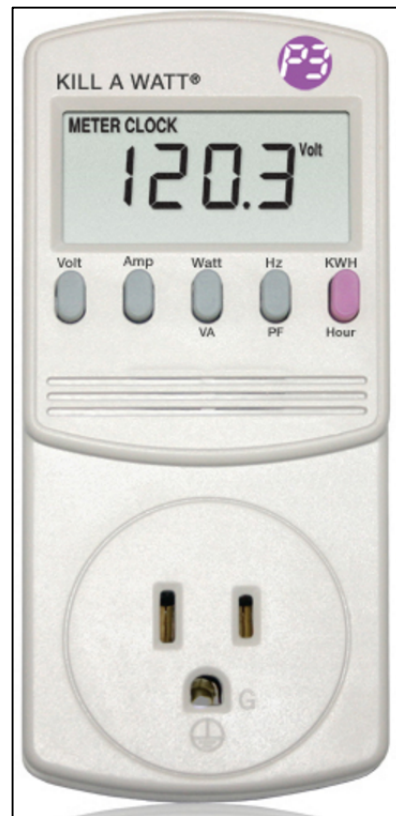


FIGURE 1.2: A Kill-A-Watt P4400 energy saver [12]

This is done by placing an IR sensor at the adapter which switches off when the TV is turned off by the remote. P3 International also has an adapter that turns on the device connected to it when it detects motion in the room. The cost of the Kill A Watt P4400 is \$22.99 and can go up to \$60 for some models [12] [56].

### 1.3.3. Belkin WEMO

WEMO is a family of smart devices manufactured by Belkin that are simple and customizable products that allow you to control home electronics from anywhere [57]. These devices offer the user more control over their appliances. WEMO has a range of products that show the energy consumption of devices (WEMO insight switch), on/off control, dimmable LED's (WEMO light switch), cookers, etc. These devices are controlled

by an application on the user's cellphone. The devices are connected to the application by Wi-Fi or 3/4G connectivity [13]. The insight switch is an adapter module which is similar in functionality to the Kill A Watt meter but much smaller in size. The application on the smartphone is the HMI for the switch and can be used to check the energy consumed by the devices that are connected to it. The switch can also be turned on/off from the application.



FIGURE 1.3: User interface of Belkin wemo insight switch [13]

Belkin also provides a range of popular household appliances such as cookers, coffee makers, heaters, air purifiers, etc. which can be controlled from anywhere.



### 1.3.4. Samsung Smart Things

SmartThings Inc. is a smart home products designer that was bought by Samsung in 2014. Samsung has built an ecosystem of smart home devices under the Smart Things range of products. These products include motion sensors, power outlet, water leak sensors, temperature sensors, security cameras, etc. [14]. This system consists of a main control device known as Hub. All communication between user and the connected devices occurs through this hub. All above mentioned products connect to the hub at user's home and this allows the user to control the devices from anywhere in the world.

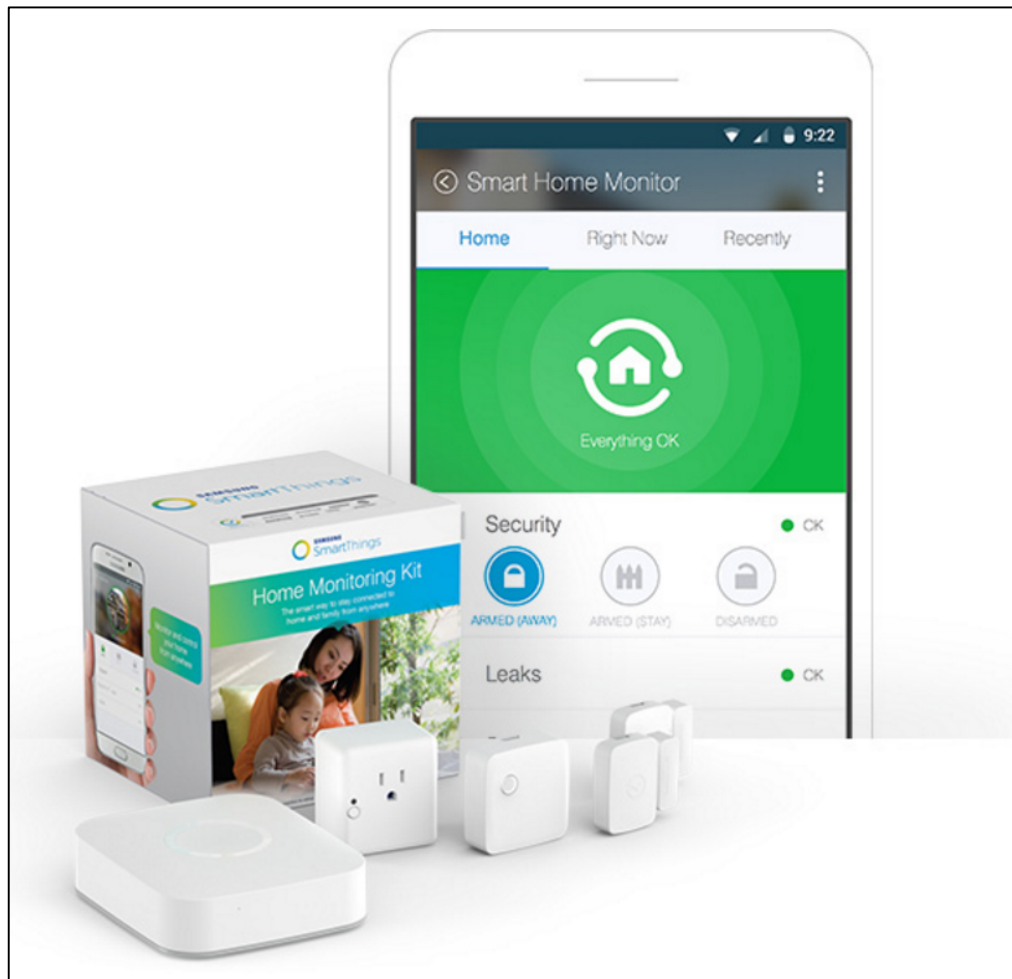


FIGURE 1.4: Samsung smart things home monitoring kit [14]

The hub requires an Ethernet connection and supports protocol such as ZigBee, Z-Wave and IP. The support for multiple protocols gives the user the option to purchase products from vendors other than Samsung and allows integration with the smart things line of products. The power outlet is similar to the WEMO insight switch. It can be used to control lights, small appliances and electronics from anywhere in the world.

Connected appliances can also be scheduled when to turn On/Off. Initial cost of setup depends on number of smart sensors purchased by the user, but the hub which is necessary costs \$99 and the other smart things range from \$30 to \$189. Samsung also offers a kit which consists of the hub and 4 sensors for \$250 [14].

Products mentioned above provide consumers different smart home capabilities. They vary in features, communication protocols and cost. New innovations and breakthrough technology advancements in combination with huge market demand for more intelligent devices has helped in increasing competition for smart home devices and internet connected appliances.

## CHAPTER 2: SMART SOCKET DESCRIPTION

### 2. Introduction

The idea of a smart home is to give users high level of control at relative ease and convenience. The integration of smart home devices has to be seamless in order to achieve wide acceptance from a large consumer base. Technologies such as NALM have been around from the early 1980s but it has not been widely implemented. Other new appliances such as TED, Kill A Watt, WEMO, etc. have found limited acceptance due to the cost of installation, limited functionality offered by some and the method of implementation. TED requires installation at the main circuit board and does not provide individual appliance monitoring. Kill A Watt, Belkin WEMO, Samsung SmartThings, etc. [10-14] require users to place an adapter in front of the power outlet. Research has been conducted in appliance load monitoring and detection by using the data accumulated from the sensors placed near the meters [18] [58]. Research in intrusive load monitoring by placing sensors for each individual appliance as adapters have also been conducted [59] [60]. All above described products and ongoing research activities either place additional adapters for appliance metering or require partial additions or changes to the home circuitry [15] [10]. Proposed method in this thesis is a product that can perform energy monitoring, provide on/off control to the user, an Android mobile application as the HMI and a background appliance recognition algorithm that can detect which appliance is connected based on the appliance name given by the user. PLC was chosen for the mode of communication to ensure that

future versions of designed socket are able to sit flush with the wall just like any regular wall outlet. Devices that utilize communication methods based on wireless communication protocols (ZigBee, Wi-Fi, Bluetooth, etc.), operate in the ISM band and are susceptible to co-channel interference from other devices operating in the same band. Although recent standards in Wi-Fi (802.11ac) have addressed the issue of overcrowded ISM band by using the 5GHz bandwidth, routers that support this protocol are expensive and many of the legacy devices do not have the capability to receive the signals as they have the 2.4 GHz receiver. Also, if the number of wirelessly connected devices to a particular network increase, it might cause latency issues and in some extreme cases might result in loss of packets or connectivity. Dead zones are another issue that has plagued wireless connectivity in many large households. Lack of transmission signals can render smart home devices that rely on wireless connectivity useless. To overcome these issues, many manufacturers are looking at technologies such as PLC and Broadband over Power Lines (BPL). PLC technology has been around for more than 50 years. There has been significant research that has been undertaken in on this technology and recent advancements in PLC technology by use of different modulation technologies is able to achieve better performance than some Wi-Fi connections.

Latest range of products from HomePlug group are capable of transmitting data securely over powerlines. Adapters can be used directly out of the box and paired in an unsecured network mode. To enable secured network mode, LEA provides few simple steps in the instruction manual which can be implemented easily. The security adds a 128bit Advanced Encryption Standard (AES) between the two NetPlugs. This means that although the NetPlugs do not encrypt the data that is being sent, the transmission layer in which the

data passes is encrypted. This makes it very difficult for attackers to hack into the transmitted signal as long as the device that sends the signal is protected. New NetPlug devices can be added to the secured network by repeating the setup process. The encryption prevents leakage of signals between two different networks or to other power line devices that do not form a part of the network. Multiple power line networks can be created within the same house as a result of secured connections.

### 2.1. Power Line Network Setup

A power line network was setup in the Renewable Energy Laboratory (REL) in the Energy Production and Infrastructure (EPIC) facility at UNC Charlotte. The power line adapters were donated by LEA networks for research on this project. The adapters that are used are from LEA networks NetPlug AV200 PLC kit. One of the adapter has to be plugged in to the router that is connected to the ISP and the other adapter can be placed at any power socket in the home to receive broadband over power lines.

### 2.2. Micro-controller Selection

With the network setup, a market survey on the available microcontrollers for rapid prototyping resulted in a list of controllers that could be used to develop the metering hardware. The minimum requirements for the controller were:

2. Low power consumption
3. High Speed operations
4. Capability to be controlled remotely via another computer or Android Application
5. Built-in or add on ADC modules
6. General Purpose Input Output (GPIO) pins for easy testing
7. Ethernet connectivity (RJ45 jack) to interface with NetPlug adapter

Initial shortlist included a range of microcontrollers based on previous experience and quick availability. Some of the controllers that were researched were Arduino Uno, Raspberry Pi, Intel Galileo and Edison. Cost of the controllers played a vital role in decision making as well. Intel Edison board lacks inbuilt Ethernet support and the cost of the Edison board is higher than all the other boards which is why it was not chosen for this project. Intel Galileo, Arduino Uno and Raspberry Pi were all developed to be rapid prototyping devices.

### 2.2.1. Arduino

The Arduino is a programmable controller that is based on the Atmel328 controller. The board has 6 analog inputs and 14 digital pins that can be used as both inputs and outputs. The Arduino can be powered by either a DC power source or through the single USB input. The Arduino Uno board can be used for metering and control projects. The board can be programmed by using the Arduino IDE that can be downloaded from the Arduino website. The Uno is coded using C language. Arduino Uno is open source and has a huge community of developers that support the project and develop libraries that enhance the productivity of the board.

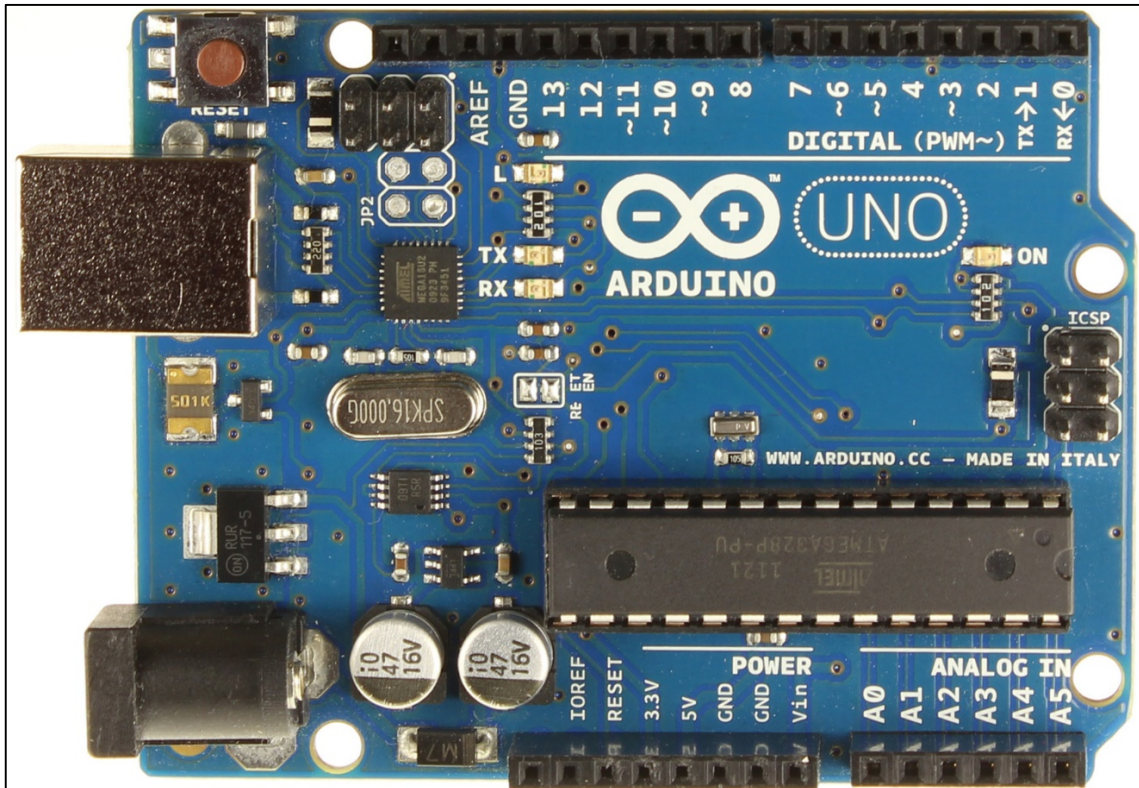


FIGURE 2.1: Top view of Arduino uno microcontroller [24]

The Arduino board lacks any audio/video interfaces. The processor is clocked at 16MHz and has limited storage capacity [24]. Initially this was the reason the Arduino Uno was rejected for this project but after later research, this board was also used. It will be explained in a later section. Although the Uno is a simple microcontroller prototyping device, there are many accessories known as Shields that can be added to the board to enhance its connectivity. Some of the shields add Ethernet connectivity, Wi-Fi, etc.

### 2.2.2. Intel Galileo

The Intel Galileo is a development board based on the Intel Quark application processor [40]. It is a System On a chip (SOC) that is of a 32-bit Intel Pentium brand. The board was designed with Arduino and hence all the input and output pins are compatible with the shields of the Arduino Uno. Internally the Galileo is powered by a single core 32-

bit processor clocked at 400MHz. This board is very powerful compared to the Uno and also has a maximum memory of 2GB. The controller board also has integrated LAN ports. It has 16 GPIO pins and 3 USB 2.0 ports.

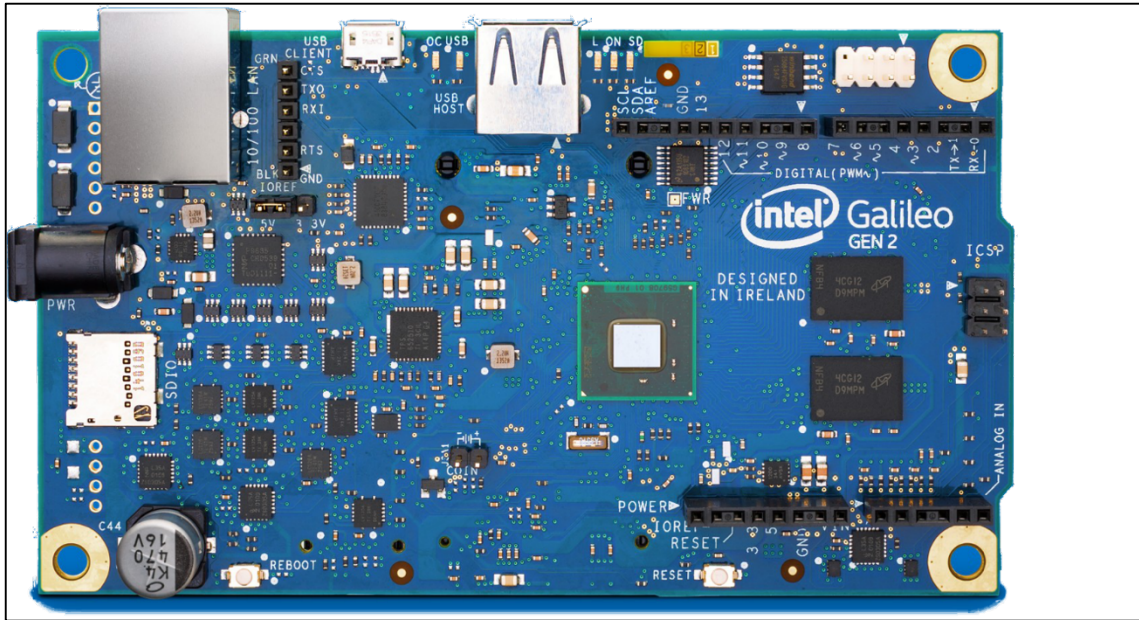


FIGURE 2.2: Top view of Intel galileo gen 2 development board [40]

The board comes with a pre-installed Operating System (OS) and can be easily switched to a different Linux based OS known as Yocto 1.4 Poky Linux release. Intel Galileo also supports the Windows 10 IOT OS. The dimensions of the Galileo board is larger than the other boards at 123.8 mm (L)  $\times$  72.0 mm (W). The device also has limited Random Access Memory (RAM) at 256MB [40]. Raspberry pi is both faster and smaller than Intel Galileo.

### 2.2.3. Raspberry Pi

The Raspberry Pi is a credit card sized computer that features a complete operating system, has audio and video peripherals, Ethernet port, USB ports a fast processor and a large number of add on devices. The Raspberry Pi 2 model B was chosen as the controller



for this project. The Pi is powered by a 900MHz, quad core ARM Cortex-A7 CPU along with 1GB RAM. It also has 4 USB ports, One Ethernet (RJ45) port, 40 GPIO pins, HDMI support and Micro SD card storage expansion. The official OS of the Raspberry Pi is a flavor of the full Ubuntu OS known as Raspbian. The Raspberry Pi also supports other OS such as Windows 10 IOT core, UBUNTU Mate, OSMC, etc. [26]. The Raspberry Pi supports multiple programming languages such as C, C++, Java, Python, etc. The Raspberry Pi can perform functions that any similarly powered computer and does this at a very small footprint. The dimensions of the Raspberry Pi are 85.60mm x 56mm x 21mm and this makes it smaller than the Galileo and suitable for the project. \

The Raspberry Pi does not have an inbuilt Analog to Digital Converter (ADC) but external ADC's that are compatible with the Raspberry Pi are widely available in the market. Some of the most popular sources for Raspberry Pi accessories are Adafruit and Sparkfun.

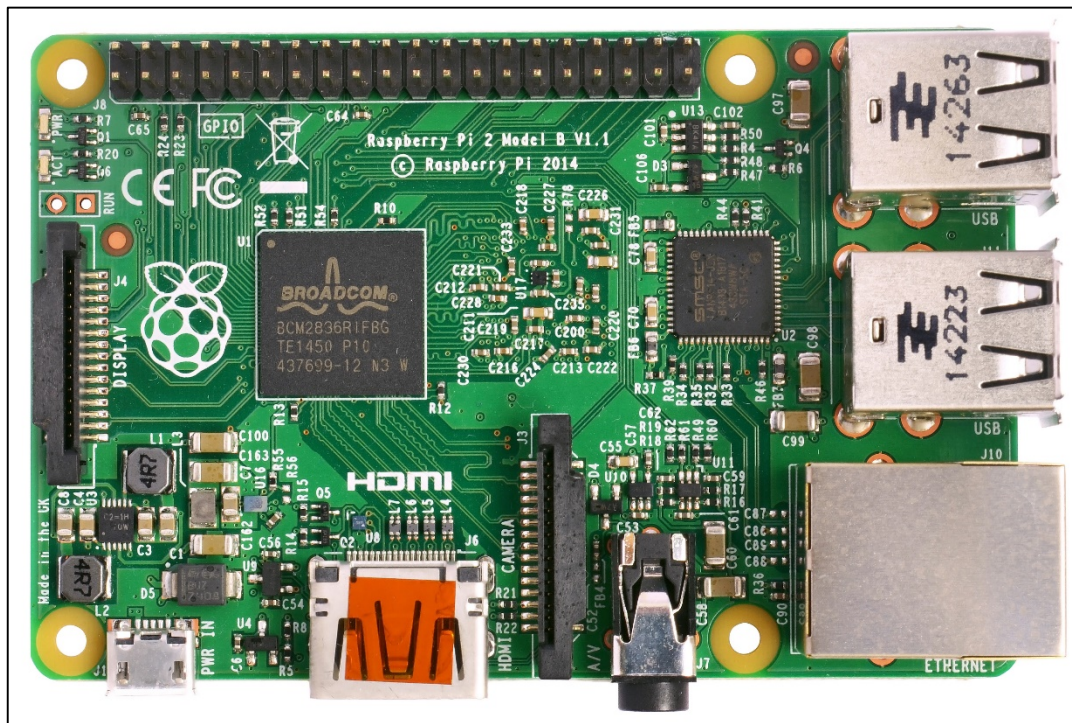


FIGURE 2.3: A picture of Raspberry pi 2 model 2 [27]

The Raspberry Pi has multiple data interfaces by which external hardware can be connected to it. It supports Serial Peripheral Interface (SPI), Two wire interface I<sup>2</sup>C, Serial Data UART, Digital Serial Interface (DSI), Camera Interface, etc.

The initial setup for raspberry pi is given below [27].

1. First, the image of the OS is downloaded from Raspberry Pi website and with help of Win32 disk imager software, it is written onto the micro SD card that will also act as the storage for the data that will be saved on raspberry Pi.
2. Once the SD card is inserted into raspberry pi, peripherals such as monitor, keyboard and mouse can be connected to the Pi and powered on.
3. The OS will automatically install and configure in a few minutes.

Once Raspberry Pi was setup in REL, a simple program to enable GPIO pins was run. This program involved turning on a LED and was taken from Raspberry Pi documentation section on the website. Raspberry Pi also has the ability to allow users to login and use it remotely via Source Shell (SSH). This was setup on the Pi so that a dedicated monitor and peripherals were not required. Simple terminal commands were used to enable SSH and a Virtual Network Computing (VNC) server was installed using the following steps:

1. `sudo apt-get install tightvncserver`
2. `tightvncserver`

After this step, Pi asks for a password which will be used when trying to login remotely. Once the installation is completed, every time a user tries to login via n SSH session using the Raspberry Pi's IP address, the following command has to be entered to start the VNC server

3. `vncserver :1 -geometry 1920x1080 -depth 24`

A VNC viewer can be downloaded on the user's computer to provide the Graphical User Interface (GUI) of the Pi [29].

## CHAPTER 3: ARCHITECTURE AND COMPONENTS OF SMART SOCKET

### 3.1. Initial Setup and Proof of Concept

A PLC network and two Raspberry Pi's were setup in REL. PLC connectivity was verified by establishing a remote desktop connection to the Pi using the power line network. The connection was as good as a wired connection with no noticeable transmission lag. To establish proof of concept, the task was to make an energy meter with a Raspberry Pi and transmit data over PLC network and this data should be accessible to any device that is connected to the network. Following steps were undertaken to achieve this:

1. Choose ADC compatible with Raspberry Pi
2. Build Voltage sensing circuit
3. Build Current sensing circuit
4. Design a webpage that shows the measured voltage, current and power consumed
5. Install a server on the Raspberry Pi and create a Database for users.

#### 3.1.1. ADC Selection for Raspberry Pi

Raspberry Pi lacks an inbuilt ADC and as a result, external ADC modules were purchased. There are many ADC's available that are compatible with Raspberry Pi. A choice had to be made between ADS1115, MCP3008 and ADS1015. All three ADC's are easy to interface with the Raspberry Pi and can be programmed in Python using libraries provided by Adafruit. The MCP3008 is an ADC manufactured by Microchip. It is a 10-bit, 8 channel analog to digital converter that can interact with the raspberry pi via SPI

connectivity. ADC is capable of taking 8 single ended inputs or 4 pseudo-differential inputs. ADC can be powered by a supply voltage of 2.7V-5.5V. Maximum sampling rate for the ADC is 200ksps when supply voltage ( $V_{dd}$ ) is 5.5V and drops to 75ksps when  $V_{dd}$  is 2.7 volts. Although the ADC is capable of achieving high sample frequencies, Raspberry pi is unable to receive data at such a high rate through the SPI interface. Also, this ADC was not chosen since it has a resolution of 10-bit and the other ADC's have a higher resolution.

ADS1015 is a 4-channel, 12bit ADC with a Programmable Gain Amplifier (PGA), manufactured by Texas Instruments. ADC connects to the Raspberry Pi via an I<sup>2</sup>C connection and provides 3300 samples/second. It can be configured to take 4 single ended inputs or 2 differential inputs. The ADC can be powered by either the 3.3V or the 5V output pin of the Raspberry Pi [30]. ADS1115 is similar to ADS1015 in terms of connectivity but it has a higher resolution of 16-bits. This means that the ADC has  $2^{16}$ (65536) values that it can detect in a given range. ADS1115 is capable of measuring 860 samples per second and the input can range between  $\pm 4.096$  volts [31]. Although the ADS1115 has a lower sampling rate than the ADS1015, it was preferred due to the high resolution.

ADS1115 was connected to the Raspberry Pi via I<sup>2</sup>C and the libraries provided by Adafruit was used to measure the voltage across an LED. Single ended voltage across LED was measured using the Python code below.

```
#!/usr/bin/python
import RPi.GPIO as GPIO
```

```
import time

import signal, sys

import datetime

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

led = 4

import time, signal, sys

from Adafruit_ADS1x15 import ADS1x15

def signal_handler(signal, frame):

    print 'You pressed Ctrl+C!'

    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

#print 'Press Ctrl+C to exit'

#ADS1015 = 0x00 # 12-bit ADC

ADS1115 = 0x01      # 16-bit ADC

# Select the gain

gain = 4096 # +/- 4.096V

# Select the sample rate

sps = 860 # 860 samples per second

# Initialise the ADC using the default mode (use default I2C address)

# Set this to ADS1015 or ADS1115 depending on the ADC you are using!
```

```

adc = ADS1x15(ic=ADS1115)

while 1:

Read channel 1 in single-ended mode using the settings above

    volts = adc.readADCSingleEnded(1, gain, sps)

    print "%.2f" % (volts)

```

### 3.1.2. Voltage Sensing Circuit

Input voltage for ADC cannot exceed +4.096 volts. Mains voltage which is 120Volts has to be stepped down to the range that is acceptable by the ADC. To do this, we can either use a transformer or a voltage divider circuit. A voltage divider circuit was built to step down the voltage from 120 to  $\approx 4$  Volts peak-peak. Circuit diagram for the connection was borrowed from a Texas Instruments energy meter documentation and changes were made to the circuit. The resistor values were calculated using Voltage Divider equation. A simple voltage divider circuit looks as shown below.

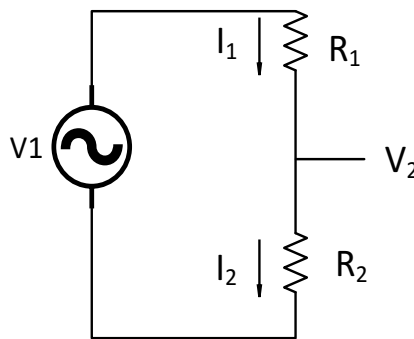


FIGURE 3.1: Simplified schematic of a voltage divider circuit

Voltage across resistor R2 and total resistance of the circuit is calculated using the equations below:

$$V_2 = I_2 * R_2 \quad (1)$$

$$R = R_1 + R_2 \quad (2)$$

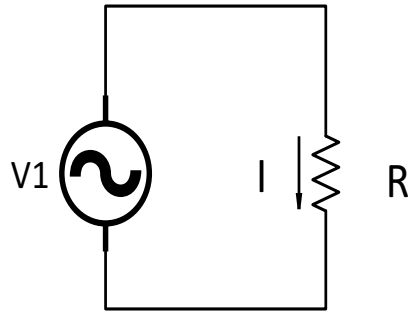


FIGURE 3.2: Modified voltage divider circuit

Using Ohm's law, we can calculate the total current that flows through the circuit

$$I = \frac{V_{in}}{R_1 + R_2} \quad (3)$$

The magnitude of the voltage across R2 can be achieved by equating I to I2 since it is the same current. The resulting equation is found to be

$$Power = K_p \frac{\sum_{n=0}^{samples} V(n)I(n)}{samples} \quad (4)$$

Input voltage can be reduced to any value based on resistor values chosen to build the circuit. Value of R1 is divided to both the line and neutral legs to maintain a balanced circuit and prevent damage to the ADC in case of any neutral spikes. Since ADC is unable to take negative voltage, output of this circuit was sent through a bridge rectifier and then supplied to the ADC. The positive terminal was connected to the pin A0 of the ADC and negative was grounded and connected to ADC pin A1. A differential connection was used to get better readings. The complete voltage sensing circuit diagram can be seen below.



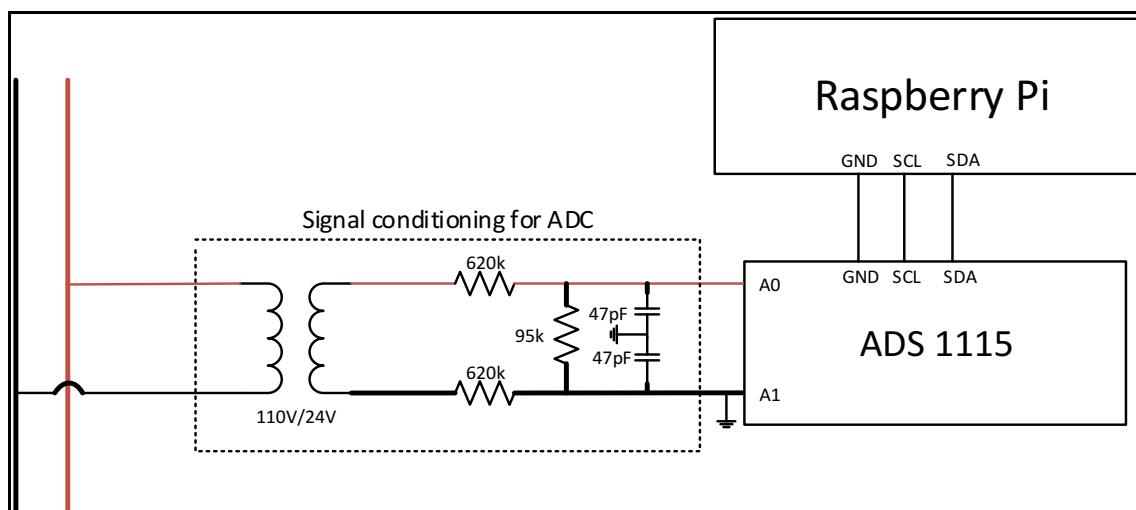


FIGURE 3.3: Schematic of a voltage sense circuit used for prototype design

### 3.1.3. Current Sensing Circuit

Current sensing circuit is different from voltage sensing circuit. There are two ways by which current flowing in a circuit can be sensed. These methods are classified as Direct and Indirect. Direct current sensing is achieved by placing a shunt resistor in series with the load and the current flowing through the system is calculated by measuring the voltage across the shunt resistor (ohm's law). Second method to measure current flowing through a circuit by placing a coil around the conductor and measure voltage that is induced on the coil. This is also known as non-intrusive and it offers an isolated solution where in the sensing circuit is and the system do not have any direct connection [32]. For proof of concept, an isolated current sensing circuit was built. A CR 9580-20 hall-effect current sensor was used. This sensor measures AC current and gives a proportional dc voltage output across a load resistance. The sensor is capable of measuring up to 20Amps AC current and the maximum dc voltage at the maximum input current is 5 volts. Since regular household appliances have a maximum current rating of under 15 amperes, peak output from the sensor will be below the maximum input for the ADC. Load resistor was chosen

to be  $1\text{M}\Omega$  as recommended on the datasheet of the sensor. Similar to voltage sensing circuit, positive output was connected to ADC pin A2 and negative was grounded and connected to ADC pin A3. The current sense circuit diagram is shown below:

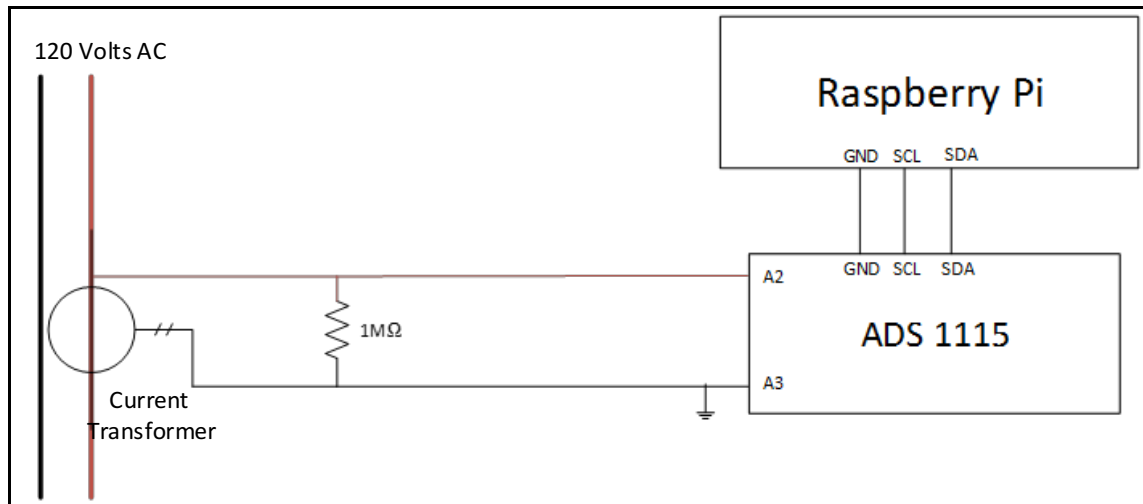


FIGURE 3.4: Schematic of a current sense circuit used for prototype design

Complete circuit to measure the voltage and current and the ADC interface with the Raspberry Pi is shown below. ADC receives analog inputs from voltage and current sensing circuits and converts it to digital values and transmits it to the Raspberry Pi through an I<sup>2</sup>C interface.

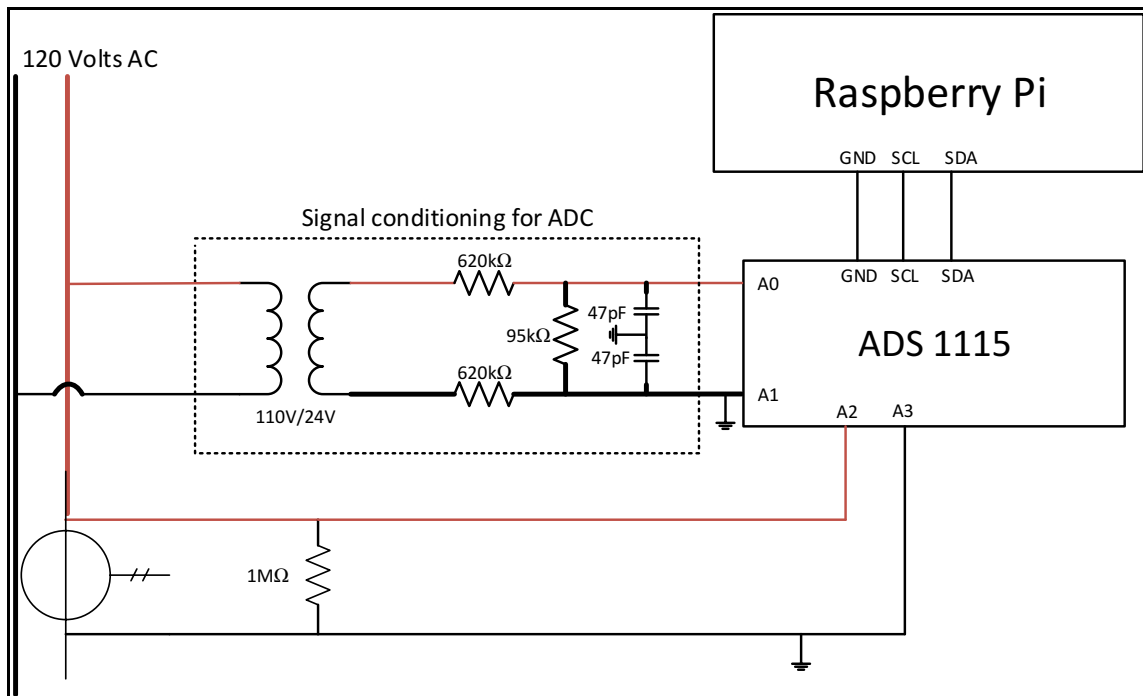


FIGURE 3.5: Schematic of sensing circuits along with raspberry pi

A Python code to measure voltage, current and power ( $V \cdot I$ ) runs on Raspberry Pi by taking the input it receives from the ADC. A snippet of the code which reads the voltage and current values is seen below.

```
# Differential reading of channels 2 and 3; and channels 0 and 1
currentdiff = adc.readADCDifferential01(4096, 860)/1000.0
voltsdiff = adc.readADCDifferential23(4096, 860)/1000.0

# Display the two different reading for comparison purposes
print "%.8f %.8f" % (-currentdiff, -voltsdiff)
```

Values for Voltage (V) and Current (I) are seen on the Raspberry Pi when code is run. Changes to the code are made in order to calculate Root Mean Square (RMS) values

of voltage ( $V_{rms}$ ) and current ( $I_{rms}$ ). From instantaneous readings, RMS values can be calculated by using given formulae:

$$V_{rms} = K_v \sqrt{\frac{\overset{\text{no.ofSamples}}{\sum_{n=1}^n V_{ph}^2(n)}}{n}} \quad (5)$$

$$I_{rms} = K_i \sqrt{\frac{\overset{\text{no.ofSamples}}{\sum_{n=1}^n I_{ph}^2(n)}}{n}} \quad (6)$$

$$Power = K_p \frac{\overset{\text{samples}}{\sum_{n=0}^n V(n)I(n)}}{\text{samples}} \quad (7)$$

In the above expressions,  $V(n)$  and  $I(n)$  are instantaneous samples of voltage and current,  $n$  is the number of samples that are taken to calculate RMS values.  $K_v$  and  $K_i$  are scaling factor values for voltage and current respectively. RMS values are calculated using above expressions by implementing it in Python code. Output was tested with different loads such as light bulbs, table fans, laptops, etc. and results were found to be accurate.

Voltage, current and power values calculated in Python program are saved to text files `voltage.txt`, `current.txt` and `power.txt` on the Pi. These files are given required permissions to be accessed by the webpage and android application. A relay is chosen to control the on/off of the devices whose voltage and current values are being measured. Power Switch Trail 2 isolated relay is chosen as it can be controlled by the GPIO pins of Raspberry Pi.

### 3.1.4. Webpage Design

Raspberry Pi is a single board computer that runs a version of the complete Linux operating system. It is capable of performing most of the operations that any regular computer can. An apache server was setup on the Pi to create a webpage that can be accessible from any other device. GPIO pins of the Raspberry Pi can be controlled over network by evoking scripts to activate or deactivate them. A basic interface to control on/off of the socket and view voltage, current and power consumed by connected appliance is designed using Hyper Text Markup Language (HTML) and PHP. HTML is the standard language that web browsers use to interpret text, images and other visual materials [61]. PHP is a scripting language that is generally used on the server side for web development [62]. PHP can also be used as a general purpose programming language. PHP code can be easily embedded into HTML code which increases the functionality of HTML webpages. A server is installed on raspberry pi in order to make data and code scripts accessible over the network. This can be accomplished by installing an Apache server on both the raspberry pi that is part of the socket and the one that is the server for the entire system. Apache server is installed along with PHP by following the steps in [63].

1. First step involves installing Apache package by typing the below command.

```
sudo apt-get install apache2 -y
```

After a successful installation, it can be verified by entering IP address of the raspberry pi in a web browser. It should show an image similar to the one below if Apache was successfully installed.

2.Second step involves installation of PHP and Apache package by executing the below command in a terminal window.

```
sudo apt-get install php5 libapache2-mod-php5 -y
```

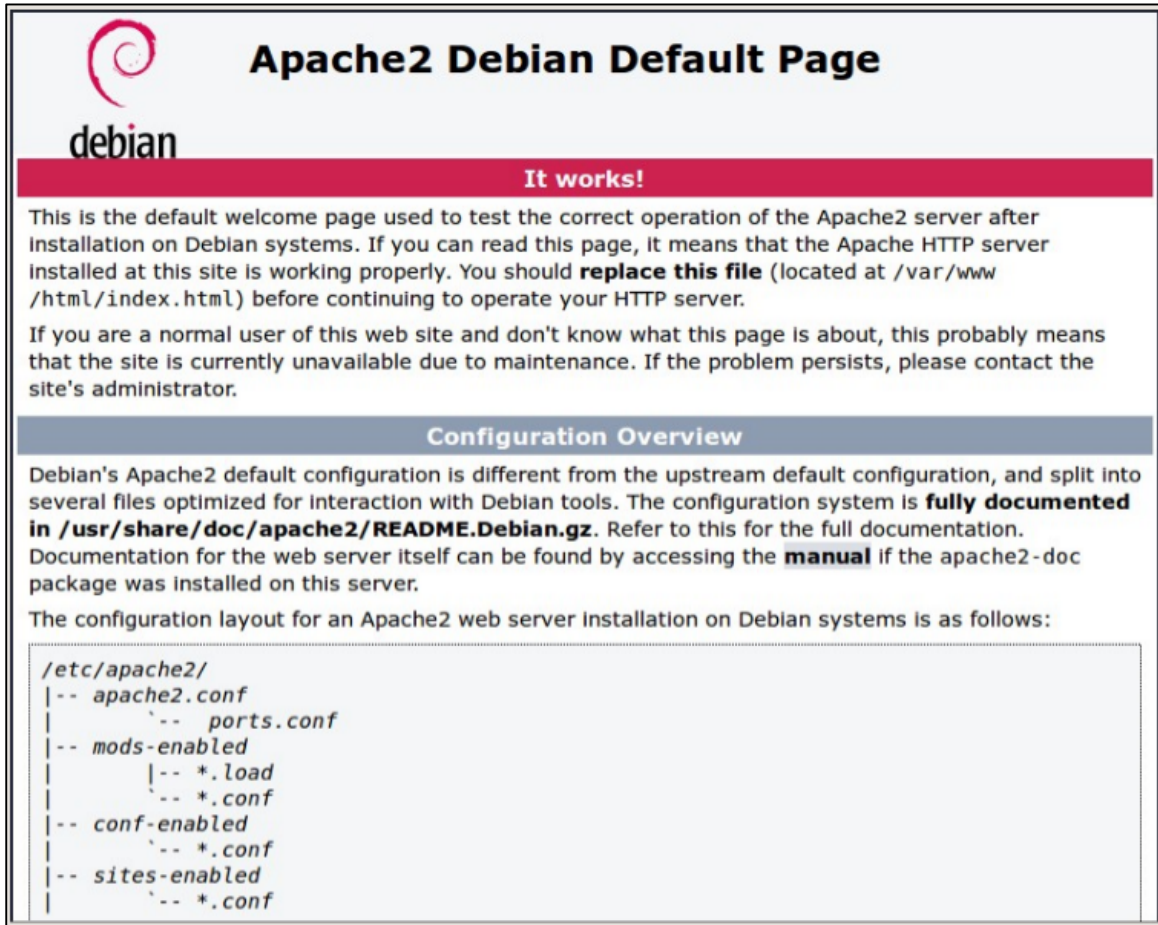


FIGURE 3.6: Default apache web server page

A script titled `page2.php` was designed with buttons to turn on, turn off the socket and three buttons to show the voltage, current and power values. Metering data is obtained from Python programs that were described in the previous section.

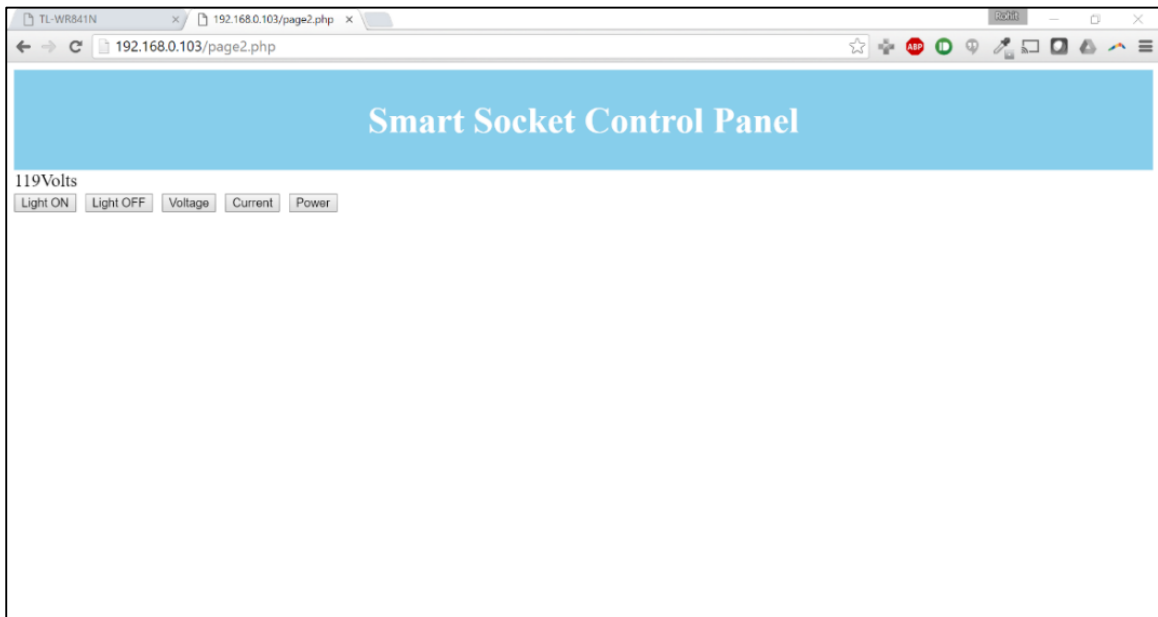


FIGURE 3.7: Smart socket prototype user interface

Python scripts save measured values of voltage current and power to a text file on the Raspberry Pi and PHP functions access these files and display values to the user on the webpage. PHP code which shows voltage read from value stored in raspberry pi can be seen below.

```
if (isset($_POST['Voltage']))
{
$file = "voltage.txt";
$fh = fopen($file,'r');
$data = fread($fh,filesize($file));
fclose($fh);
echo $data;
echo Volts;
}
```

### 3.1.5. Server and Database Installation

A database is created on the raspberry pi that is acting as a server. This database contains tables that have user login information and other details. This database also acts as a back end user verification to the android application that is discussed in chapter 5. A table called “users” is created that contains usernames, passwords and other contact details of users. A user can create an account by filling a registration form in the android application. This table is first created in the database using a PHP script which is executed from a browser window. Once this script is successfully executed, it displays “Done with creation!” message. This table can be verified by logging into the database using phpmyadmin. A screenshot of the table in the database along with the PHP script used to create it is shown below.

```
// PHP script to create a “user” database and a “userlist” table in the database.
<?php
$con = mysql_connect("localhost","root","raspberrypi");
if (!$con){ die('Could not connect: ' . mysql_error());}
mysql_query("DROP SCHEMA USERS");
// Create database
if (!mysql_select_db('users', $con)) {
    echo("Creating Database 'users'!\n<br>");
    if (mysql_query("CREATE DATABASE users",$con)){ echo"Database created<br>";}
else { echo "Error creating database: " . mysql_error() ."<br>";}
    mysql_select_db("users", $con);
}
$sql= "CREATE TABLE userlist
(
```



```

userID int(6) NOT NULL AUTO_INCREMENT ,
username varchar(20) DEFAULT NULL ,

```

```

password varchar(20) DEFAULT NULL,
firstname varchar(20) DEFAULT NULL,
lastname varchar(20) DEFAULT NULL,
email varchar(20) DEFAULT NULL,
phone          varchar(12) DEFAULT NULL,
domain varchar(30) DEFAULT NULL,
PRIMARY KEY (`userID`)
)";

```

```
// Execute query
```

```
mysql_query($sql);
```

```
echo("Done with creation!\n<br>");
```

```
mysql_close();
```

```
?>
```

The screenshot shows the phpMyAdmin interface for the 'users' database. The 'userlist' table is selected, and the SQL query 'SELECT \* FROM `userlist`' is executed. The results show 4 rows of user data:

	userID	username	password	firstname	lastname	email	phone	domain
<input type="checkbox"/>	1001	a	b	aba	abi	abi@gmail.com	7045545454	grid
<input type="checkbox"/>	1002	rsheshadr	rohit007	rohit	seshadri	rohiteshadri92@gmail	7034790734	NULL
<input type="checkbox"/>	1003	abcd	abcd	tulsi	burle	tburle@uncc.edu	7049573625	null
<input type="checkbox"/>	1006	al234	password	rahul	agarwal	rah.agarwal31@gmail	superman	null

The interface includes a sidebar with navigation options, a top menu with actions like 'Browse', 'Structure', and 'SQL', and a bottom section for 'Query results operations' and 'Bookmark this SQL query'.

FIGURE 3.8: A screen shot of the userlist database used by android application for user validation

The above figure shows “userlist” table which was created in “users” database using above given PHP code. Contents of the table were added by registering users using the android application.

### 3.2. Arduino as Energy Meter:

After establishing initial setup and proof of concept, a decision was taken to change energy measurement from Raspberry Pi to an Arduino Uno board. Arduino board has more real time and analog capability compared to raspberry pi which is important when measuring energy utilization. Measurements such as voltage, current and power were obtained from the raspberry pi setup described in above sections. But in order to calculate reactive power, active power and power factor, raspberry pi hardware would have to run a continuous scripts and dedicate more resources towards this task. This might cause latency in turn on/off tasks which would make the product unsatisfactory to use. Therefore, a low cost and small size Arduino board [24] was used for energy measurement. This board was interfaced to the raspberry pi via Universal Serial Bus (USB) port.

#### 3.2.1. Arduino Setup

Since Arduino does not have an operating system that runs on it like the raspberry pi, it requires an Integrated Development Environment (IDE) that is available for download [<https://www.arduino.cc/en/Main/Software>]. Arduino has a vast developer community with many open source libraries and projects. Arduino software IDE consists of a text editor to write code, a message area, text console and a toolbar with common functions and menus.

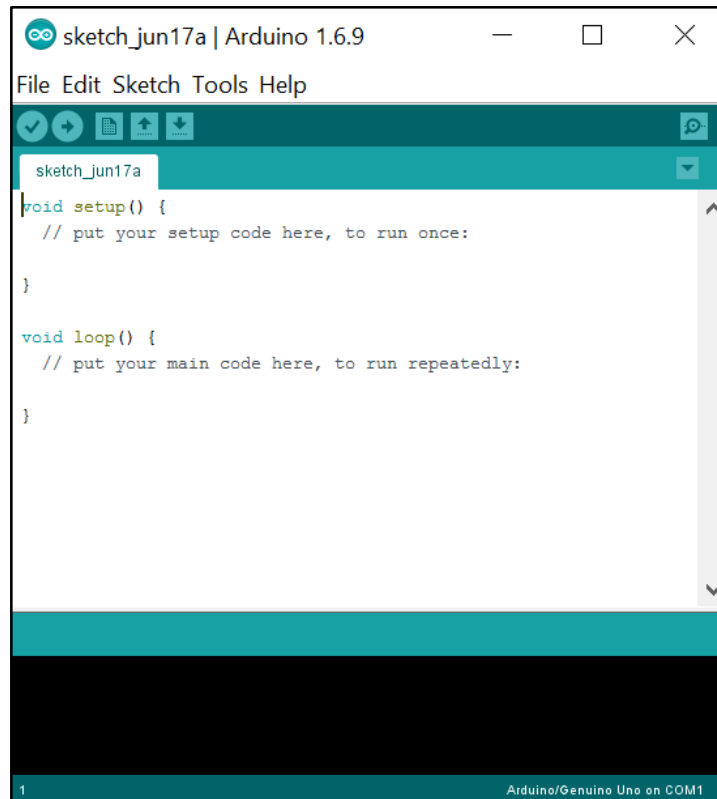


FIGURE 3.9: Structure of an Arduino sketch seen in the Arduino IDE

Programs are written in C/C++ and are known as sketches. After every sketch is written, the code is first checked for errors using the compile option and then uploaded to the board using a USB cable. A typical program contains two parts, first is a setup method (`void setup()`) that is called once every time the board is powered up. It is used to initialize variables, incorporate libraries and pin modes [24]. Second part of an Arduino program consists of a loop method (`loop()`) that continuously runs the program.

```
// Setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}
```

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

As seen from the code above, pin 13 is set as output mode in void setup() function. Pin 13 is then turned on and off every second and repeated until Arduino is powered off as it is in loop() function. Functionalities of Arduino board can be increased with use of libraries. Libraries make it easy to interface hardware and can be added by using import library option. Apart from standard libraries that come with the IDE, additional user installed libraries can be added by downloading them from places such as Github, etc. For this thesis, Libraries were installed from Github that were developed for the OpenEnergyMonitor project. OpenEnergyMonitor project was founded out of a desire for open-source tools (Licence GNU GPL V3) to help us understand and relate to our use of energy, our energy systems and the challenge of sustainable energy [64]. OpenEnergyMonitor provides vast resources on how to build electricity monitors with good theory, Arduino libraries and hardware examples. EmonLib library is used to perform Voltage and Current sampling and calculations. This library was downloaded from Electricity monitoring library, found on GitHub at [65]. Downloaded file was renamed to EmonLib and extra folders present were removed. This file was then saved in Arduino libraries folder. Once Arduino IDE is restarted, EmonLib examples can be loaded from within the IDE.

Library contains examples to measure either only voltage or both voltage and current. EmonLib is included and used in loop() method to calculate real, apparent power, voltage and current. Arduino uses its analog inputs to receive voltage and current signals and performs calculations to show required results.

### 3.2.2. Hardware Interface

Arduino Uno has an ATmega328 microcontroller with six analog input pins which can be used to measure voltage from ground up to a maximum of 5 volts. An analog front end is built to convert the input voltage and current signals to a level that is within safe limits for Arduino board. Two separate circuits are built; one for voltage sensing and one for current sensing.

### 3.2.3. Voltage Sensing Circuit

Typically, mains voltage varies between 119-121 volts. This voltage is too high to directly supply to Arduino analog inputs. Therefore, a circuit is built to stepdown this voltage to under 5 volts. A 120V/5V transformer is used to step down 120 AC volts to 5 volts AC. Resulting wave is around 6 volts with a positive peak of +6 and negative peak of -6 volts. This wave is scaled down with help of a voltage divider and an offset is added to remove negative component of the wave. Voltage divider is designed to limit peak voltage to around 1 volt. Resistor values is chosen as  $R_2 = 60k$  and  $R_1 = 10k$  and resulting peak voltage is calculated as follows.

$$V_{pk\ pk} = \frac{R_1}{R_1 + R_2} * V_{pk\ in} \quad (8)$$

$$V_{pk\ pk} = \frac{10k}{70k} * 7.07 \quad (9)$$

$$V_{pk\ pk} = 1.01V \quad (10)$$

A voltage bias is provided to this signal by resistors R3 and R4. This bias is half of Arduino supply voltage. To achieve this, R3 and R4 have to be of equal magnitude and high value to reduce energy consumption. 470K ohm is used as R3 and R4.

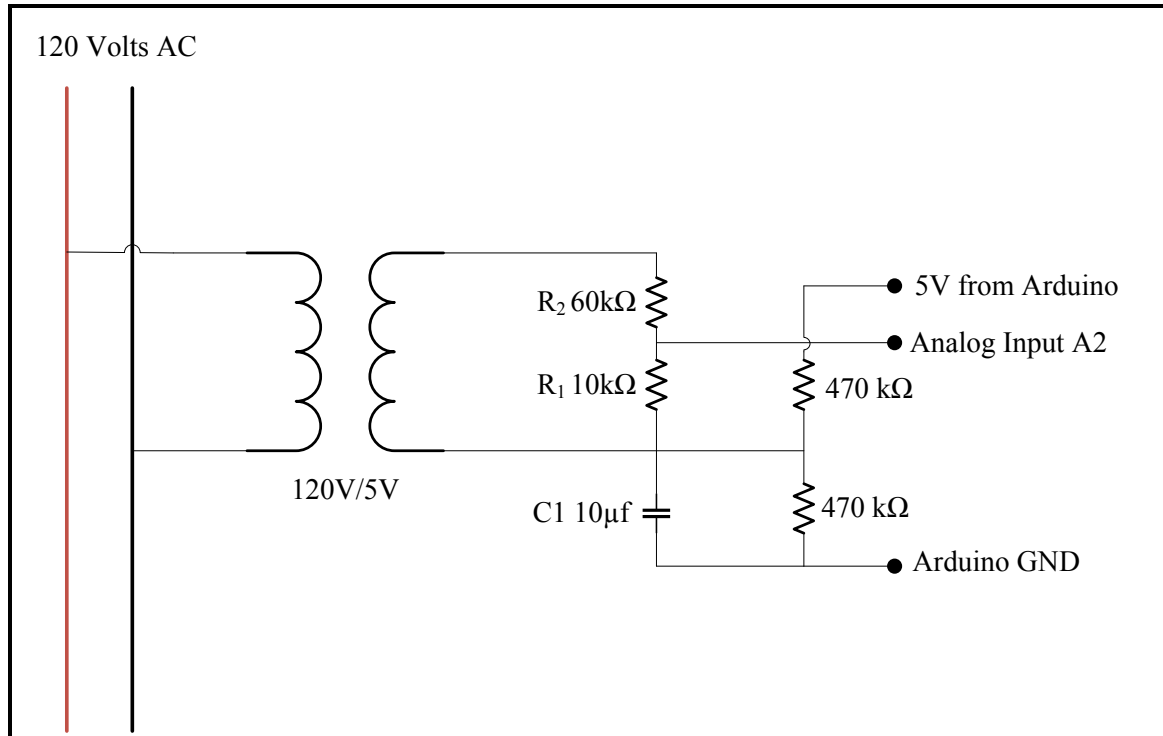


FIGURE 3.10: Schematic of voltage sensing circuit used for Arduino microcontroller

After adding a bias, the resulting output has a positive peak of  $2.5V + 1.01V = 3.6V$  and a negative peak of  $2.5V - 1.01V = 1.49V$ . These values are under maximum input voltage value of 5 volts. A small capacitor C1 ( $10\mu f$ ) is added to provide a low impedance path to ground for AC signal.

#### 3.2.4. Current Sensing Circuit

Current consumed by loads cannot be measured like voltage. A current transformer is an instrument transformer that reduces high voltage current to a lower value which can

be safely monitored. An LF 205-S current transducer was used for this thesis. It is a closed loop hall-effect current transducer with excellent accuracy and high immunity to external interference. Maximum input current for this transducer with 12V supply is 200Amps and can go up to 420Amps depending on the load resistance value. This high current transducer was later replaced with a smaller current sensor. Load resistance for this transducer calculated from its data sheet for maximum of 50Amps was found to be 150 $\Omega$ . Similar to voltage sensing circuit, a DC bias is added to keep the signal within Arduino's range.

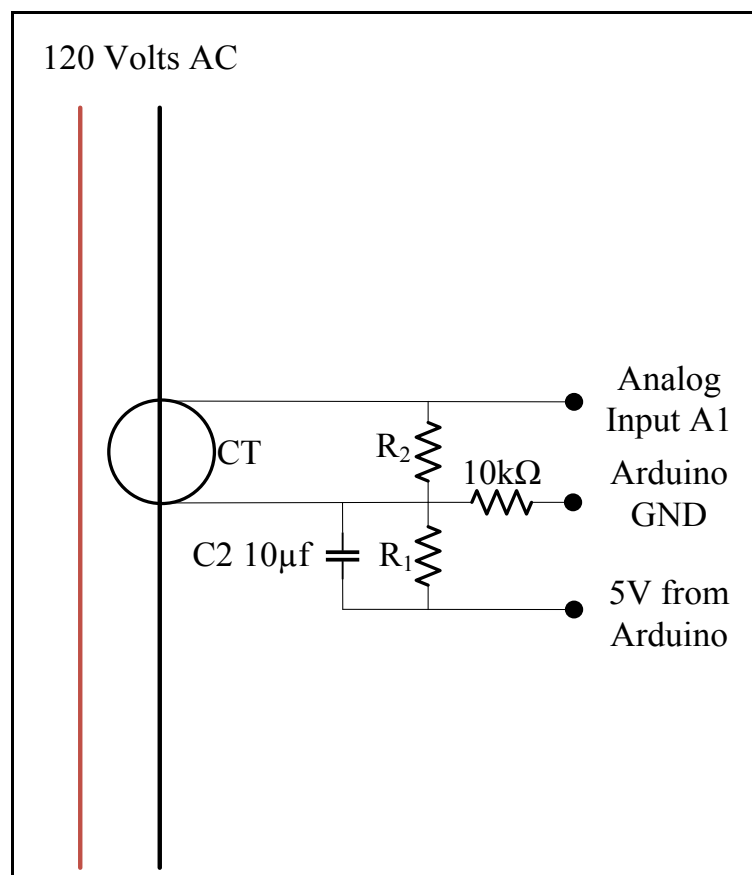


FIGURE 3.11: Schematic of current sensing circuit for Arduino microcontroller

Voltage and Current sensing circuits are interfaced with Arduino analog inputs. For voltage sensing circuit, output from the voltage divider is connected to analog input A2 and output from biasing circuit is connected to Arduino ground terminal. Similarly, for

current sensing circuit, output from the voltage divider is connected to analog input A1 and output from biasing circuit is connected to Arduino ground terminal. After all connections are made, Arduino is connected to a computer and necessary code is loaded onto it. A serial data screen shows output from Arduino based on output statements from uploaded code. An example code is uploaded for calibration purposes as seen below.

```
// EmonLibrary examples openenergymonitor.org, Licence GNU GPL V3
#include "EmonLib.h"      // Include Emon Library
EnergyMonitor emon1;     // Create an instance

void setup()
{
  Serial.begin(9600);
  emon1.voltage(2,146, 4.3); // Voltage: input pin, calibration, phase_shift
  emon1.current(1, 22.2);   // Current: input pin, calibration.
}

void loop()
{
  emon1.calcVI(20,2000);   // Calculate all. No.of half wavelengths (crossings), time-out
  emon1.serialprint();     // Print out all variables (realpower, apparent power, Vrms, Irms, power
  // factor)

  float realPower    = emon1.realPower;    //extract Real Power into variable
  float apparentPower = emon1.apparentPower; //extract Apparent Power into variable
  float powerFactor  = emon1.powerFactor;  //extract Power Factor into Variable
  float supplyVoltage = emon1.Vrms;        //extract Vrms into Variable
  float Irms         = emon1.Irms;         //extract Irms into Variable
}
```



Void setup consists of two statements: `emon1.voltage(2, 234.26, 1.7);` and `emon1.current(1, 111.1);` which refer to functions in EmonLib that setup pins on Arduino for voltage and current sensing. The syntax for voltage sensing function is “voltage (pin number, calibration, phase shift)” and for current is “current (pin number, calibration)”. Calibration values for voltage was obtained by using trial and error method. A multi-meter was used to measure actual mains voltage and calibration value was adjusted in Arduino code to make it as close to mains voltage. Calibration for current was performed in a similar manner. Calibration constant was adjusted so that current reported from Arduino is same as measured current. For phase shift correction, a resistive load was connected to the circuit and phase shift value was altered to get same real and apparent power readings from Arduino.

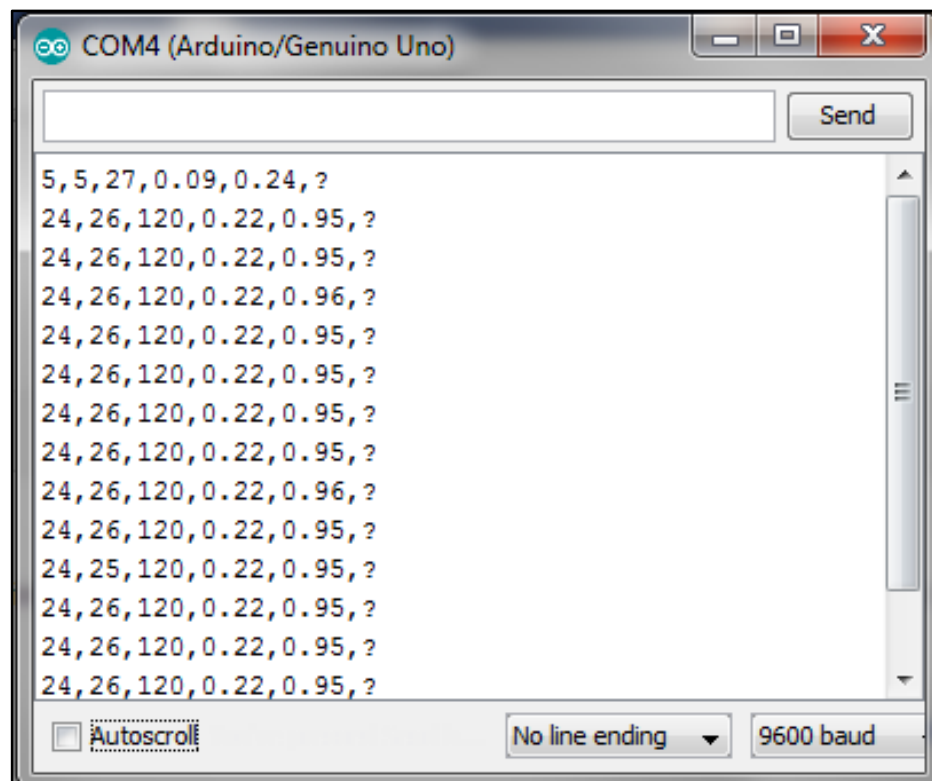


FIGURE 3.12: A screen capture of Arduino serial data monitoring screen

### 3.2.5. EmonLib Calculations

EmonLib library is used to calculate values for voltage, current, real power, apparent power and power factor. Void setup() function which contains voltage and current setup functions, provide EmonLib the pins that are to be used for sensing along with the calibration values required for calculations. Void loop() function consists of a statement that calls a function in EmonLib to perform required calculations. The statement “emon1.calcVI(20, 2000)” calculates real power, apparent power, power factor, RMS voltage and RMS current from a sample window of mains AC voltage. Size of this window is defined by number of half wavelengths which in this case is 20. Before starting the sampling process, Arduino waits for input sine wave to be close to zero which is mid-scale level for the ADC. Once the wave reaches zero crossing, voltage and current values are sampled based on number of half wavelengths and number of loops completed. After sampling voltage and current values, digital filtering is applied to remove offset that was applied and center the wave to zero. These filtered values of voltages and currents are squared, added. Voltage values are later phase shifted to bring it in phase with current. Instantaneous power value is calculated by multiplying phase shifted voltage values with filtered current values and consecutive values are added. RMS values for voltage and current are calculated based on number of times voltage wave crossed initial voltage value as shown below:

$$V_{ratio} = VCAL * \frac{\text{supply voltage}}{1000 * ADC_{count}} \quad (11)$$

$$V_{rms} = V_{ratio} * \sqrt{\frac{\text{sum}V}{\text{No. of Samples}}} \quad (12)$$

In the above expressions, VCAL is voltage calibration constant and sumV is the sum of square of filtered voltage values. Similarly, RMS value of current is calculated as shown below:

$$I_{ratio} = ICAL * \frac{\text{supply voltage}}{1000 * ADC_{count}} \quad (13)$$

$$I_{rms} = I_{ratio} * \sqrt{\frac{\text{sumI}}{\text{No. of Samples}}} \quad (14)$$

ICAL is current calibration constant and sumI is sum of square of filtered current values. After calculating RMS values of current and voltage, EmonLib calculates real power, apparent power and power factor using the following formulae:

$$\text{Real Power} = V_{ratio} * I_{ratio} * \frac{\text{sumP}}{\text{No. Of Samples}} \quad (15)$$

sumP is sum of square of instantaneous power values.

$$\text{Apparent Power} = V_{rms} * I_{rms} \quad (16)$$

$$\text{Power Factor} = \frac{\text{Real Power}}{\text{Apparent Power}} \quad (17)$$

After these calculations are performed, sumV, sumI and sumP are reset to zero and calculated values of Vrms, Irms, apparent power, real power and power factor are returned as serial output using the function “`emon1.serialprint()`”. After verifying measured values with actual values, Arduino was interfaced with the Raspberry pi to have a network connected energy meter.

### 3.2.6. Arduino Interface with Raspberry Pi:

Arduino and Raspberry pi can be interfaced using the GPIO pins, I2C or Serial pins. Using serial communication is easiest and requires the least amount of additional hardware.

A USB cable is used to connect Arduino to Raspberry pi. A plugin has to be first installed on raspberry pi to enable serial read and write operations on it. This can be done by installing pySerial by entering “sudo apt-get install python-serial” in the terminal window [66]. A simple serial communication code is written in python on Raspberry pi to read data that is sent from Arduino.

```
import serial
real_p = 0
apparent = 0
v_rms1 = 0
I_rms1 = 0
factor = 0
ser = serial.Serial('/dev/ttyACM0',9600)
while 1:
    a = ser.readline()
    print a
    def writetofile(text):
        f=open('unknown.txt','w')
        f.write(""+text)
        f.closed
    writetofile(a)
    list = a.split(",")
    realPower = list[0]
    apparentPower = list[1]
    real_p = float(realPower)
    v_rms = list[2]
    I_rms = list[3]
    powerFactor = list[4]
```

```

apparent = float(apparentPower)
v_rms1 = float (v_rms)
I_rms1 = float (I_rms)
factor = float (powerFactor)
def writetofile(text):
    f=open('voltage.txt','w')
    f.write(""+text)
    f.closed
writetofile("%.1f"%v_rms1)
def writetofile(text):
    f=open('current.txt','w')
    f.write(""+text)
    f.closed
writetofile("%.2f"%I_rms1)
def writetofile(text):
    f=open('power.txt','w')
    f.write(""+text)
    f.closed
writetofile("%.1f"%real_p)

```

From the above code it can be seen that entire serial data received from Arduino (real power, apparent power, Vrms, Irms, power factor) is first written to a text file called “unknown.txt” then, the data is split and voltage, current and real power is saved to separate text files. This data is visible to the user from the android application. Data saved in unknown.txt is sent to the server and used for load analysis algorithm. Data from Arduino can now be accessed throughout the network which would have earlier been difficult due to lack of network connectivity on the Arduino board. Data is sent to smart socket server

periodically and can be accessed by users using Smart Socket mobile application. This process is explained in the next two chapters.

## CHAPTER 4: LOAD MONITORING AND RECOGNITION ALGORITHMS

From the beginning of 20<sup>th</sup> century, dependence on coal and natural gas for energy generation has decreased. This is due to increased penetration of solar and wind energy. One major reason for this shift in energy source is global warming. Apart from utilities using clean sources for energy generation, home owners started doing their part to reduce unwanted energy consumption by monitoring their devices. Energy conservation studies show that approximately 5-15% savings can be acquired if people have direct access to their energy usage data. Regular power meters present outside every home gives users their total energy consumption every month and some meters give users their power factor, etc. But these readings do not inform users individual appliance power consumption.

There are different ways to monitor the amount of energy an electrical appliance consumes when it is switched on. Some methods require access into the user's homes and are known as Intrusive Appliance Load Monitoring (IALM) [15] [16]. This method uses sensors and metering equipment to be attached to individual appliances to monitor its energy usage. Another method used to perform load analysis is known as Nonintrusive Appliance Load Monitoring (NALM) [15]. This method requires little to no access into homes and is achieved by placing sensors and data accumulators outside homes near the utility power meter. A brief explanation of these techniques is given in this chapter.

#### 4.1. Non-Intrusive Appliance Load Monitoring (NALM)

One of the earliest implementation of NALM was done by George W. Hart [15]. NALM is able to monitor all the loads on the circuit of a typical home without the need to place individual sensors on any individual appliance. The concept is based on consumption of real and reactive power at any particular time. By calculating the total power consumed by the home, NALM can predict what devices are connected inside the user's house. NALM consists of a meter extension that is placed as an extension to the connector at the main meter box. This device captures signatures over a period of time and analysis is performed on these signatures to determine what device is connected by detecting step changes in real and reactive power consumption. During the setup process, all the appliances in the user's home are turned on and off individually in order to record the device signature. A name for each device is entered manually into the measuring device. Once all devices are mapped, NALM automatically detects future turn on and turn off events of the devices and stores the data for future energy consumption analysis.

Different methods exist for NALM signature recognition. Analysis can be done based on different load signatures applied to the analysis process. Appliance signature can be analyzed by taking different signatures into consideration. Non-intrusive signatures of appliances are broadly divided into steady state, transient state and other signatures.

##### 4.1.1. Steady State Signatures

Steady state analysis is based on step changes in active and reactive power consumption values form one constant value to another [16]. Steady state change is detected when an appliance is turned on or off which triggers a change in the total power consumed. This change in power is then compared with the stored values and if a match is



found, the device is identified. The power consumed by a device when turned on, even in steady state varies as the voltage supplied to the customer could vary by a range of 10% and as a result the power which is the product of voltage and current can vary by a range of 20 % [15] [17]. To overcome these changes in measured values, normalized power calculations are used.

$$P_{\text{norm}}(t) = [V_{\text{ref}}/V(t)]^2 \cdot P(t) \quad (18)$$

$$Q_{\text{norm}}(t) = [V_{\text{ref}}/V(t)]^2 \cdot Q(t) \quad (19)$$

In the above equations,  $P_{\text{norm}}(t)$  and  $Q_{\text{norm}}(t)$  is calculated normalized values of real and reactive power.  $V_{\text{ref}}$  is expected voltage value that the utility must provide (120Volts).  $V(t)$  and  $P(t)$  is actual measured value obtained from the sensors.

#### 4.1.2. Harmonic Frequency Signatures

Steady state analysis can provide accurate detection for most of the devices that are resistive in nature whereas devices that are inductive and capacitive can draw power in a similar way and cannot be separated. Additional information about devices can be determined based on harmonic analysis. Most of the appliances do not draw power linearly with respect to the supply voltage. This is due to the presence of power electronic devices which are known to produce harmonic currents. Harmonic currents can be used to identify appliances that draw similar power from the fundamental components [18]. Since NALM measures the power consumed by all devices in the house, it is possible to miss certain devices that consume low power. Steady state and harmonic analysis can be used in combination to achieve better accuracy.

#### 4.1.3. Transient Signature

Transient signature detection requires high frequency metering capabilities in order to capture the quick transient changes that the devices produce. These transient changes can occur when devices are turned on or when certain devices change their operating states. Transient signatures appear for a very short period of time and often, highly sensitive A/D converters that sample at frequencies close to 100MHz are required to accurately capture these signatures [19]. Such devices are expensive and add to the cost of the NALM which might make it un-economical. Therefore, most of the NALM technologies use steady state and harmonic current analysis to monitor connected device.

#### 4.1.4. NALM Algorithm

NALM algorithm implemented by G.W. Hart consisted steps for measuring, calculations, analysis and tracking. Sensors used by his model were configured to calculate RMS voltage, real power and reactive power from mains by taking samples of voltage and current waveforms [15]. These samples were then used to calculate normalized power as explained earlier, to minimize the effect of changes in utility voltage. A separate edge detection algorithm was used to detect sudden changes in power consumption based on obtained normalized power values. During system installation, all appliances in a user's home are turned on and off individually to establish a database. This database consists individual appliance power changes and is used to detect which appliance is turned on at a later stage. These power signatures are then grouped into clusters based on their values. Similar sets of events are placed in the same cluster. The next step involved building appliance models based on clusters developed. Once these steps were completed, an

appliance could be tracked depending on its energy spike on turn on and determining which cluster it is from.

#### 4.2. Intrusive Appliance Load Monitoring

IALM methodologies offer better insight into appliance load monitoring by placing sensors at every appliance. Individual appliance characteristics can be mapped at greater resolution compared to NALM and can be used to offer better recognition and detection. Typically, energy meters are placed at every electrical outlet at a user's home. These meters perform similar tasks as the NALM measuring device but they can provide information specific to a single device compared to the comprehensive information received using NALM. Also, individual circuits of a home can be monitored and controlled using IALM which is not possible using NALM. Examples of IALM devices are Belkin WeMo devices, Samsung Smart Things power outlet, etc. which were explained in detail in Chapter 1 [12-14].

Smart power socket system is as an intrusive load monitoring scheme as every appliance is monitored individually. The motivation was to develop a system that can be used to monitor appliances in an intrusive manner without the costs associated with a regular IALM. These individual monitoring devices also have the ability to turn On/Off the devices which is not possible to achieve in a NALM scheme. Development was undertaken with an effort to keep cost of subsequent models at a reasonable level.

#### 4.3. Load Recognition Algorithm

G.W. Hart's research set the tone for further research on load monitoring. Advances in NALM include appliance naming, usage of complex algorithms such as neural networks, Nearest Neighbor algorithm, Bayes filters, Fast Fourier transforms, etc. for device

detection and recognition. Most commonly used methods are Bayes filters and Fast Fourier Transforms. Data collection method for all algorithms is similar and as explained earlier. Measuring equipment send information which is extracted based on what is required for the respective algorithm.

#### 4.3.1. Bayes Filter Approach

Bayes filter is a probability based approach to estimate the unknown probability density function from data received using a mathematical process model. Density function is achieved over time after successive repetition of the process. A typical Bayesian network is expressed by the figure above where top half of figure represents the states and bottom half represents respective observations. The states are represented by timestamps and observations are information received from measuring devices (real power, apparent power, etc.) also known as features.

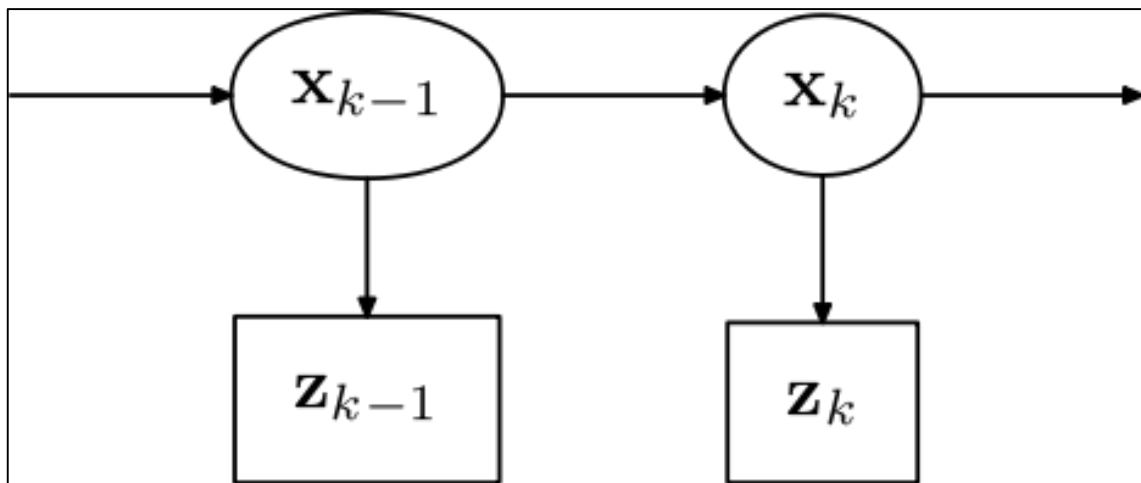


FIGURE 4.1: Representation of a Bayesian network

A constant number of data samples is selected which is known as window size. The window structure is based on a first in first out organization where the data to enter the buffer first is the first to leave. Window data shift by 1 sample for increase in state value.

Contents of a window at any time  $t$  is given as the following for a sample window with seven samples [applying power meters for appliance recognition] is shown below where  $W_{th}$  is total power consumption from time  $t-1$  to  $t$ .

$$Z'_t = \{W_{th}, W_{th-1}, \dots, W_{th-6}\} \quad (20)$$

The model design used states that there exists a pattern for using appliances for example, while preparing a meal, a person uses the refrigerator, microwave and kitchen lights. This pattern is analyzed using a Dynamic Bayesian Network and Markov property and the conditional probability is derived as follows:

$$p(X_t | Z_{1:t}) = \frac{p(Z_t | X_t)p(X_t | Z_{1:t-1})}{p(Z_t | Z_{1:t-1})} \quad (21)$$

$$p(X_t | Z_{1:t}) = \frac{p(Z_t | X_t) \dots_{X_{t-1}} p(X_t | X_{t-1}) p(X_{t-1} | Z_{1:t-10})}{p(Z_t | Z_{1:t-1})} \quad (22)$$

Where  $p(Z_t | Z_{1:t-1})$  is normalized term of the first expression. After performing these analysis, parameter estimation is done using entropy minimization and Gaussian distributions.

#### 4.3.2. k-NN Algorithm

k-Nearest Neighbors Algorithm (k-NN) is an instance based learning method used for classification and regression for pattern recognition. This algorithm is based on predicting unknown values by matching with a list of known values. A simple form of k-NN algorithm is executed by comparing records with known attributes to find the unknown attribute of a new record based on which element is nearest to it. The records generally represent a point in n-dimensional space where n is the number of attributes of that record. When a new record is given, k-NN algorithm searches the n-dimensional space to get k

record that are nearest to the new record and predicts a class label for it based on class labels of these  $k$  nearest records [41]. Nearness of records is determined by the metric distance between them. In most cases, Euclidean distance is used as a measure of how near a record is to another. Euclidean distance is defined as the straight line distance between two points in Euclidean space. For example, if there are two coordinates  $p$  and  $q$  in an  $n$ -dimensional space where  $p = (p_1, p_2, p_3, \dots, p_n)$  and  $q = (q_1, q_2, q_3, \dots, q_n)$  are two points, the Euclidean distance between these points  $d(p, q)$  or  $d(q, p)$  can be calculated as

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \quad (23)$$

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (24)$$

In the above expression,  $n$  is the number of dimensions and  $i$  is number of attributes for each record. A simple example to explain the functioning of  $k$ -NN algorithm is shown below.

Table 4.1: Data used for  $k$ -NN example explained below

PLANT	FLOWERS	THORNS	TYPE OF PLANT
A	3	104	Cactus
B	2	100	Cactus
C	1	81	Cactus
D	101	10	Flower
E	99	5	Flower
F	98	2	Flower
G	18	90	?

The above table shows a list of records known as plants (A-F) and gives the attributes in the form of number of flowers and thorns each plant has. It also classifies them into cactus and flowering plants based on its attributes. This table forms our dataset. A new record 'G' is given in the end where the number of flowers and thorns are given but the type of plant is unknown. By using k-NN algorithm, we can determine the nearest type of plant and safely assume that this unknown plant will fall into this category. First, the Euclidian distance between the unknown record and record A is calculated.

$$\begin{aligned}
 d(? , A) &= \sqrt{(18 - 3)^2 + (104 - 90)^2} \\
 d(? , A) &= \sqrt{(15)^2 + (14)^2} \\
 d(? , A) &= \sqrt{421} \\
 d(? , A) &= 20.5
 \end{aligned}
 \tag{25-28}$$

Similarly, Euclidian distance between unknown record and all known records is calculated and found to be:

$$\begin{aligned}
 d(? , B) &= 18.8 \\
 d(? , C) &= 19.2 \\
 d(? , D) &= 115.2 \\
 d(? , E) &= 117.4 \\
 d(? , F) &= 118.9
 \end{aligned}
 \tag{29-33}$$

If the known records are arranged in descending order based on these calculated distances, the following table is obtained.

Table 4.2: k-NN distances calculated from the data given in table 4.1

Plant	Distance to G
B	18.8
C	19.2
A	20.5
D	115.2
E	117.4
F	118.9

From above table, it can be seen that the nearest neighbor to plant G is plant B which is of type Cactus. Therefore, plant G can be assumed to be a cactus. This can be verified from the original table by checking if the number of thorns in G is greater than number of flowers which is true. K-NN provides a fast and simple method to classify records into classes even if the dataset is very large. The number of nearest neighbors ( $k$ ) can be increased to get a more accurate result for large data sets.

In the above example, both flowers and thorns columns were on the same scale. In certain cases like energy monitoring, each attribute of a record can be in a different scale. Voltage is expressed in Volts, current in Amperes, power in Watts and power factor value is less than or equal to 1. In such cases, attributes with larger values will have a greater effect on final distance between records. To overcome this effect, all attribute columns are normalized using Z-score standardization such that all columns have a mean as 0 and standard deviation of 1. Mean 0 can be achieved by first calculating the mean of the column



and subtracting this mean from each attribute in the column. To set standard deviation to 1, every attribute in the column is divided by the standard deviation of the column. This relationship is given by the formula [42]:

$$x = \frac{x_{ij} - \mu_j}{\sigma_j} \quad (34)$$

Here,  $x_{ij}$  is the  $j^{\text{th}}$  attribute of the  $i^{\text{th}}$  record,  $\mu_j$  is the mean and  $\sigma_j$  is the standard deviation of all values in  $j^{\text{th}}$  attribute. Distances calculated using these normalized values is known as Standardized Euclidean Distance.

K-NN algorithm is implemented using a python script on the Smart Socket server. It is executed from the Android application when a user tries to detect which appliance is connected. Generally, k-NN algorithm is used to classify records based on attribute values but in this case, the algorithm is used to match an unknown dataset received from a power socket with a pre-existing data set that contains a list of all attributes of known devices. The python code uses SciPy which is a Python based open-source ecosystem with software packages for mathematics, science and engineering [43]. The packages used in the code is NumPy and Pandas. Pandas is an open-source library that provides easy to use and high performance data analysis tools for Python language [44]. Data that is used for k-NN analysis consists of 6 attributes namely, real power, apparent power, voltage, current, power factor and device name. In this case, device name is unknown attribute and the algorithm is used to find a record with similar attributes as the record with unknown attribute. Therefore, instead of classifying appliances based on what type of load they are, every device is given a name and the algorithm is used to fetch this name from a list of saved records.

```

#Source: https://www.dataquest.io/blog/k-nearest-neighbors-in-python/
# Modified by Rohit Seshadri
#!/usr/bin/env python
#handle data from csv file
import math
import pandas
with open("knn1.txt", 'r') as csvfile:
    data = pandas.read_csv(csvfile)
print(data.columns.values)
#select data with '?' to match
unknown_device = data[data["Device"] == "?"].iloc[0]
#choose columns to calculate distance
distance_columns = ['Real_power', 'Apparent_power', 'Voltage', 'Current', 'Power_factor']
def euclidean_distance(row):
    """
    Euclidean distance function
    """
    inner_value = 0
    for k in distance_columns:
        inner_value += (row[k] - unknown_device[k]) ** 2
    return math.sqrt(inner_value)
# Find the distance from each device in the dataset to data to be matched.
unknown_distance = data.apply(euclidean_distance, axis=1)
data_numeric = data[distance_columns]

# Normalize all columns
data_normalized = (data_numeric - data_numeric.mean()) / data_numeric.std()
from scipy.spatial import distance
# Fill in NA values in data_normalized

```

```

data_normalized.fillna(0, inplace=True)
# Find the normalized vector for unknown device.
unknown_normalized = data_normalized[data["Device"] == "?"]
# Find the distance between unknown device and other devices.
euclidean_distances = data_normalized.apply(lambda row: distance.euclidean(row,
unknown_normalized), axis=1)
# Create a new dataframe with distances.
distance_frame = pandas.DataFrame(data={"dist": euclidean_distances, "idx":
euclidean_distances.index})
distance_frame.sort("dist", inplace=True)
# Find the device that is similar from the list of known devices
similar_device = distance_frame.iloc[0]["idx"]
same_device = data.loc[int(similar_device)]["Device"]
device_type = data.loc[int(similar_device)]["Power_factor"]
print (data.loc[int(similar_device)]["Device"])
def writetofile(text):
    f=open('devicedetect.txt','w')
    f.write(""+text)
    f.closed
writetofile(same_device)

```

k-NN algorithm code used in smart socket application, calculates distance between unknown record and all other pre-existing records in knn.txt file (Fig. 4.9.). A python script “knn.py” first reads this text file as a Comma Separated Value (CSV) file where attributes in the first line are used as column names [ 'Real\_power', 'Apparent\_power', 'Voltage', 'Current', 'Power\_factor']. All other records are classified based on these column names. All columns are then normalized as explained earlier using “scipy.spatial” library. Euclidean distance between unknown record and other known records is calculated and the

record which is closest (distance equal to zero in this case) is fetched. From this record, attribute under column 'Device' is saved in a text file "devicedetect.txt" which is accessible by the device detect screen in smart socket Android application explained in the next chapter.

## CHAPTER 5: ANDROID APPLICATION DEVELOPMENT

### 5.1. Android Application Development

Android is a mobile operating system which is developed by Google. It is based on a Linux kernel and is used to power many devices such as mobile phones, Television sets, Automobile entertainment systems, smart watches, etc. Currently there are over 1 billion active monthly users of the Android operating system [33]. A large number of Original Equipment Manufacturers (OEM) such as Sony, Samsung, LG, etc. use Android as the OS for their mobile devices, televisions, wearables, etc. As of 2015 Q2 estimates, more than 82% of all smart phones in the world have Android as the OS [35]. Android has undergone multiple revisions adding enhanced features to improve the user interface.

Table 5.1: List of all released Android versions by Google

VERSION	CODE NAME	RELEASE
		DATE
2.2 - 2.2.3	FROYO	20-May-10
2.3 - 2.3.7	Gingerbread	9-Feb-11
4.0 - 4.0.4	Ice Cream Sandwich	16-Dec-11
4.1 -4.3.1	Jelly Bean	9-Jul-12
4.4 - 4.4.4	KitKat	31-Oct-13
5.0 - 5.1.1	Lollipop	3-Nov-14
6.0 - 6.0.1	Marshmallow	5-Oct-15

Some of the major versions of Android released to the public are shown in the table above.

Since Android is open source, developers were given access to the source codes and this allowed them to create applications that add functionality to the device. Java programming language has access to the Android Application Programming Interface (API). Google introduced an application store known as Google Play Store where developers can make their applications available for users to download and install on their devices. Android also supports 64-bit architecture hardware from the 5.0 “Lollipop” version. The minimum memory requirements for devices to run Android 5.1 and higher is 512 MB RAM [34]. Due to low system requirements, ease of use, customizability and high market penetration, an application based on the Android OS was built for the thesis.

## 5.2. Smart Socket Android Application

To enhance user experience and simplify the control procedure, development of an android application was started. The android application serves as a primary control interface for the socket and will allow users to schedule, turn on/off and detect what device is plugged in the socket. Android Studio IDE was used to develop required java and Extensible Markup Language XML files. An android application consists of activities which can be customized based on the functionality required. Android studio allows developers to easily link their layout files with the background java codes. The XML file is used to represent a layout and this layout can be shown on the mobile screen with Java class. Each activity consists of an XML layout file which is represented in Android Studio with a preview UI screen and some default interface elements. When an XML file is

created, a .java class is created for the activity. There are sub directories in the /res folder which offer further customizability. This folder contains the resources that the app uses such as drawables, layout, launcher icons, etc. [36]. An application built in Android Studio can be tested immediately on any android device if it is compatible with the android version of the application. Android studio also has an android emulator which can be downloaded from within itself. Emulator can be used if the developer does not have an android device. For this thesis, a Motorola Moto X (2013) was used for the initial development stages to test the application. This device was later changed to Nexus 6P made by Google. The application was designed to be a user interface to control the power socket an essentially add smart capabilities to an ordinary socket. Therefore, the name for the application was decided as Smart Socket.

#### 5.2.1. Login Activity

Smart Socket interface is simple and user friendly with simple navigation and animations. The application does not have complex menus and uses simple words to describe the actions that can be performed. A home activity was designed first to form as the first screen of the app. A user is asked to enter the login details to access controls for the sockets. If the user is trying to login for the first time, a register button present on the screen allows for the creation of a new account.

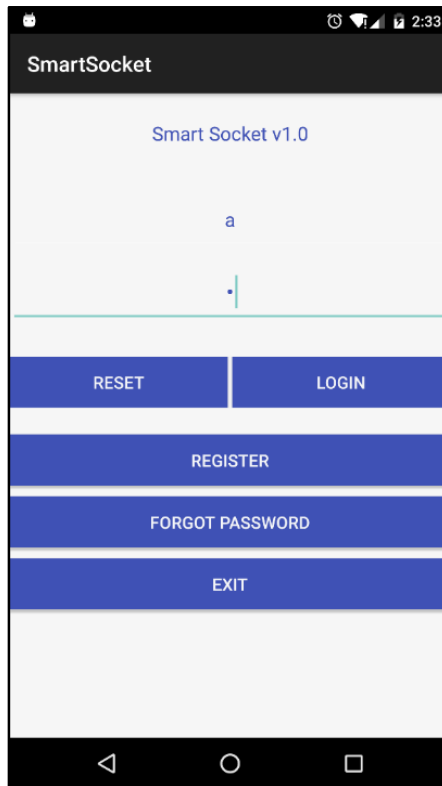


FIGURE 5.1: Screenshot of Smart power socket login screen

Register button opens a new screen in the application with a form that the user has to fill in order to create a new account. After user agrees to all terms and conditions of the application and Smart Socket, register button sends this data to the server where it gets saved to the table which was created earlier. Every time the user tries to login, application sends login information to the server which then compares the information present in the table. If the username and password matches with already present data in the table, server returns “true” and user is allowed to login. This process of checking valid credentials in the database occurs by calling a PHP script on the server when user clicks login button.

```
// PHP code for login verification.
<?php
```



```
mysql_connect("localhost","root","rohit007");
mysql_select_db("users");
```

```
$usr=$_POST["username"];
$pwdIn=$_POST["password"];
$q=mysql_query("SELECT password FROM userlist WHERE username = '$usr'");
while($e=mysql_fetch_assoc($q)){
    $output[]=$e;
    $pwd = $e['password'];
}
if($pwd == $pwdIn)
    print("true");
else
    print("false");
mysql_close();?>
```

### 5.2.2. Sockets Activity

Once the user successfully logs in, a floating action button is present at the bottom right side of the screen to add more sockets to the system. A provision to add a new socket can be implemented by entering a unique code that is provided with the socket. The entered code is checked in the database (from NMAP list) and if an existing socket data matches with entered code, the socket can be added to user list.

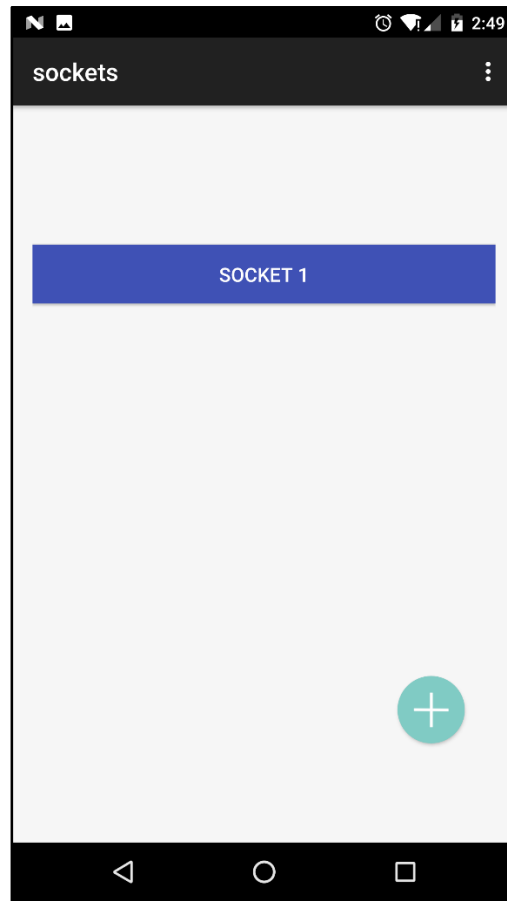


FIGURE 5.2: Screenshot of smart power socket, socket selection screen

Every socket has a dedicated options menu and can be accessed from the above seen list of sockets. A menu with options for Energy monitoring, Scheduling and a Graph view of recent energy usage in the socket. A screenshot of this screen is shown below.

### 5.2.3. Energy Monitor Activity

Energy monitor button opens a new screen which shows the most recent monitoring data received from the socket. It shows voltage, current and power consumed by the appliance connected to the socket.

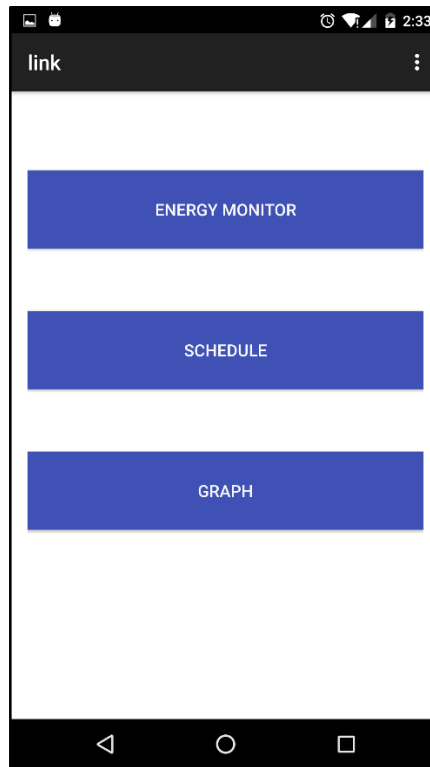


FIGURE 5.3: Screenshot of smart power socket menu screen

Energy monitor screen also has controls to turn the appliance On/Off. A switch widget was used from Android Studio which simulates an actual switch. When the switch in Smart Socket application is turned on, the application sends an “on” command to the socket and starts a PHP script “on.php” that changes the status of the corresponding GPIO pin of raspberry pi. This is done by executing an http post from Smart Socket application. The PHP file executes an escape shell command to execute a python script “on.py”. This python script sends a GPIO high signal to the PowerSwitch Tail 2 relay that is connected to a load. Similarly, when switch in energy monitor screen is toggled to off, a script similar to “on.php” called as “off.php” is executed. This PHP script executes a python script “off.py” which turns off the switch by sending a GPIO low signal to PowerSwitch tail 2. The appliance that is connected is turned on. A toast message is shown in the application

which gives confirmation to the user. A similar process occurs when the switch is changed to off.

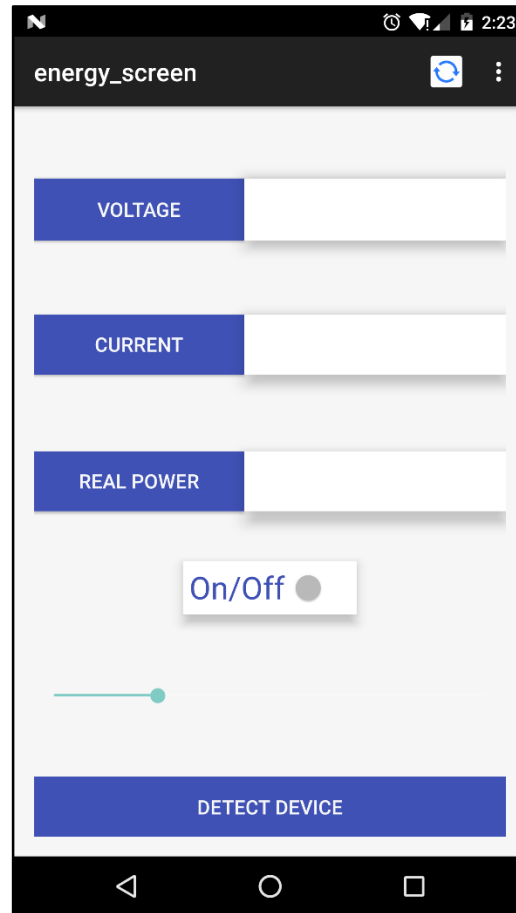


FIGURE 5.4: Screenshot of smart power socket energy monitor screen

An AC phase angle control dimmer is controlled using a seek bar as seen in the above screenshot. AC dimmer module manufactured by sunroom technologies was used [<http://www.sunrom.com/>]. This dimmer module is capable of 256 step control by either parallel method using GPIO pins or serial method using serial communication interface between raspberry pi and AC dimmer module. The input can be simple 8-bit binary signal from microcontroller which is isolated with the use of opto-couplers or serial data input [46].

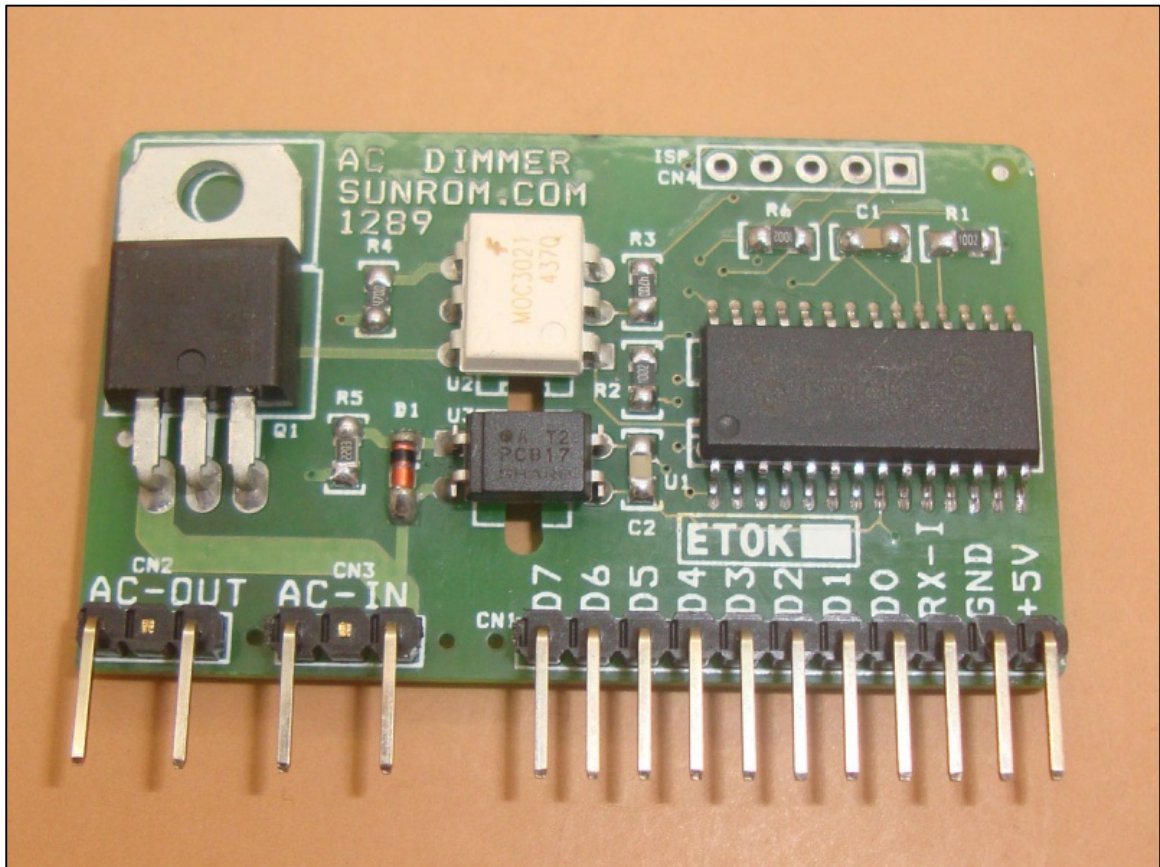


FIGURE 5.6: A picture of AC phase angle control module by Sunrom technologies

The board can be used in applications where dimming of 110-220V AC power is required like dimming of resistive loads and light inductive load. Total of 256 levels of power control can be set from totally off (0%) to full on (100%) as per input control levels. AC dimmer uses a triac to allow only part of AC source waveform to be delivered to the load [47]. Dimming intensity can be controlled by varying firing angle of the triac by changing GPIO inputs to the dimmer module.

TABLE 5.2: Specifications of ac dimmer module by sunrom technologies

Parameter	Pin
AC Input	80V AC to 250V AC
AC Load Current	12 Amp Maximum
AC Load Type	Bulb (Resistive Load) up to 1000 Watts Fan (Inductive Load) upto 100 Watts
Frequency of mains	50 Hz or 60 Hz
Control Input Voltage	5V DC Isolated

#### 5.2.4. Schedule Activity

Users can also schedule when to turn on their appliances from within the application. From menu screen, schedule button opens a new view which contains text fields where the switch on and switch off times can be entered. This feature allows users to automatically schedule appliance usage times based on their daily preferences. For example, they can schedule the common area lights to turn off after dinner time in order to save energy, turn off laptop chargers after certain duration in order to prevent over charging of the batteries.

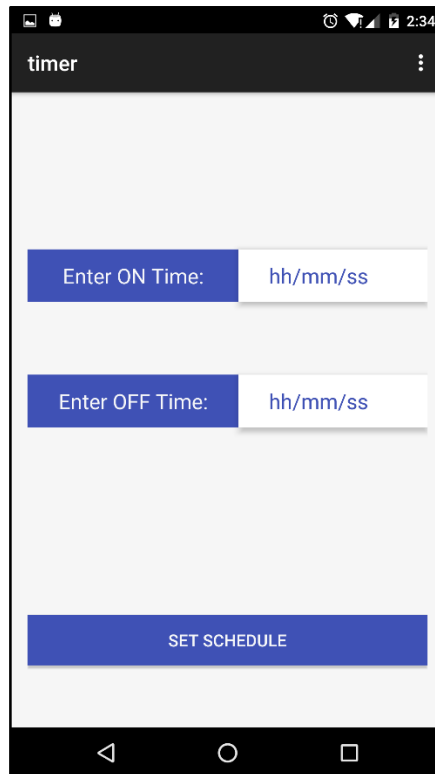


FIGURE 5.5: Screenshot of smart power socket schedule screen

After a user enters on and off time in the text fields and selects “set schedule”, the application parses the text entered into strings and calls an http client to transfer it to the power socket. “Set Schedule” button calls a PHP script that saves the received strings to a text file.

```
<?php
$Start = $_POST["On"];
$Stop = $_POST["Off"];
$file = fopen("time.txt","w");
echo fwrite($file,$Start);
echo fwrite($file,"\n");
echo fwrite($file,$Stop);
fclose($file);
```

```
?>
```

A separate python script “stampmod.py” runs on the power socket that continuously checks if any changes are made to “time.txt” file by monitoring the timestamp it and comparing with current time on the power socket. If any change in timestamp is detected, stampmod.py reads new data in time.txt and saves the new switch on and off times. These new on and off times are compared continuously to current time and when on time and current time is same, a GPIO high signal is sent to the PowerSwitch Tail 2 relay. Similarly, when off time is same as current time, a GPIO low signal is sent to the relay and switch is turned off. The python code “Stampmod.py” can be seen below.

```
#Code to check timestamp changes to file “time.txt” and update On/Off times.
import time
import signal, sys
import datetime
import os
import RPi.GPIO as GPIO
import os.path
from stat import *
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
led = 4
GPIO.setup(led,GPIO.OUT)
    #read file for initial on/off times
file = 'time.txt'
file = open('time.txt','r')
data = file.read()
```



```
#led on time:
onh = data[0:2]

ledh = int(onh)
onm = data[3:5]
ledm = int(onm)
ons = data[6:8]
leds = int(ons)
#led off time
ofh = data[9:11]
ledoh = int(ofh)
ofm = data[12:14]
ledom = int(ofm)
ofs = data[15:17]
ledos = int(ofs)

#print "1: ",ledh,ledm,leds,ledoh,ledom,ledos
#loop to switch on GPIO when timestamp changes
while 1:
    file = 'time.txt'
    last_time1 = os.stat(file).st_mtime
        #loop to check change in timestamp continuously
    while 1:
        #print "1: ",last_time1
        last_time2 = os.stat(file).st_mtime
        #print "2: ",last_time2
        time.sleep(0.5)
    #if timestamp changes, get new on/off times
    if last_time2>last_time1:
        #print "mod"
        #print last_time1,last_time2
```

```
        time.sleep(0.2)
        file = open('time.txt','r')
        data = file.read()

#switch on time:

        onh = data[0:2]
        ledh = int(onh)
        onm = data[3:5]
        ledm = int(onm)
        ons = data[6:8]
        leds = int(ons)

#switch off time

        ofh = data[9:11]
        ledoh = int(ofh)
        ofm = data[12:14]
        ledom = int(ofm)
        ofs = data[15:17]
        ledos = int(ofs)

#print "2: ", ledh,ledm,leds,ledoh,ledom,ledos
        break

#compare given time to current time and switch on GPIO
        dt = list(time.localtime())
        hour = dt[3]
        minute = dt[4]
        second = dt[5]
        time.sleep(0.5)
        #os.system('clear')
        #print hour, minute, second

#switch on logic:
```

```

#print "3: ",ledh,ledm,leds,ledoh,ledom,ledos
    if hour==ledh:

        if minute==ledm:
            if second==leds:
                GPIO.setmode(GPIO.BCM)
                GPIO.setup(led,GPIO.OUT)
                GPIO.output(led,1)

#print "led on"
    if hour==ledoh:
        if minute==ledom:
            if second==ledos:
                GPIO.setmode(GPIO.BCM)
                GPIO.setup(led,GPIO.OUT)
                GPIO.output(led,0)
                #print "led off"

#GPIO.cleanup()
#print last_time1,last_time2
#print"loop end"

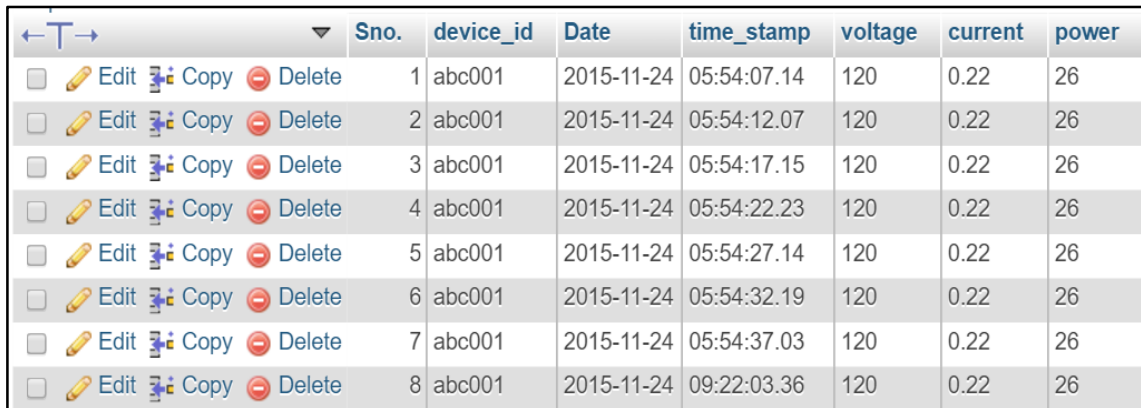
```

Smart Socket gives users the ability to remotely schedule automatic device turn on/off based on pre-determined schedules according to convenience.

### 5.2.5. Graph Activity

A graph activity was created to show power consumed by devices connected to the socket in a graphical manner. A simple x-y axis graph with the x-axis showing time and y-axis showing power consumed in watts. Values for power consumed and respective timestamp is recovered from the database. Power socket sends an updated set of values

every minute to the server. Server contains a record of all information that is received from the power socket in the monitor table. Data for graph is fetched from this table when user refreshes the graph. Two PHP scripts are called by the application which fetches power and respective timestamp values from the database.



	Sno.	device_id	Date	time_stamp	voltage	current	power
<input type="checkbox"/> Edit Copy Delete	1	abc001	2015-11-24	05:54:07.14	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	2	abc001	2015-11-24	05:54:12.07	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	3	abc001	2015-11-24	05:54:17.15	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	4	abc001	2015-11-24	05:54:22.23	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	5	abc001	2015-11-24	05:54:27.14	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	6	abc001	2015-11-24	05:54:32.19	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	7	abc001	2015-11-24	05:54:37.03	120	0.22	26
<input type="checkbox"/> Edit Copy Delete	8	abc001	2015-11-24	09:22:03.36	120	0.22	26

FIGURE 5.6: A screen capture of database user to save appliance energy information

Java code in graph class is written to determine axis magnitude based on power values received from fetched values thus allowing the graph to automatically adjust. Android Studio does not have built in graph widgets. Therefore, additional dependencies were added to import graph views. Graph View is an open source external library for Android to programmatically create diagrams. Custom graph designs can be incorporated using [37]. In order to use Graph View in Smart Socket application, 'com.jjoe64:graphview:4.0.0' was added in build.gradle file into dependencies block. Once the dependencies are compiled, a layout can be created in xml.

```
<com.jjoe64.graphview.GraphView
    android:layout_width="match_parent"
    android:layout_height="200dip"
    android:id="@+id/graph"
    app:seriesType="line"
    app:seriesTitle = "Test Graph"
```

```

        android:title="graph"
        android:background="#3f51b5"
        android:elevation="5dp" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update"
    android:id="@+id/button_update"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:background="#3f51b5"
    android:textColor="#FAFAFAFA"
    android:textSize="16dp" />

```

A new GraphView class was created and in its xml file the above mentioned code was added. It created a graph view widget along with an update button that allows users to refresh the graph. This graph widget was programmed to show 10 most recent values of power consumed and its respective timestamp. Data was fetched from monitor table as an array in a two-step process and was parsed into individual strings. In the first step, two PHP scripts are called from GraphActivity when user selects “Update Graph”. These scripts fetch power and timestamp values from “monitor” table and copy the values to two text files named “pwr.txt” and “time.txt”. Below is the PHP code used to fetch 10 latest timestamps from “monitor” table and store in “time.txt” file. A similar code is used to get power values.

```

//PHP script to get timestamp values from database and store in “time.txt”

<?php

//set the connection variables

$con=mysqli_connect("localhost","root","raspberry","users");

```

```

if (mysqli_connect_errno())
    {
        echo "failed to connect:".mysqli_connect_error();
    }

$sql = "SELECT `time_stamp` FROM `monitor` ORDER BY `Sno.` LIMIT 10";
$result = mysqli_query($con,$sql);

//echo "result is: ".$result.";

while($row = mysqli_fetch_array($result,MYSQLI_NUM)){

printf("%s\n",$row[0]);

$file = fopen("time.txt","a");

$txt = $row[0]." ";

fwrite($file,$txt);

fclose($file);

}

mysqli_free_result($result);

printf("%s\n",$result);

mysqli_close($con)

?>

```

After storing these values, a function in `GraphView.java` fetches these values from the text file to the application as a string. This data is later parsed into individual arrays for each timestamp and power value.

```

//Android Java code used to fetch data from "time.txt" and parse into string.

try {

    URL textfile = new URL(serv_ipkey+"/time.txt");

    BufferedReader in = new BufferedReader(

        new InputStreamReader(textfile.openStream()));

```

```

String inputLine;

inputLine = in.readLine();

Log.v("string", inputLine);

in.close();

ay.setText(inputLine);

String[] array = inputLine.split(" ");

time1 =array[0];

t1 = Float.parseFloat(time1);

Log.v("array0", array[0]);

time2 =array[1];

t2 = Float.parseFloat(time2);

.

.

time10 =array[9];

t10 = Float.parseFloat(time10);

```

Graph View requires the data point values to be in either integer or float data type. Therefor the strings are parsed into float as shown in above code. Once data is in required format, data points can be assigned to the graph widget as shown below.

```

// Code to add new data points to the graph.

com.jjoe64.graphview.GraphView graph = (com.jjoe64.graphview.GraphView)
findViewById(R.id.graph);

LineGraphSeries<DataPoint> series = new LineGraphSeries<>(new DataPoint[] {

    new DataPoint(t1, pwr1),

    new DataPoint(t2, pwr2),

```

```
.  
. .  
. .  
new DataPoint(t9, pwr9),  
new DataPoint(t10, pwr10),  
});  
graph.addSeries(series);
```

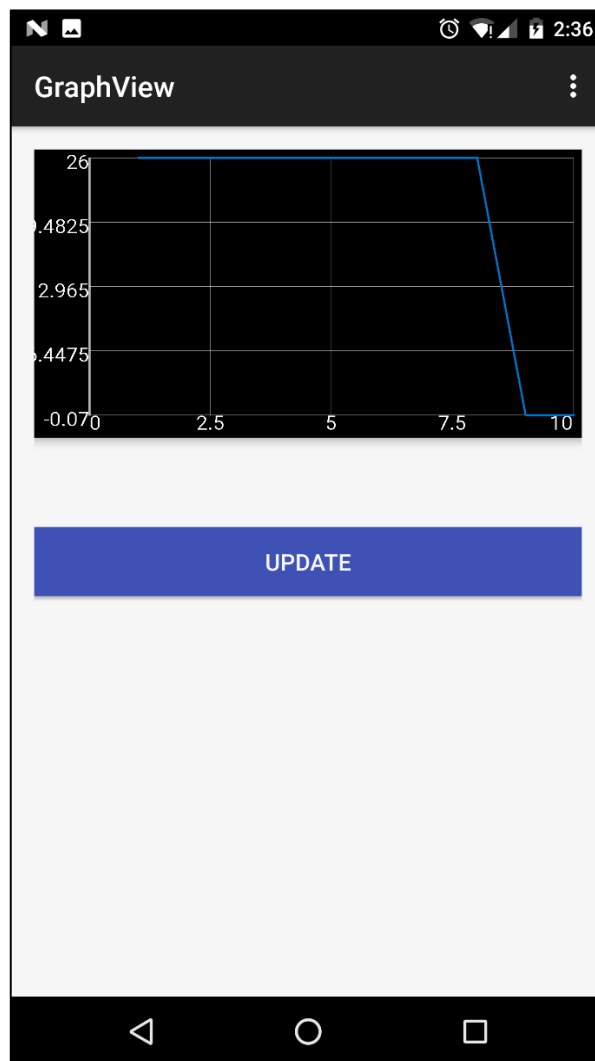


FIGURE 5.7: Screenshot of smart power socket graph screen



A screenshot of the graph is seen above. Update button refreshes the contents and shows a more recent plot based on values obtained from the server.

### 5.2.6. Detect Activity

This activity is called from the Energy Monitor screen. It shows a new screen with two options for users. First option is Detect Device (Fig 5.8) which as the name suggests, tries to detect which device is connected to the power socket. This is done in a sequence shown in the code below, where first a command is sent to the smart socket server to run k-NN algorithm by calling a PHP script. PHP script `knn_mod.php` runs on the server and saves the result obtained in a text file `devicedetect.txt`. A sleep function in the app allows it to wait for a few seconds while k-NN runs on the server.

```
device_detect.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
  
    public void onClick(View v) {  
  
        String url1 = "http://192.168.0.204/data.php";  
  
        HttpClient client1 = new DefaultHttpClient();  
  
        try {  
  
            client1.execute(new HttpGet(url1));  
  
        } catch (IOException e) {  
  
            //do something here  
  
        }  
  
        try{  
  
            Toast.makeText(getApplicationContext(), "Detecting Device", Toast.LENGTH_SHORT).show();  
  
            Thread.sleep(1000);  
  
        }catch (InterruptedException e){  
  
            e.printStackTrace();  
  
        }  
  
    }  
  
}
```

```
}  
  
String url = serv_ip+"/knn_mod.php";  
HttpClient client = new DefaultHttpClient();  
  
try {  
    client.execute(new HttpGet(url));  
} catch (IOException e) {  
    //do something here  
}  
  
try{  
    Toast.makeText(getBaseContext(), "Detecting Device", Toast.LENGTH_SHORT).show();  
    Thread.sleep(5000);  
} catch (InterruptedException e){  
    e.printStackTrace();  
}  
  
try {  
    URL textfile = new URL(serv_ip+"/devicedetect.txt");  
    BufferedReader in = new BufferedReader(  
        new InputStreamReader(textfile.openStream()));  
  
    String inputLine;  
  
    inputLine = in.readLine();  
  
    in.close();  
  
    dev = inputLine;  
  
    device.setText(dev);  
} catch (Exception e) {  
  
    Log.e("log_tag", "device not opened " + e.toString());  
  
    Toast.makeText(getBaseContext(), "devicedetect.txt not opened",
```

```
Toast.LENGTH_SHORT).show();
}
```

Once the result is saved in `devicedetect.txt`, a second PHP script is executed from the application which fetches this result from the server. This result is then displayed in a text field above detect device button. During detection process, unknown device's characteristics are added to a pre-existing device list with its last column left as “?” as seen in Figure 5.9.

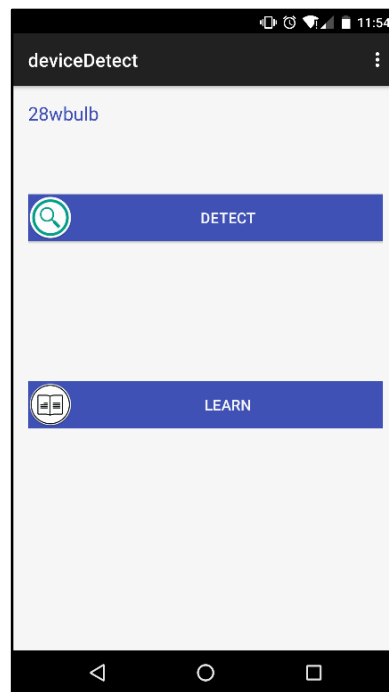


FIGURE 5.8: Screenshot of smart power socket device detect screen

The k-NN algorithm tries to find a record with similar attributes. If the device is successfully detected, its name is displayed. If it is an unknown device, a “?” is shown which indicates that it is a new device and user has to use the learn option to assign a name to it.

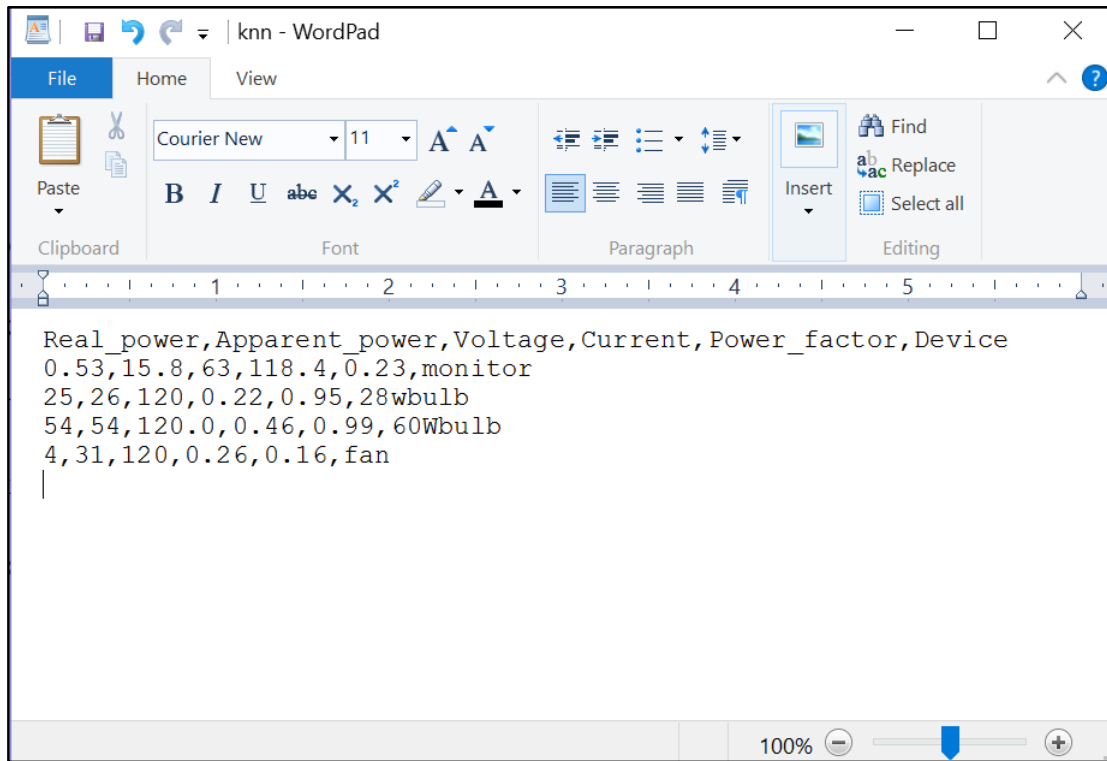


FIGURE 5.9: Screenshot of knn.txt file from smart socket server with the appliance list

### 5.2.7. Learn Activity

A continuation to Detect activity, learn activity is used to allow users to name their devices. This name is saved along with electrical characteristics of the device on the server which can later be used for k-NN algorithm. A name can be associated with a specific record (electric characteristic) by entering its name in a text field as shown in Fig 5.10. This name is appended to the record and saved on the server. Changes are made to knn.txt file where “?” in device column is replaced with the name entered by the user as seen in Fig 5.11.

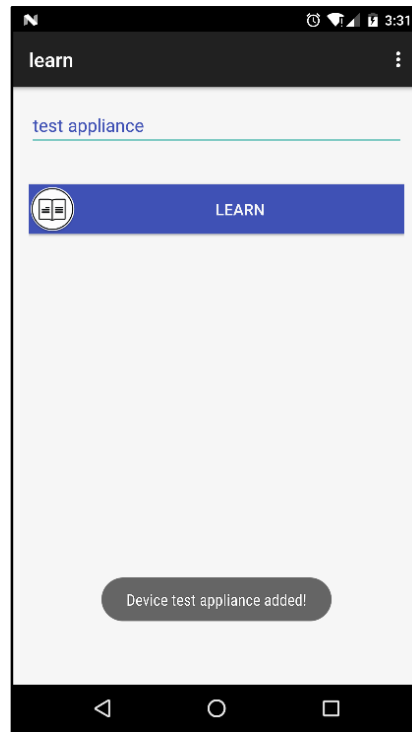


FIGURE 5.10: Screenshot of smart power socket learn screen

Information containing electrical characteristics of an appliance is sent periodically to the server where it is uploaded to a table in database. When learn activity is used, this information is also saved in a text file known as knn.txt which contains a list of all devices saved by the user along with their respective electrical characteristics. Knn.txt is used as a dataset for k-NN algorithm as described in previous section.

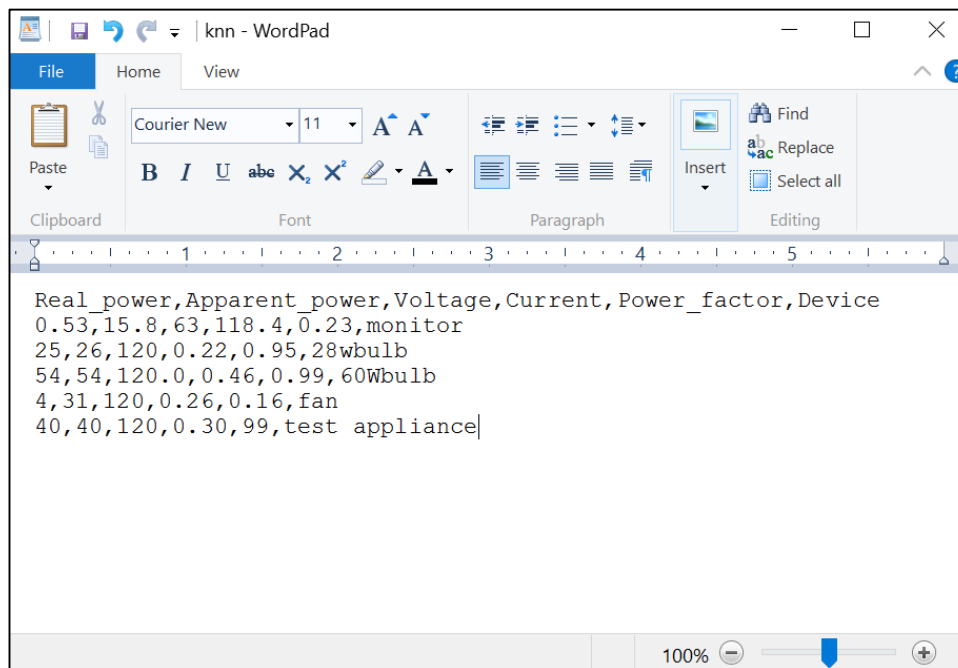


FIGURE 5.11: Modified knn.txt after addition of new appliance name

## CHAPTER 6: COMMUNICATION AND CONTROL DESIGN

The Smart Socket ecosystem has three types of devices: a server, a mobile application and power sockets. Each of these devices have a set of functions that they perform in an orderly manner. A detailed explanation of each device role is given in this section along with the algorithm flowcharts. A brief introduction on socket communication is given below.

### 6.1. Socket Communication

Power sockets communicate with the server using socket communication. A network socket is established to facilitate data transfer between a client and server by running two programs on devices that are on the same network. Client-Server architecture refers to two applications that can communicate and transfer data. Of these two process, one acts as a client and another acts as the sever [38]. A 2-tier architecture is implemented where clients interact directly with the server. This type of architecture is similar to how web browsers interact with web server. The server in Smart Socket system is an iterative server. An iterative server handles both connection requests and data transfers involved [39]. Iterative severes are capable of handling only one client at a time. The other clients wait in queue until the server handles its request. This type of server is best suited for short data transfers as long transfer risk long queue buildup. The Smart Socket devices transfer data in small sizes involving only the load signature. Socket programs can be written in multiple languages such as C, Java, Python, etc. The Smart Socket server uses two client-

server programs of which one is used to keep a network map of all the power sockets and another program executes every minute to transfer data from the power socket to the server.

### 6.1.1. Smart Socket Server

This server acts as a hub for all power sockets in the network and also serves as a link between power socket and mobile application. It contains a database that holds user information, data sent from individual power sockets as explained in chapter 4. In addition to above functions, server continuously checks if any new power sockets are added to the network by using network mapping. It also executes KNN algorithm when user asks to detect connected appliance as explained in chapter 5. Two network socket programs run continuously. One of the code is written in C which performs network mapping and keeps a record of all devices (power sockets) connected. NMAP is a free open source network mapper that is used for network discovery. Network mapping is important from a security point of view as when a user tries to add a new power socket to the network, they can simply enter the socket id and the server will automatically fetch IP address of the socket using network mapping. The raspberry pi on which a server was installed starts to run a network scan immediately by first capturing IP address of itself and saving it with its role as “server” in a text file. Socket program is executed to broadcast role within the network. These two steps are executed using a single shell script. This script is a continuous loop which executes the client-server socket program every 5 seconds.

```
// Shell script to save IP address and execute client-server socket program every 5 seconds.
#IP=$(ip addr show eth0 | awk '/inet / {print $2}' | cut -d/ -f 1)
ip=$(ip addr show eth0 | grep -o 'inet [0-9]\+\.[0-9]\+\.[0-9]\+' | grep -o [0-9].*)
#echo $ip
```



```
nmap -sP -PA21,22,25,3389 $ip.1/24 -oG - | awk '/Up$/{print $2}' > ipAddr.txt
while [ 1 ]
do
./userrole 8000
sleep 5
done
```

This script fetches IP address of the Ethernet port along with system subnet mask. IP address of all connected devices within the network is also obtained and saved in a text file named “ipAddr.txt”. Once IP addresses are saved in “ipAddr.txt”, a second code to obtain all device names/roles is executed in an infinite loop. This code is called “deviceRoles.c” and its structure is as follows:

1. First, the IP address of the device is obtained by calling a function `getIPAddr()`.
2. IP addresses that were saved before in the file `ipAddr.txt` is read and stored into a structure data type `Device.ip` by calling a function `readIPAddress()`.
3. A TCP socket is created and a port number is assigned to it. The system is asked to allow connections to be made using this port.

Once above steps are completed, the system is ready to receive device ID’s from all devices that are connected in the network and indent its role to those devices by calling these functions repeatedly.

4. `getRolesFrmDevices()` is used to get device id’s from other devices and updates a structured data type struct `Device.role` for a respective device IP address. This is similar to the earlier struct `Device.ip`.

5. Functions to accept, send and receive new client connections are also repeated.

A similar code exists on the power socket as well. Power socket saves its IP address locally along with its unique id. By this process, every device in the network knows all other devices by their ids and it creates a foundation for the server to add new power sockets. A flowchart based on the algorithm used is shown below.

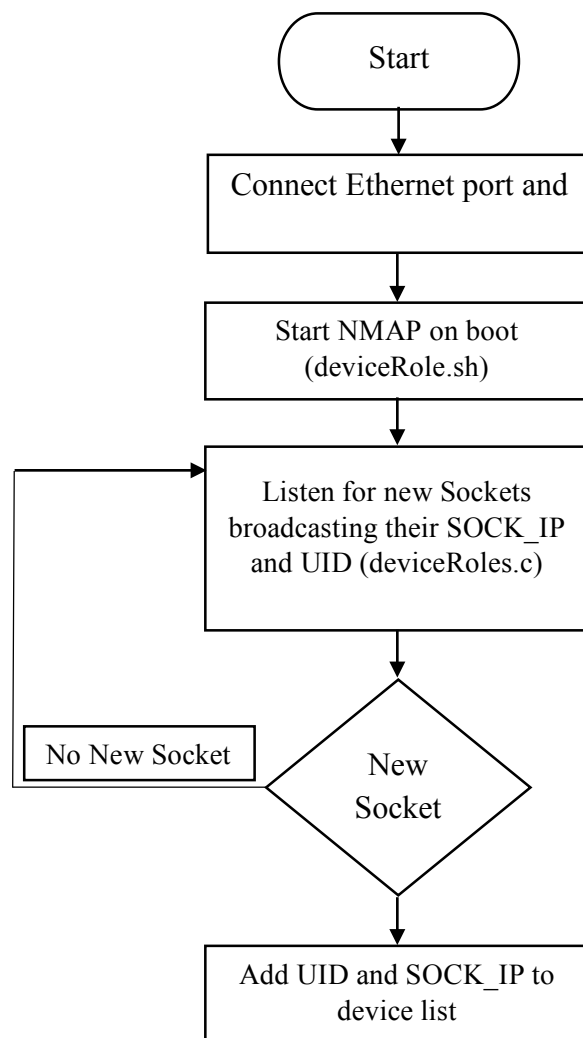


FIGURE 6.1: A flow chart showing smart power socket server process architecture

### 6.1.2. Power Socket

The power socket consists of an Arduino board that measures voltage, current, power consumed, apparent power, real power and power factor of the load. It also has a Raspberry pi that reads above mentioned information from Arduino and saves it in files. This information is sent to the smart socket server every minute. A schedule script runs to check if a user has set a schedule for turn on/off of the socket as mentioned in chapter 5. A flow chart showing these operations is given below.

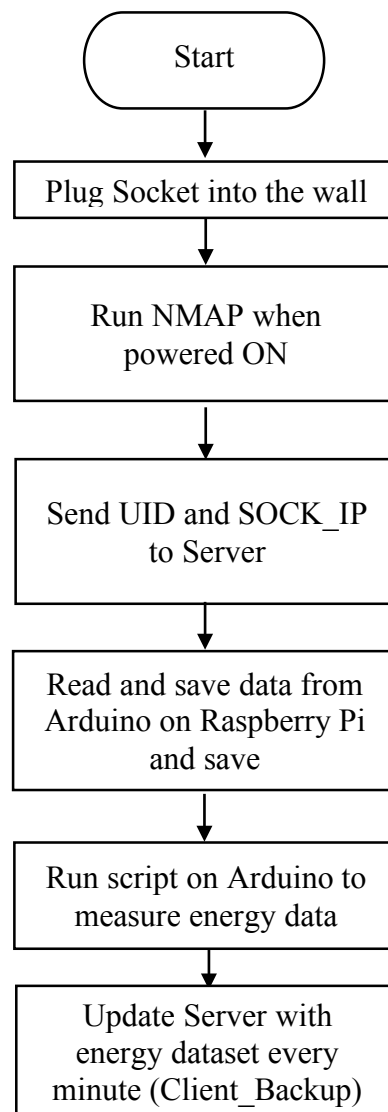


FIGURE 6.2: Power socket flowchart

Power socket's first steps after turn on are similar to that of the server. Device id is sent to the server using a script that is similar to deviceRoles.c. It also saves IP address and roles of other devices locally in a text file. Simultaneously, data from Arduino is read through a python script Arduino\_read.py and saved in a text file. A client-server code written in python executes every minute on the socket to transfer data saved in this file to the server.

Communication between two devices (power socket and server) occurs by means of two network sockets as explained in earlier sections. Python script first creates these sockets by using socket.socket() function which is available in socket module. The general syntax for this function is

```
s = socket.socket(socket_family, socket_type, protocol = 0)
```

Here, socket\_family is a protocol family that is used for transport mechanism. This can be either AF\_INET, PF\_INET, PF\_UNIX, etc. Socket\_type refers to type of communication between two sockets and is typically SOCK\_STREAM. Protocol is generally omitted and defaults to 0. Once a socket is setup, a server or client method is used with above created socket object. User defined functions can also be used to make customized client-server socket programs. Some of the most common methods are s.bind(), s.accept(), s.listen and s.connect(). These methods are used to bind hostname addresses to a socket, accept TCP client connections, listen for TCP connections and initiate server connections respectively. Other commonly used methods can be found in [45].

```
def setupSocket():
```

```
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.connect((host, port))

return s
```

A socket is created in a python script as seen above. `AF_INET` is the protocol family and `SOCK_STREAM` is the type of communication between sockets. `S.connect()` function is used with host (server) address and port number assigned. This function is used to connect to smart socket server. Two functions are written to transfer data and are called from the main loop as seen below:

```
while True:

    backup(location)

    print("Backup Complete!")

    break
```

`backup(location)` is a call to evoke the function that is defined by the same name. This function creates the socket and initiates communication between the client and server.

```
def backup(location):

    s = setupSocket()

    response = sendData(s, location)

    return response
```

`sendData(s, location)` is a function that reads data from the file located at the destination specified by “location” and transfers it to the server in chunks of 1024 bytes. Once entire file is sent to the server, `sendData()` closes the file and socket as seen below.

```
def sendData(s, location):

    print(location)
```

```
dat = open(location, 'rb')  
  
transfer= dat.read(1024)  
  
s.send(str.encode("STORE " + location))  
  
t = time()  
  
while transfer:  
  
    s.send(transfer)  
  
    transfer = dat.read(1024)  
  
dat.close()  
  
s.close()  
  
return "Done sending"
```

This client-server code is used to transfer data from the power socket to server and it is executed every minute to keep an updated log on the server. This data is used for the graph activity in Android application and can be used for further analysis.

## 6.2. Android Application Flow

Smart socket android application communicates with both server and power socket to send commands and receive information. Different PHP scripts and text files are accessed from the application as explained in previous sections and chapters. Certain commands which are of high priority but do not require significant processing times such as turn on/off commands are sent directly to the power socket to ensure minimum delay. Commands and processes which require access to stored data and long processing times such as data for graph and device detection process are sent to the server. This was done to limit hardware dependency at the power socket.

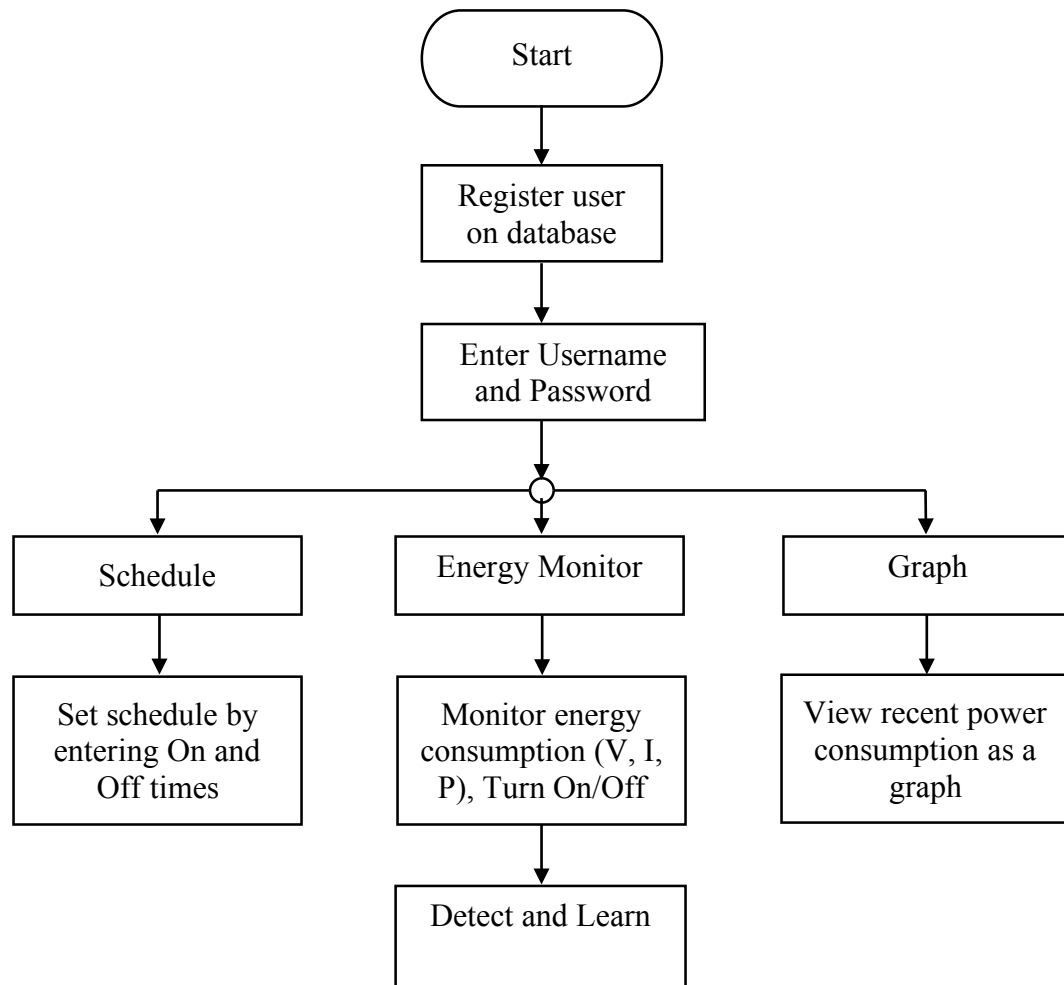


FIGURE 6.3: Flowchart showing a sequence of functionalities in the smart socket android application

Smart socket android application consists of 6 main activity screens which are login, energy monitor, appliance scheduling, Graph, appliance detection and appliance learning. Each of these activities has been explained in detail in Chapter 5. A sequential order of these activities can be seen as a flowchart Figure 6.3. A user is first shown a login screen and also an option to register for new users. After selecting the socket, three options are shown which are energy monitoring, scheduling and graph view. Function of each button and its background process calls are explained in the flow diagram.

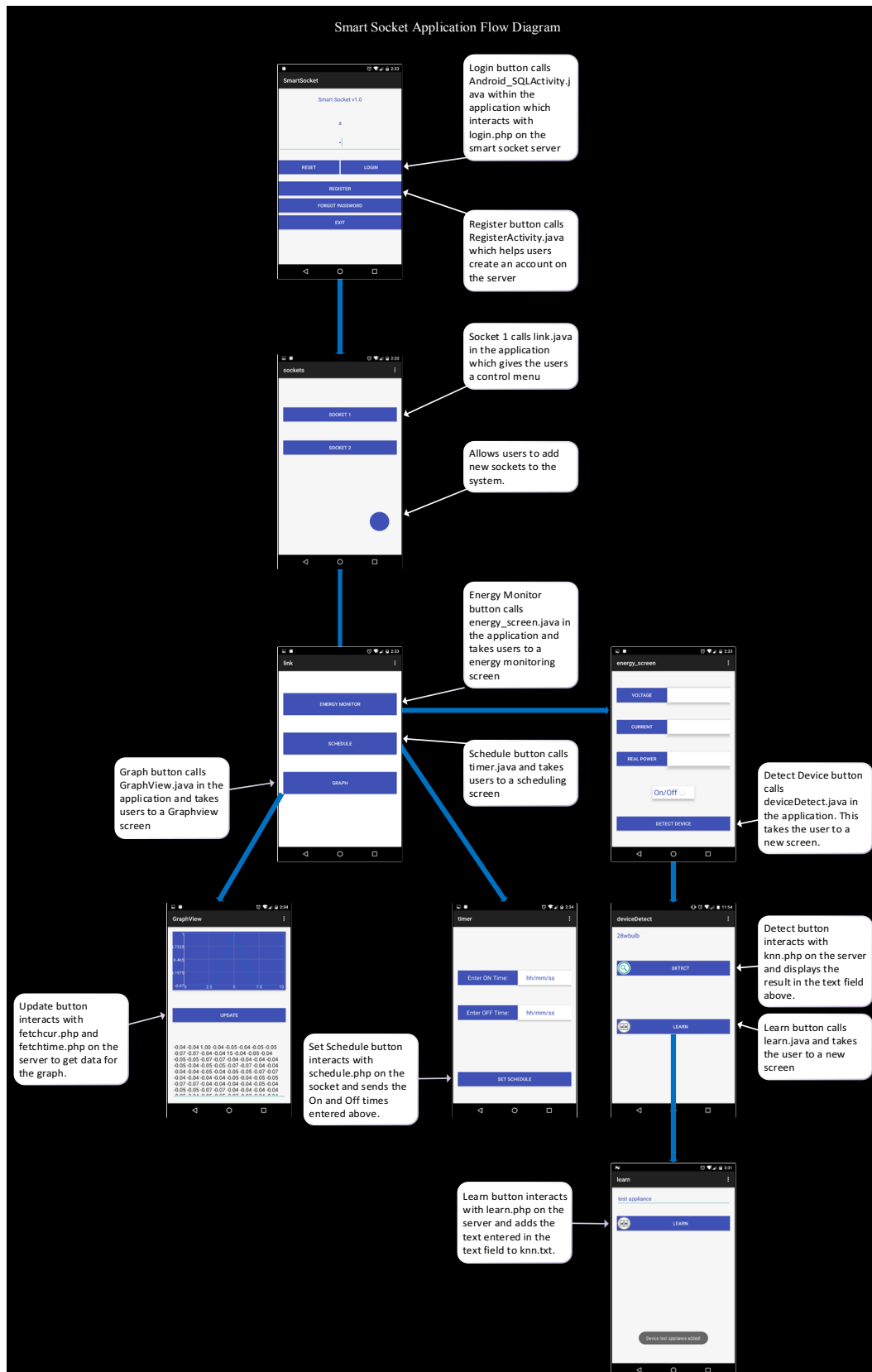


FIGURE 6.4: Smart socket application flow diagram



## CHAPTER 7: LABORATORY SETUP

### 7.1. Smart Socket System Design

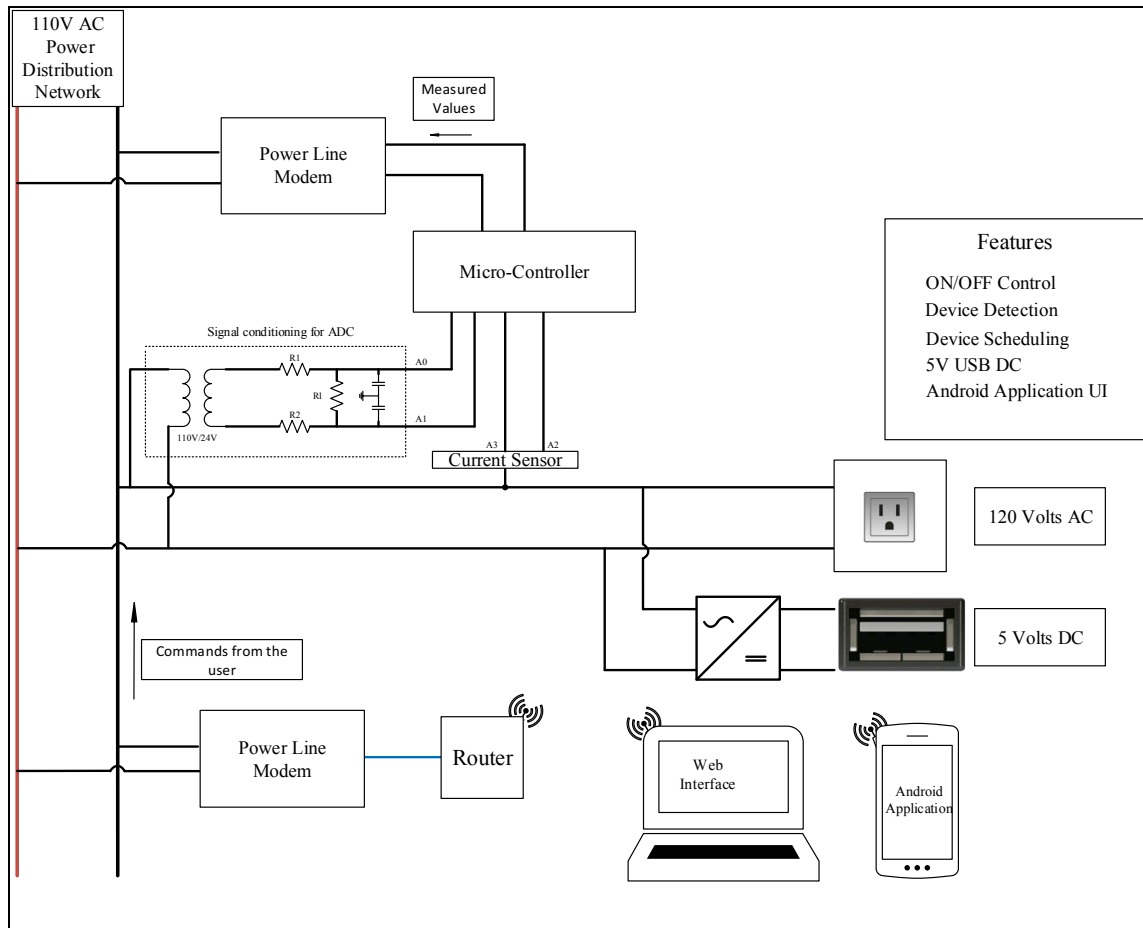


FIGURE 7.1: One line diagram of the entire smart socket system

Figure 7.1 represents the entire smart socket system. This system comprises of a power socket that can switch On/Off, measure power consumed by connected devices, scheduled when to turn on and turn off from an android application. This system contains a server that stores all measured values obtained from the power socket and uses k-NN

algorithm to detect which device is connected to the power socket. The power socket also consists of a 5V DC USB output which allows users to charge their handheld devices directly without the need for an adapter.

## 7.2. Proof of Concept Setup

The first setup contained a Raspberry pi with an ADS1115 ADC interfaced to it to sense analog inputs (V, I). A voltage divider is built to step down the voltage from 120V to under 4V to get input signal under the range of ADC. A Current transformer is used to provide a voltage proportional to current drawn by connected load within input range of ADC as explained in Chapter 3. Figure 7.2 shows the laboratory setup.

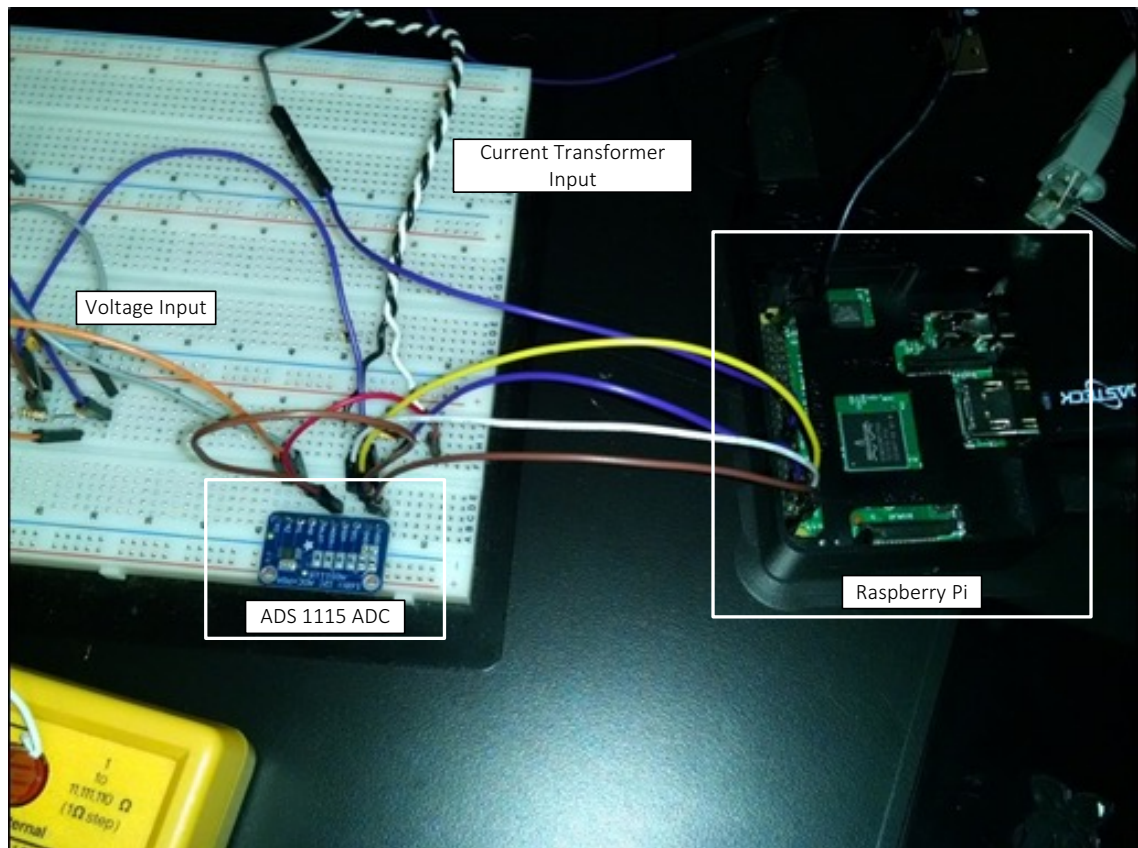


FIGURE 7.2: Proof of concept setup with Raspberry Pi and ADS1115 ADC

A python code “power1.py” is run on the raspberry pi to read analog inputs given to ADS1115. This code calculates RMS voltage and current values after sampling and prints it on a terminal screen as seen in Figure 7.3. Modifications were later made to this code to calculate power consumes as well.

```

%voltage)
0.0373 119.2061
0.0377 119.2061
0.0369 119.0281
0.1617 0.0000
0.2388 119.1298
0.2358 118.9773
0.2407 118.9773
0.2381 118.9010
0.2388 118.9265
0.2374 119.0281
0.2349 118.9265
0.2371 118.9519
0.2372 118.8248
0.2380 118.7485
0.2380 0.0000
0.2383 118.8502
0.2391 118.7739
0.2382 119.0027
0.2379 118.9519
0.2359 118.8248
0.2386 119.0027
0.0573 119.0027
^Cpi@raspberrypi /var/www/html $ sudo python power1.py
pi@raspberrypi /var/www/html $

```

FIGURE 7.3: Voltage and current readings from ADS1115

### 7.3. Alpha Prototype

As explained in section 3.2, ADS1115 was replaced with Arduino microcontroller due to available library support and reduce processes from raspberry pi. Current transformer was replaced and LF 205-S was used to provide AC current wave for sampling. Voltage, current, real power, apparent power and power factor values are calculated and sent to raspberry pi via serial interface as seen in Figure 7.4. Voltage divider used in previous setup is replaced by a 120V/5V transformer to minimize losses caused due to resistance.

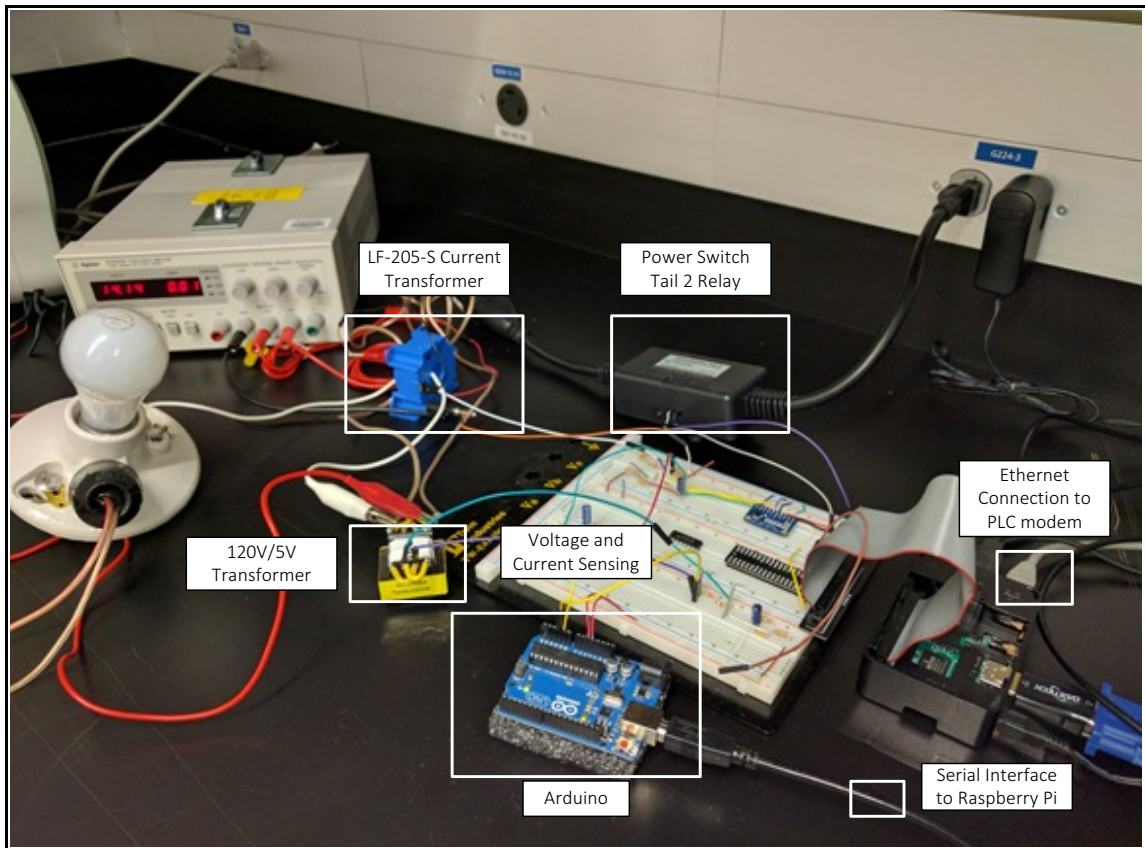


FIGURE 7.4: Alpha prototype setup using Arduino and Raspberry Pi

A server was added to this setup and android application user interface was developed during this phase. Power line network modem (not seen in picture) provided by Lea communications was used to setup a PLC network between server and power socket to transfer data. A 5V DC USB power supply was added to this design as a charging point for small handheld devices.

#### 7.4. Final Thesis Design

Modifications to certain components in alpha prototype were made for the final design. An AC phase angle control dimmer circuit was incorporated to allow dimming of resistive and inductive loads up to 1000 Watts and 100 watts respectively. LF 205-S current sensor was replaced with a compact PCB design ACS712 current sensor. ACS712 current sensor operates from 5V and outputs analog voltage proportional to current measured on the sensing terminals [67].

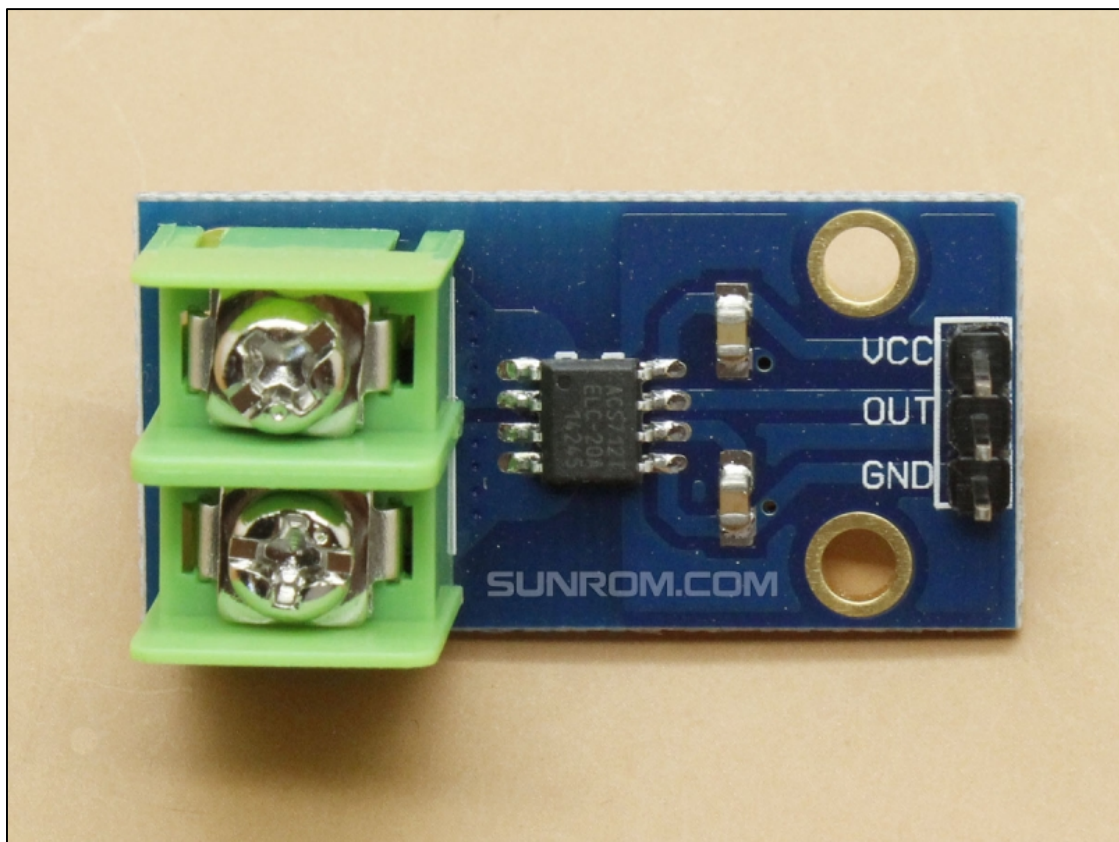


FIGURE 7.5: Top view of ACS712 current sensor designed by sunrom technologies [67]



FIGURE 7.6: A picture of the electric project box used in the final design

In order to minimize overall footprint of components, they were later placed into an 8 inch x 6 inch electric project box (FIGURE 7.6 and FIGURE 7.7). Voltage sensing circuit and 5V USB output was mounted on a general purpose circuit board (PCB) and placed in the box. Cutouts for 120V AC outlet and 5V DC female USB port were drilled into the lid using a bench drill machine. Top view of the open box with all components can be seen in FIGURE 7.7 and closed view is seen in FIGURE 7.8.

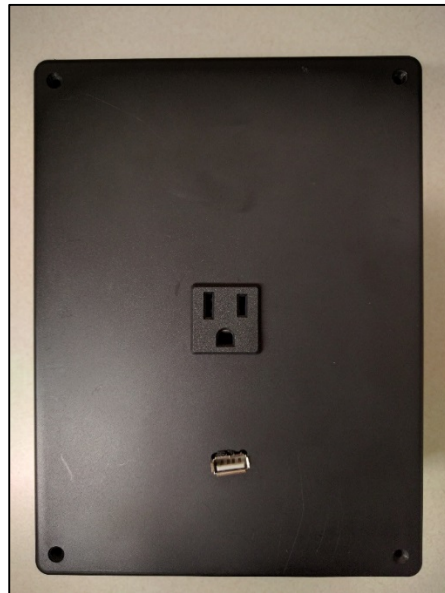


FIGURE 7.8: Top view of smart power socket

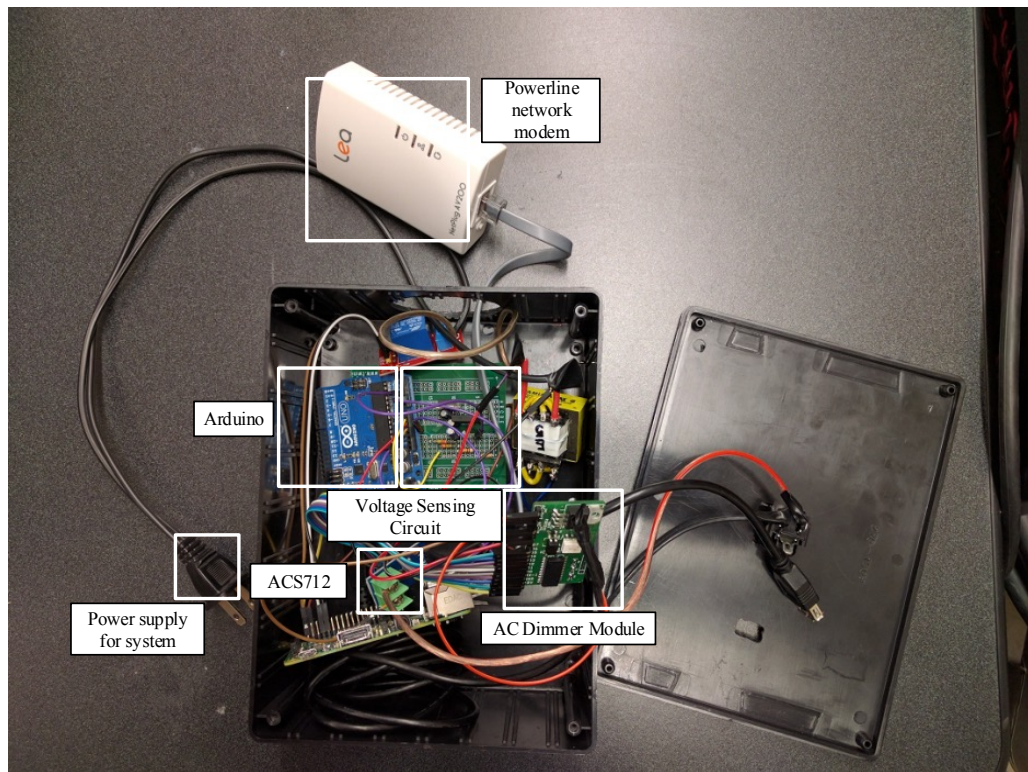


FIGURE 7.9: Smart power socket components inside an electrical box

## CHAPTER 8: CONCLUSIONS AND FUTURE SCOPE

A brief overview on communication technologies and protocols was discussed along with an extensive market survey on commercially available home automation and smart plugs. A smart power socket that utilizes existing home power lines for communication was presented in this thesis. Every aspect of the design process starting from microcontroller selection to final hardware setup was discussed in detail with source code explanation.

A power socket ecosystem with three main components was designed. An Android mobile application as the interface along with a server which performs regression algorithm calculations and stores data received from the power socket. Smart power socket designed as part of this thesis uses readily available rapid prototyping controllers. This choice was made to keep cost low and allow easy migration to other cheaper controllers with lesser functionalities. A conscious effort was made to minimize the total cost. Arduino and Raspberry Pi used in this design can be replaced by cheaper microcontrollers or chips (SoC). Future models of this power socket can implement a Qualcomm Atheros QAC7000 single chip GreenPhy solution instead of power line network modem used here (LEA networks). Overall footprint of smart power socket can be significantly reduced and it can be flush with the wall without problems faced by similar Wi-Fi enabled devices that have connectivity issues between walls.



Further research can be conducted in improving k-NN algorithm used in this thesis to detect appliances which have varying loads and multiple operating modes. Training data set can be increased with large number of household devices to improve accuracy of detection scripts. Additional changes can be made in the android application to include a process to add new power sockets. IP address of these new power sockets can be obtained from the network mapping database which is already present. Smart power socket ecosystem can also be modified to facilitate file transfer between two computers connected to different power sockets using the same power lines as a medium to transfer data.

## REFERENCES

- [1] International Energy Agency, “World Energy Statistics 2015”, (IEA): [http://www.iaea.org/publications/freepublications/publication/KeyWorld\\_Statistics\\_2015.pdf](http://www.iaea.org/publications/freepublications/publication/KeyWorld_Statistics_2015.pdf)
- [2] NEMA, “Smart Grid consumer collaborative, Truth about Smart Meters.” <https://www.nema.org/Technical/Documents/SGCC-Myths-vs.-acts-Fact-Sheet.pdf>
- [3] Smart home USA, “How X10 works”. <http://www.smarthomeusa.com/how-x10-works/>
- [4] Poul Ejnar Røvsing, Peter Gorm Larsen, Thomas Skjødeberg Toftegaard and Daniel Lux; “A Reality Check on Home Automation Technologies”
- [5] Smart home USA, “Universal Powerline Bus”: <http://www.smarthomeusa.com/universal-powerline-bus/>
- [6] Md. Zahurul Huq, Prof. Syed Islam, “HOME AREA NETWORK TECHNOLOGY ASSESSMENT FOR DEMAND RESPONSE IN SMART GRID ENVIRONMENT”
- [7] Carles Gomez and Josep Paradells, “Wireless Home Automation Networks: A Survey of Architectures and Technologies”, CONSUMER COMMUNICATIONS AND NETWORKING
- [8] About Z-Wave Technology, March 2013. [http://z-wavealliance.org/about\\_z-wave\\_technology/](http://z-wavealliance.org/about_z-wave_technology/)
- [9] Galeev, Mikhail T. "Catching the Z-Wave": <http://www.embedded.com/design/connectivity/4025721/Catching-the-Z-Wave>
- [10] “How TED Works”; <http://www.theenergydetective.com/tedworks>
- [11] “TED Pro Home”; <http://www.theenergydetective.com/tedprohome.html>
- [12] P3 International, Kill A Watt, P4400, <http://www.p3international.com/products/P4400.html>

- [13] Belkin WEMO home automation products, <http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/#whyWemo>
- [14] Samsung Smart Things, “How It Works”; <https://www.smarthings.com/how-it-works>
- [15] George W. Hart, Nonintrusive Appliance Load Monitoring. *Proceedings of the IEEE*, 80(12):1870-1891, Dec 1992.
- [16] H. Najmeddine et al., State of art on load monitoring methods. *Proceedings of 2nd IEEE International Conference on Power and Energy*, December 1-3, 2008, Johor Baharu, Malaysia, pp. 1256-1258.
- [17] A.J. Bijker, X. Xia and J. Zhang, Active power residential non-intrusive appliance load monitoring system. *Proceedings of IEEE AFRICON 2009*, September 23-25, 2009, Nairobi, Kenya, pp. 1-6.
- [18] G.W. Hart, Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, vol. 8, no. 2, June 1989, pp. 12-16.
- [19] Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, Sutharshan Rajasegarar, “Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey” *Sensors Basel* 2012 December; 12(12): 16838–16866. Published online 2012 December 6. doi: 10.3390/s121216838
- [20] Elena Mainardi and Marcello Bonfe (2008). *Powerline Communication in Home-Building Automation Systems, Robotics and Automation in Construction*, Carlos Balaguer and Mohamed Abderrahim (Ed.), ISBN: 978-953-7619-13-8, InTech, Available from:

- [http://www.intechopen.com/books/robotics\\_and\\_automation\\_in\\_construction/powerline\\_communication\\_in\\_home-building\\_automation\\_systems](http://www.intechopen.com/books/robotics_and_automation_in_construction/powerline_communication_in_home-building_automation_systems)
- [21] Home Plug, “HomePlug™ AV2 Technology”, [http://www.homeplug.org/media/filer\\_public/2c/32/2c327fc8-25bb-409e-abf7398534c24dc/homeplug\\_av2\\_whitepaper\\_130909.pdf](http://www.homeplug.org/media/filer_public/2c/32/2c327fc8-25bb-409e-abf7398534c24dc/homeplug_av2_whitepaper_130909.pdf)
- [22] GreenPHY by HomePlug, [http://groups.homeplug.org/tech/whitepapers/HomePlug\\_GreenPHY\\_Overview.pdf](http://groups.homeplug.org/tech/whitepapers/HomePlug_GreenPHY_Overview.pdf)
- [23] Home Plug Alliance, “HomePlug Green PHY™: perfect fit for Smart Energy / Internet of Things (IoT) applications”, <http://www.homeplug.org/tech-resources/green-phy-iot/>
- [24] Arduino Uno Board, “Overview”, <http://www.arduino.cc/en/Main/ArduinoBoardUno>
- [25] Arduino reference, setup(), <https://www.arduino.cc/en/Reference/Setup>
- [26] Raspberry Pi 2 Model B+, <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>
- [27] Raspberry Pi Help, “Quick Start Guide”, <https://www.raspberrypi.org/help/quick-start-guide/>
- [28] Raspberry Pi Documentation “GPIO: Raspberry Pi models A and B” <https://www.raspberrypi.org/documentation/usage/gpio/>
- [29] Raspberry Pi VNC Server, [http://elinux.org/RPi\\_VNC\\_Server](http://elinux.org/RPi_VNC_Server), shared under Creative Commons Attribution-ShareAlike 3.0 Unported license.
- [30] Texas Instruments, “ADS1015 Datasheet”, <https://www.adafruit.com/datasheets/ads1015.pdf>
- [31] Texas Instruments, “ADS1115 Datasheet”, <https://www.adafruit.com/datasheets/ads1115.pdf>

- [32] Pete Semig and Collin Wells, Texas Instruments, “A Current Sensing Tutorial--Part 1: Fundamentals”
- [33] Google shows off new version of Android, announces 1 billion active monthly users. Techspot. Retrieved June 30, 2014
- [34] "Android KitKat", Android Developers Portal, <https://developer.android.com/about/versions/kitkat.html>, Retrieved November 16, 2013.
- [35] IDC analyze the future, Smartphone OS Market Share, 2015 Q2, <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
- [36] Android Developer, Creating an Android Project: <http://developer.android.com/training/basics/firstapp/creating-project.html>
- [37] Graph View-open source graph plotting library for Android, <http://www.android-graphview.org/>
- [38] Unix socket- Client Server model, [http://www.tutorialspoint.com/unix\\_sockets/client\\_server\\_model.htm](http://www.tutorialspoint.com/unix_sockets/client_server_model.htm)
- [39] Concurrent and iterative servers, IBM knowledge center, [https://www.ibm.com/support/knowledgecenter/SSLTBW\\_2.1.0/com.ibm.zos.v2r1.hali001/concurrentanditerativeservers.htm](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.hali001/concurrentanditerativeservers.htm)
- [40] Intel Galileo; <http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-overview.html>
- [41] M. Moradian and A. Baraani, k-Nearest-neighbor-based-associationalgorithm. Journal of Theoretical and Applied Information Technology, vol. 6, Dec. 2009, pp 123-129.
- [42] Stanford University Department of Statistics, Correspondence analysis and related methods. Available: <http://www.econ.upf.edu/~michael/stanford/maeb4.pdf>

- [43] SciPy ecosystem, <https://www.scipy.org/>
- [44] PANDAS overview, <http://pandas.pydata.org/>
- [45] Python Network Programming, Tutorials Point, [http://www.tutorialspoint.com/python/python\\_networking.htm](http://www.tutorialspoint.com/python/python_networking.htm)
- [46] Dimmer module- digital control- 256 steps, <http://www.sunrom.com/p/dimmer-module-digital-control-256-steps>
- [47] Dimmer theory, PCB heaven [http://www.pcbheaven.com/wikipages/Dimmer\\_Theory/](http://www.pcbheaven.com/wikipages/Dimmer_Theory/)
- [48] Power Line Communications: State of the Art and Future Trends, Niovi Pavlidou, Aristotle University of Thessaloniki, A. 1. Han Vinck, University of Essen, Javad Yazdani and Bahram Honaty, University of Lancaster
- [49] Power Line Communications: State of the Art and Future Trends, Niovi Pavlidou, Aristotle University of Thessaloniki, A. 1. Han Vinck, University of Essen, Javad Yazdani and Bahram Honaty, University of Lancaster
- [50] A Study on Power Line Communication Abdul Mannan, D.K.Saxena, Mahroosh Bandy, International Journal of Scientific and Research Publications, Volume 4, Issue 7, July 2014
- [51] Home automation info, the home automation reference site, [http://www.homeautomationinfo.com/Drupal/technology\\_upb](http://www.homeautomationinfo.com/Drupal/technology_upb)
- [52] "ATIS Telecom Glossary 2007". atis.org., Retrieved 2008-03-16 <http://www.atis.org/glossary/definition.aspx?id=231>
- [53] Smart homes, ZigBee Alliance, <http://www.zigbee.org/what-is-zigbee/494-2/>
- [54] ZigBee Coordinators at a glance, retrieved from [https://www.controlanything.com/Relay/Relay/ZB\\_ZIGBEE\\_COORDINATORS](https://www.controlanything.com/Relay/Relay/ZB_ZIGBEE_COORDINATORS)

- [55] 6LoWPAN working group, <https://datatracker.ietf.org/wg/6lowpan/charter/>
- [56] Kill-A-Watt EZ, P4460, <http://www.p3international.com/products/p4460.html>
- [57] Belkin WEMO home automation, <http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/>
- [58] Ruzzelli, A.G.; Nicolas, C.; Schoofs, A.; O'Hare, G.M.P. Real-Time Recognition and Profiling of Appliances through a Single Electricity Sensor. In Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, Boston, MA, USA, 21–25 June 2010; pp. 1–9.
- [59] Manyazewal Tesfaye Fitta, Load Classification and Appliance Fingerprinting for Residential Load Monitoring System, Espoo 13.08.2010
- [60] Plogg smart plug, <http://www.plogg.co.uk/>
- [61] Introduction to HTML, W3schools, [http://www.w3schools.com/hTml/html\\_intro.asp](http://www.w3schools.com/hTml/html_intro.asp)
- [62] PHP5 tutorial, W3schools, <http://www.w3schools.com/php/default.asp>
- [63] Setting up an apache web server on raspberry pi, <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>
- [64] OpenEnergyMonitor, about us, <https://openenergymonitor.org/emon/sustainable-energy>
- [65] Electricity monitoring library, GitHub, <https://github.com/openenergymonitor/EmonLib>
- [66] Connect Raspberry pi and Arduino with serial USB cable, <https://oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/>
- [67] Sunrom technologies current sensor, 20A- ACS712, <http://www.sunrom.com/p/current-sensor-20a-ac712>