Automated Energy Meter using Wi-Fi enabled Raspberry-Pi

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

By

Pendyala Arun Chandra – 12BEC0300

Golla Mohith Vamsi – 12BEC0144

Yamparala Sri Manoj – 12BEC0056

Under the guidance of

Prof. Gerardine Immaculate Mary.

School of Electronics Engineering,

VIT University, Vellore.

VIT® UNIVERSITY (Estd. u/s 3 of UGC Act 1956) VELLORE ▪ CHENNAI www.vit.ac.in

May, 2016

DECLARATION

We hereby declare that the thesis entitled "Automated Energy Meter using Wi-Fi enabled Raspberry-Pi" submitted by us, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT University, is a record of bonafide work carried out by us under the supervision of Prof. Gerardine Immaculate Mary.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidates

CERTIFICATE

This is to certify that the thesis entitled "Automated Energy Meter using Wi-Fi enabled Raspberry-Pi" submitted by **Arun Chandra P, Mohith Vamsi G, Sri Manoj Y**, School of Electronics Engineering, VIT University, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by them under my supervision, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

Program Chair [B.Tech ECE]
School of Electronics Engineering

ACKNOWLEDGEMENT

It gives us immense pleasure to offer our sincere thanks to our project guide and mentor Prof. Gerardine Immaculate Mary. It would have not been possible to accomplish the milestones of our project without her invaluable academic support, meticulous scrutiny, technical supervision and timely advice.

We are also grateful to our review panel members Dr. Alex Noel Joseph Raj, Prof. Arun Kumar K.R. and Prof. Venugopal P whose vital suggestions helped us in elevating the project to a different level. It was their prompt inspirations, timely suggestions and dynamism that enabled us to complete our thesis.

We also thank Dr. Elizabeth Rufus, Dean of School of Electronics Engineering and Prof. Renuga Devi S., Head of Department of Department of communication Engineering , SENSE, VIT University for giving this opportunity to apply the theoretical knowledge we have gained during our past four years. We also place on record, our sincere thanks to the Management of VIT for their support and provision of the necessary resources and facilities for completing the project in timely manner.

We take this opportunity to express our deep sense of gratitude to all those who helped us in completion of our project, directly or indirectly. In this regard, we heartily thank our parents for their relentless support and encouragement and lab assistant of Embedded Systems Lab
for being cooperative.

Pendyala Arun Chandra

Golla Mohith Vamsi

Yamparala Sri Manoj

EXECUTIVE SUMMARY

In a large country like India collecting Energy Meter readings from every individual house can be a very difficult task involving large amount of work hours. We aim to reduce manual intervention in this process with the help of our low cost system developed around a Raspberry Pi. The system developed by us also removes the man made errors in the billing process. The back bone of our project is a Raspberry Pi model 2 with 1GB of RAM and 900 MHZ processor. A Light Detecting Resistor Module(LDR) is used to count the number of pulses on the electric meter. In the case of the meter used 32000 pulses equal to 1 KWH. LDR module outputs a ‘0’ whenever it detects light and ‘1’ when a pulse is not detected. We wrote a program to count the number of pulses in python running on the Raspberry Pi. A scheduler has been used to display the number of pulses and the respective energy usage during a specified period of time. In order to export the data to spreadsheets we made use of Google Developer Console and Google’s API. A basic webpage was developed to display the contents of the spreadsheet. Keeping in mind the case where some users cannot have access to internet we have used a GSM module to send the user’s energy consumption data via SMS.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
GPIO	General Purpose Input Output
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
JSON	Java Script Object Notation
JWT	JSON Web Tokens
LDR	Light Dependent Resistor
LED	Light Emitting Diode
RPi	Raspberry Pi
SMS	Short Message Service

1. INTRODUCTION

1.1. Objective

India is presently witnessing substantial rise in investments especially industrial and infrastructural areas. This growth is further being augmented by the increase in power generation capacity and reliable power distribution and measurement network. In some developed countries like the United States smart meters are already deployed. Although some companies have introduced smart meters, many households have conventional meters. This provides a chance for companies as well as engineers to develop innovative metering solutions

The process of collecting the data from the energy meters with the help of sensors and storing them at a central server is called Automatic Meter Reading (AMR). This technique can be immensely useful in hugely populated countries like India. In India employees are hired to go from house to house and collect the reading from the energy meters. India uses a slab based pricing model to charge the customers for the usage of electricity. After a particular slab, the price of power per kWh increases sharply. This method is used in order to reduce the burden on poor people. But any error or purposeful mistake in the reading can cause huge loss to the power companies. Automatic Energy Meter can provide a cheap and effective solution to this problem. This also prevents the manual labor being used for billing.

The electricity meters market in India is estimated to grow at a Compound Annual Growth Rate(CAGR) of 10-12% between FY 2010-11 and 2017-18. This indicates that introduction of smart meters is a continuous process. Automatic Energy Meters can also be used as a means to increase the penetration of Internet of Things (IOT) in a developing country like India. The RPi used in our model can be used for future home automation solutions like water usage meter, switching off power to individual appliances and security monitoring.

1.2. Motivation

In the present day scenario where every process is being automated, the process of acquiring of energy consumption from the meter fixed at the user’s place is still being implemented by sending manual labor. By making this process automated we can reduce the costs spent on manual labor for the service provider. In a slab based billing system for a country like India where cost for each unit used changes with the change in the number of units of energy consumed. So, when a person’s energy consumption for a month just crosses a slab by a very low margin, his bill will increase highly. Providing the user to access the energy consumption of his place remotely whenever he wants, would allow him to reduce his bills. These acted as pivotal reasons which motivated us to take up this project.

1.3. Background

The proposed system can be incorporated to the installed energy meters without much effort. It consists of a light sensor module which detects the pulses given out by the meter and the python program developed computes the energy consumed by the load and sends these values monthly to the Google spreadsheets via Drive API. The users can access the information on a webpage as well as an Android App. In addition to this, they also receive an SMS with the help of a GSM module interfaced to the RPi.

1.3.1. Benefits of the product

The basic idea of this project is to get remote access of the energy meter in our homes by using the basic concept of Internet of Things (IOT). Getting remote access of the meter helps in automation of things and hence reduces the need for physical effort which reduces the need of labor, so a lot of cost for labor can be saved for the government.

Another benefit of the proposed idea is that the errors which occur due to manual readings can be omitted. The inaccuracy in the billing can be prevented especially in a country like India which uses a slab based pricing model to charge the customers for the usage of electricity. After a particular slab, the price of power per kWh increases sharply. This method is used in order to reduce the burden on poor people as the readings are accurate.

The major benefit of this proposed idea is giving the customer remote access of the energy meter fixed outside his house.

1.3.2. Proposed Idea

Our proposed model uses RPi counts the number of pulses detected by Light sensor module. The sensor is placed on the energy meter. The RPi counts the number of pulses to calculate the energy consumption in kWh according to the conversion mentioned on the energy meter. This data is then sent to Google Spreadsheets using Google API. The spreadsheet can be accessed on a website and android app that were designed. A GSM module is interfaced with the RPi to send energy usage data to the respective customer via SMS.

1.3.3. Our Approach

This project uses RPi 2 model B board. It has features similar to the RPi 1 model B+ such as 4 USB ports, 40 GPIO pins, micro SD card slot etc. In addition to these above mentioned features, it has 1GB RAM, 900 MHZ quad-core ARM Cortex-A7 CPU. The RPi which was used runs on the operating system, Raspbian Wheezy OS which is a Linux distribution.

RPi based automated energy meter was implemented using Zigbee Protocol. Our project simplifies the design by using WiFi protocol. The ZigBee technology is designed to carry small amounts of data over a short distance while consuming very little power. On the other hand, Wi-Fi is a local area network (LAN) that provides internet access within a limited range. In our model, Wi-Fi seems to be a better option as the data uploaded to the internet is accessible everywhere if wireless networks are present.

1.3.4. Literature Survey

- C. D. Bonganay, J. C. Magno, A. G. Marcellana, J. M. E. Morante and N. G. Perez, "Automated electric meter reading and monitoring system using ZigBee-integrated RPi single board computer via Modbus," Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on, Bhopal, 2014

This paper proposes the automation of energy meter using a zigbee network. The basic idea of automating the energy meter has been taken from this paper. We have chosen the Wi-Fi network instead of zigbee due to the trade-off as explained in 4.6.

- S. Shahidi, M. A. Gaffar and K. M. Salim, "Design and implementation of digital energy meter with data sending capability using GSM network," Advances in Electrical Engineering (ICAEE), 2013 International Conference on, Dhaka, 2013

  This paper proposes sending the acquired data from the sensor to the customer's mobile phone by an SMS. We have incorporated this idea in our project to send the acquired data from Light Sensor Module to the customer by connecting a GSM module to the RPi.
- T. Ahmed et al, "Automatic electric meter reading system: A cost-feasible alternative approach in meter reading for Bangladesh perspective using low-cost digital wattmeter and WiMax technology," M.S. thesis, Dept. Comp. Science, American International University, Dhaka, Bangladesh, 2010.

  This paper proposes an Automated Meter Reading System (AMR) for the common energy meters which are used in the country of Bangladesh. The energy meter which was used in this paper was an analog meter which shows the reading by rotating a disk which is different from our proposed energy meter. We have taken the idea of taking the reading data from energy meter and automating the reading system process.

- L. Quan-Xi and L. Gang, "Design of remote automatic meter reading system based on ZigBee and GPRS" in Third International Symposium on Computer Science and Computational Technology, P.R China, 2010, pp. 186-189.

  This paper proposes the idea of automating the reading system using ZigBee protocol and a GSM network. Our proposed idea works using Wi-Fi network. As an additional feature we have also added the feature of sending the acquired data using GSM network.

2. PROJECT DESCRIPTION AND GOALS

2.1. Block Diagram

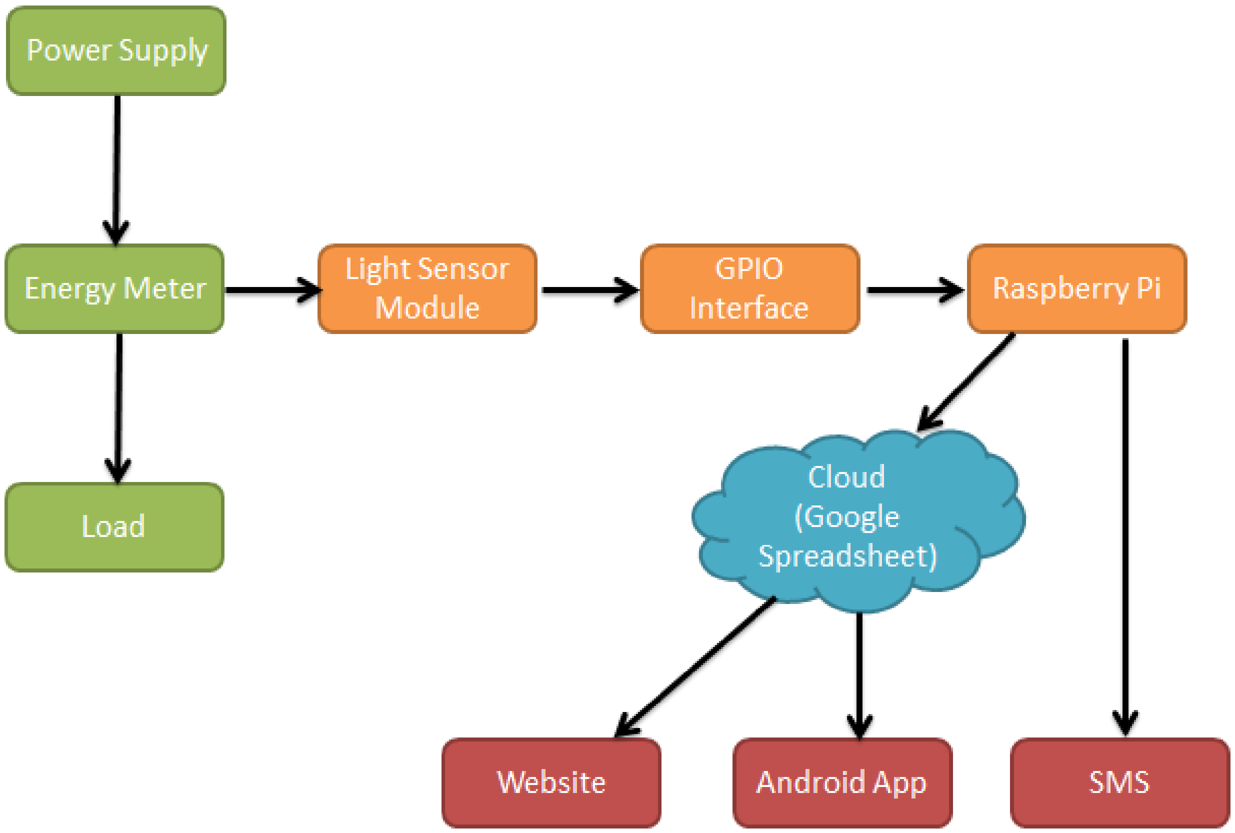


Figure 2.1: Basic Block Diagram

Power Supply: This is the basic power supply which is supplied to the user. In India the power supply is given at 220V AC. The same is taken as input power for our project.

Energy Meter: The power supply from the service provider is connected to the energy meter. We have used a digital meter which is commonly used in India. It consists of an LED on it and our job is to count the number of times the LED has blinked from which the energy consumption can be interpreted.

Load: The energy meter is connected to the load. In general, the load refers to all the electronic appliances. The main purpose of the project is to automate the reading of the energy consumption of these appliances.

Light Sensor Module: This is the sensor used to interface the reading of the energy meter to the microcontroller. It is just a diode which sends binary data to the microcontroller whenever the LED on the energy meter blinks.

GPIO Interface: It's the interface on which all the programming has been done. The operation system which had been used was Raspbian OS. All the programming was done using Python Language

RPi: This is the microcontroller used to automate the reading system. Model 2 B has been used. The microcontroller should stay switched on all the time as it needs to receive the acquired data from Light Sensor Module and send the data at regular intervals to the user.

Cloud: It's the Google Spreadsheet to which the data which is acquired is sent using the Wi-Fi module which is connected to the USB port of the RPi.

Website: It's a HTML site which has been designed so that user can find the energy consumption of his space remotely and calculate his payable bills. The data which is hosted in this site is taken from the Google Spreadsheet.

Android Application: An android app has been designed so that the data is on website can be seen even on app which helps the user to directly open the app when he wants to access the energy consumption data on his mobile smartphone.

SMS: A message will be sent periodically to the user using the GSM network to the user‘s mobile phone. This is done by connecting a GSM module to the RPi.

2.2. Goals

By the end of project we expect to do the complete automation of the energy meter using the proposed RPi. We expect to provide the customer three ways to know the energy consumption. The first way is from a website designed for the customer, so he could login with his credentials and find the details of the power consumption. The second way is by providing an android app where the user can use the application on his android smartphone and know the details just by opening the application. The final way is using an SMS, here the user receives the message directly to his registered mobile number monthly. We expect to complete these three tasks by the end of the project.

3. TECHNICAL SPECIFICATIONS

3.1. Hardware Specifications

Hardware components like Raspberry Pi, Wi-Fi module, GSM module, energy meter, LDR module have been used and their specifications are described below:

3.1.1. Raspberry Pi

Raspberry Pi is an inexpensive, credit card sized computer which has become an enabling technology for IoT (Internet of things).Several models have been released so far but all of them have certain features in common such as Broadcom system on a chip (SOC) which includes an ARM compatible CPU and an on chip graphics processing unit GPU (a VideoCore IV). The speed of CPU range from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM depending on the models. SD cards are used to store the operating system and program memory in either the SDHC (Secure Digital High Capacity) or MicroSDHC sizes. In this project, we have used RPi 2 Model B which belongs to the second generation of RPi which has the following specifications:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

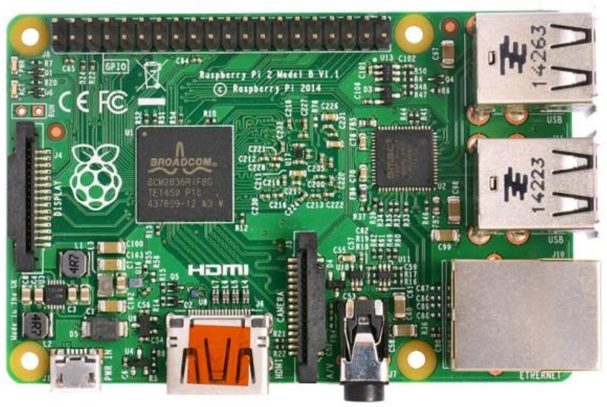


Figure 2.1: RPi 2 Model B

The RPi boards operate at a voltage of 3.3 V such that HIGH signal is indicated by 3.3V and LOW signal is indicated by 0V.

Compared to the earlier models such as RPi model B which has 26 pins and 2 USB ports, this model has 40 pins and 4 USB ports. Also the SD card slot has been replaced by MicroSD card slot which further decreases the area of the Board. Ethernet port is used for enabling wired connectivity with other devices. It can be used to a computer for accessing the RPi and executing the commands on the terminal window.

GPIO interface of RPi

The RPi 2 Model B has 40 general purpose input/output pins (GPIO) .

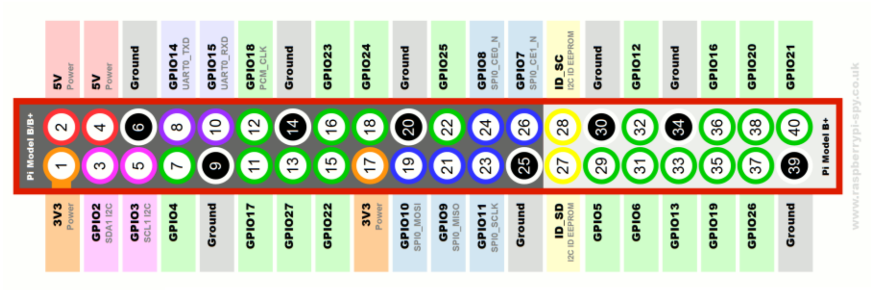


Figure 3.2: Pin layout of RPi 2 model B

The pins include general purpose pins, Pins for supplying power supply (3.3V, 5V) to external modules. The interface also supports SPI (Serial peripheral Interface), I2C (Inter integrated Communication), UART(Universal Asynchronous Receiver/Transmitter) communication.

3.1.2. Wi-Fi module

The RPi Board becomes a Wi-Fi enabled board that communicates with other applications over internet by connecting a Wi-Fi adapter to one of the four USB ports of the board. The RPi can be accessed from a computer through Secure Shell (SSH) over Wi-Fi connection. Commands can be executed on the RPi through the terminal window in the computer.

A GUI is provided for setting up Wi-Fi connections in the Raspbian Wheezy version and the other current releases. A compatible Wi-Fi adapter is required for the purpose of enabling internet connectivity wirelessly and the best available option is the Edimax EW-7811Un which is nano size USB adapter supporting upto 150 Mbps data rate.

Figure 3.3: Wi-Fi Adapter

3.1.3. Light sensor Module

A photo resistor or light dependent resistor has been used in this project. A photo resistor has the property of variable resistance which depends on the increase in incident light intensity. A photo resistor is made of high resistance semiconductor. The resistance can be as high as few mega ohms which decreases to a few hundred ohms upon incidenting light on it.

The module which we have used consists of analog and digital output pins. For our project, we have interfaced the digital output with the GPIO pin of the RPi board. The output signal from our module is LOW in bright light and HIGH in dark. The comparator on the light sensor module triggers the digital output.

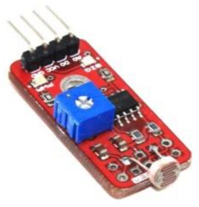


Figure 3.4: Light Sensor Module

3.1.4. Energy meter

Electronic energy meters are used now-a-days instead of the conventional mechanical energy meters with rotating discs, for monitoring the energy consumption by domestic households and industries. In our project, we have used a single phase energy meter which is as shown in the figure 3.5.

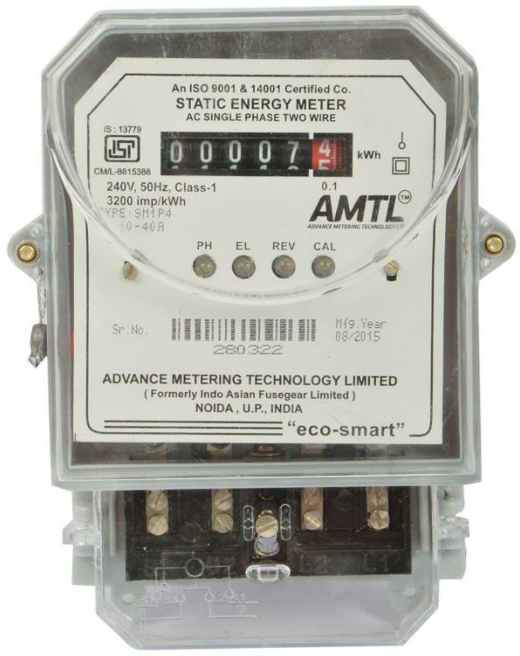


Figure 3.5: Energy meter

The light sensor module is used to detect the pulses given out by the CAL LED of the energy meter. The other LEDs – EL and REV are used to indicate the malfunctioning of the meter. PH is used to indicate the ON or OFF condition of the AC power supply given to the load. Analog and Digital energy meters are currently available in the market. But the counter type analog meters are of lesser cost compared to the digital counterparts. The relation between energy consumed value and number of pulses given out by the CAL LED. In our case, 1 kWh corresponds to 3200 pulses produced. This relation is incorporated in the program logic. Bulbs have been used as load in our project. The connections to the energy meter are as shown in the figure 3.6. IN refers to the connections from the power supply and OUT stands fo the connections to the load. P stands for Positive terminal and N is for neutral or ground.

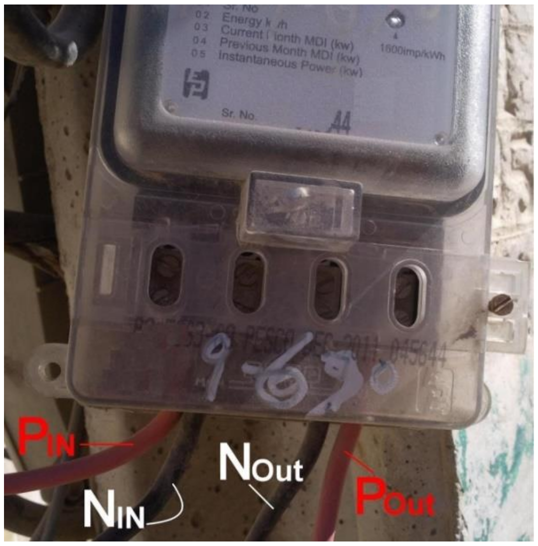


Figure 3.6: Energy meter connections

3.1.5. GSM Module

A GSM modem is a special type of modem which employs a SIM card and operates over a subscription to a mobile operator such that its functionality is similar to that of a mobile phone. GSM modem has been used in our project for sending SMS (short message service) messages to the customers.

The GSM modem is interfaced to the RPi board through UART communication such that the TX (transmitter), RX(Receiver) and GND (ground) pins of the RPi board and the GSM module are used for this purpose.

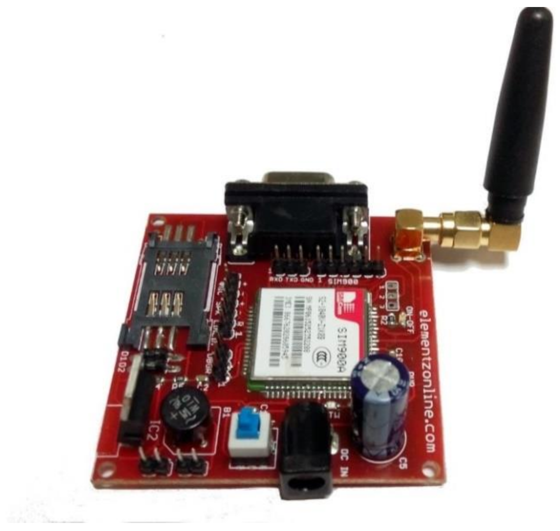


Figure 3.7: GSM modem

We have used a SIM 900A modem board which has the following key features:

- baud rate is configurable from 1200-115200 through AT command
- AT cellular command interface
- Communication through RS232 and TTL
- Default Baud rate : 9600
- Data bits: 8 , parity: none ,stop bit: 1

3.2. Software Specifications

We have used the following APIs and libraries such as APScheduler, gspread library, GPIO library, Google Drive API, HTML and they have been briefly described below:

3.2.1. APScheduler:

As the RPi does not have an interrupt mechanism, scheduling the tasks is the most crucial part of a program. Advanced Python Scheduler is a library for python which lets users schedule the tasks to be performed either once or periodically. It is a flexible solution as old jobs can be deleted and new ones can be created on the fly. If the jobs are stored in a database the scheduler can retrieve it even after a restart. The backends which are supported to store the jobs are SQLAlchemy, MongoDB and Redis.

One of the most important advantages of APScheduler is that it is platform independent. It has three inbuilt scheduling systems that we can use:

1. Scheduling with specific start and end times
2. Interval based Scheduling
3. Run a job once on a specified date or time

The main components are triggers, jobstores, executors and schedulers.

Triggers are the ones that contain the logic for scheduling. Each job has its own trigger which determines when the job should run next.

The jobs to be scheduled are contained in **Job Stores**. The default job store simply keeps the jobs in memory, but there are other job stores to store in different databases. Job stores are the ones that act as middlemen for saving, loading, updating and searching jobs in the memory. Job stores must never be shared between schedulers.

The running of jobs is handled by **Executors**. This is done by submitting the designated callable in a job to a thread or process pool. When the job is executed, the executor notifies the scheduler which then emits an appropriate event.

All the elements of the program are bound together by **Schedulers**. We have one scheduler running in our application. We do not deal with the job stores, executors or triggers directly. Instead, the scheduler provides the proper interface to handle all those. All the tasks like configuring the job stores and executors, adding, modifying and removing jobs is done with the help of the scheduler.

There are different types of schedulers, but the most basic ones are as follows:

- Blocking Scheduler is the one which is used when the entire program needs to be scheduled.

- Background Scheduler is the one to use when we need the scheduler to run in the background as part of the program.

3.2.2. gspread Library:

In order to access the google spreadsheets from the RPi remotely we used gspread and oauth2client library. gspread is used to create a spreadsheet, edit the number of rows and columns etc; while oauth2client is used to verify the credentials of the clients using gspread library.

The main commands of gspread library are as follows:

gspread.login(email, password) : This command is used to login to the Google API

gspread.authorize(credentials): This command takes the oauth2 credentials as argument and allows remote login to Google API

open(title): When a particular spreadsheet has to be opened it can be given as argument to this command. When there are two spreadsheets with same name it opens the first sheet. If no spreadsheet with the given name is found it raises 'gspread.SpreadsheetNotFound'.

open_by_key(key): Each spreadsheet is assigned a specific key by Google. This command is used to open the spreadsheet by the key assigned to it.

open_by_url(url): This command is used to open a spreadsheet with it's url.

open_all(title=None): This command is used to open all the spreadsheets of an account.

Models:

Common spreadsheet objects are called Models. It includes spreadsheets, worksheet and a cell.

class gspread.Spreadsheet(client, feed_entry): This is a class for accessing spreadsheet objects. This class includes the following commands.

add_worksheet(title, rows, cols): This command adds a new worksheet to a spreadsheet with the specified title, rows and columns

oauth2 client is used to provide credentials by Google to edit a spreadsheet. We can access this credentials at Google developers Console and selecting the API for which we need the credentials. There are two ways in which we can use the credentials. We can enter the credentials manually or we can download a JSON file which can be used in our program. JSON stands for Java Script Object Notation. It is a light weight information exchange format which is language independent.

3.2.3. GPIO Library for RPi:

General Purpose Input Output (GPIO) pins are the most distinguishing features of the RPi. They are present on the top end of the board. These pins act as interface between RPi and the outside world. Most of the RPi boards have 40 GPIO pins of which 26 can be used for interfacing and other pins are for power and ground purposes.

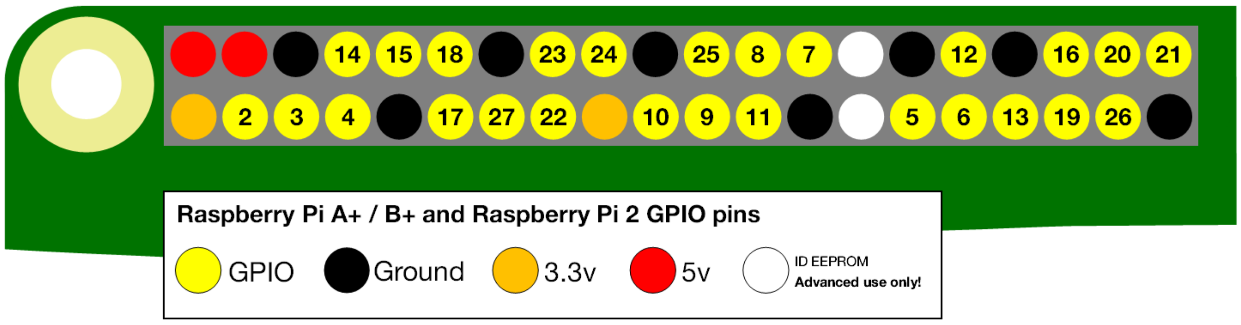


Figure 3.8: GPIO Pin Layout

In order to use the GPIO pins effectively we need to import the GPIO library into our program. The numbering of the pins can be different and we use the board numbering which is used as default all over.

GPIO.setboard(GPIO.board) is the command used to set the numbering to default state.
Similar to other micro controllers the GPIO pins must be declared in advanced as output or input GPIO.setup(pin number, GPIO.IN/OUT) is the command used for configuration.

3.2.4. Google Drive:

Google Drive has a free 15GB file storage capacity for all its users. We used it to store the files related to our website and made the folder public and accessible by anyone on the internet. Google also provides us with a link to the folder which can be used to access our website. The link provided by Google is very long and is difficult to remember. So we used a free domain website to get a domain name for website which can be accessed from anywhere through Internet.

3.2.5. HTML

We designed a website to access the data using HTML. HTML stands for Hyper Text Markup Language. It is used to describe webpages. It contains tags at the start and end of each line which depict every part of the webpage. This tags are called as markup tags and in this case HTML tags. HTML tags are keywords (tag names) surrounded by angle brackets:
<tagname>INFORMATION</tagname>

HTML tags normally come in pairs like <p> and </p>. The start tag is called opening tag and the end tag is called closing tag. The end tag is written like the start tag with a slash before the tag name.

At the beginning of the document it is declared as HTML type. Different tags that we have used are as follows:

- The text between <html> and </html> describes the entire HTML document.
- The text between <head> and </head> gives information regarding the document
- The text between <title> and </title> contains title for the document
- The text between <body> and </body> is the visible page content which is seen by the user

- The text between <h1> and </h1> describes a heading
- The text between <p> and </p> depicts a paragraph

3.2.6. Selection of Operating System for RPi

There are a number of operating systems available to work on RPi. We had to select an operating system which is easy for installation, the looks and usability of the OS should be user friendly, the community behind the OS should be supportive so that when a problem arises there would be support from developers to get a help from. From the wide open options we have shortened ourselves to 5 options which are as listed below:

- **Raspbian**
- **Risc OS**
- **Plan 9**
- **Android**
- **Arch**

Raspbian: This is the most widely used operating system by RPi users. It is the officially supported operating system by the manufactures of the Raspberry Pi. From the outset, Raspbian has given the user a bland, but also a functional desktop. The RPi's resources are kept well in hand and were not wasted on inefficient interface which looks good to the eyes. Coming to usability, Raspbian looks quick and fast initially but the availability of a large number of apps and after installation of high number of programs, the user would feel the Operating System lagging.

Arch: This operating system displays only the Terminal. Arch is a streamlined OS, the system responds quickly even after installing a desktop environment such as OpenBox,. The usability Arch Operating System depends on what you want to install, which makes Arch OS, the most configurable and usable OS.

Android: Android is a good looking operating system. The current Android Project is still in the development stages. Working on the 256MB RPi becomes troublesome but as we are using a 1GB RPi, it's usable. The operating system doesn't look like the one which we could use daily as it's still in its development stages, which means there are some bugs in the operating system.

Risc OS: Similar to Raspbian, Risc OS has a good looking GUI. The resolution is at 1080p. Initially, it will be a bit difficult to use and requires some previous experience to get off the ground. For example, by default, the Ethernet port is disabled – to enable it there are clear instructions, but a newcomer will lose his interest after going through all those instructions. But after using the Risc OS for some time we can feel comfortable with the interface and features.

Plan 9: This operating system is completely different from other operating systems but there is plenty of documentation available so that we could learn and implement. Rio is the Plan 9 windowing system, the interface looks like Amiga and Atari ST days. It takes some time to get used to this operating system as we get overwhelmed by the sheer amount of documentation reading material.

Final selection of OS:

From the above options, we have chosen Raspbian as the operating system we can operate because it's the easiest operating system to install and community support for this is very high as it's the official operating system supported by the manufacturer of RPi itself.

There are two versions of Raspbian

- Jessie
- Wheezy

Jessie is the latest version of Raspbian but we have used the wheezy version because of a problem faced after installing the Jessie OS where the Wi-Fi doesn't connect to any networks, there wasn't any support in the community. So we have tried the wheezy OS which connected properly to the Wi-Fi networks.

3.2.7. Basic Linux Commands Used

- **apt-get update:** The version of Raspbian gets updated.
- **apt-get upgrade:** All the installed software packages gets upgraded.
- **clear:** terminal screen gets cleared.
- **nano abc.txt:** The file abc.txt in "Nano", the Linux text editor is opened.
- **poweroff:** Shutdowns the RPi immediately
- **raspi-config:** Configuration settings menu is opened.
- **reboot:** Immediately the RPi is rebooted.
- **shutdown -h now:** shutdown immediately.
- **shutdown -h 12:30 :** Shutdown at a later time.
- **startx:** The Graphical User Interface(GUI) is opened.
- **cat xyz.txt:** Contents of file example.txt is displayed.
- **cd /xyz/pqr:** Current directory is changed to the /xyz/pqr directory.
- **ls -l:** All the files in the current directory are listed along with file size, date modified, and permissions.
- **mkdir abc_directory**: abc_directory, inside the current directory a new directory will be created.
- **mv abc:** Moves the file or directory named abc to a specified location. For example, *mv filesample.txt /home/pi/office/* moves *filesample.txt* in the current directory to the */home/pi/office directory*.
- **rm xyz.txt**: xyz.txt file will be deleted.
- **rmdir xyz_directory:** If the directory xyz_directory is empty it will be deleted.
- **ifconfig:** The presently being used wireless connection status will be displayed (whether an IP address has been acquired by wlan0).
- **df -h:** The disk space available information will be displayed.
- **lsusb:** Hardware connected to RPi's USB ports are listed.

4. DESIGN APPROACH

In this chapter the approach followed is described in detail starting from RPi, Wi-Fi module Setup. The reason we have chosen Python for programming over C++ has been explained in detail. Then we have explained different libraries used and the code snippet that is part of the library with their particular function. We included flow chart of the algorithm in figure 4.3 .

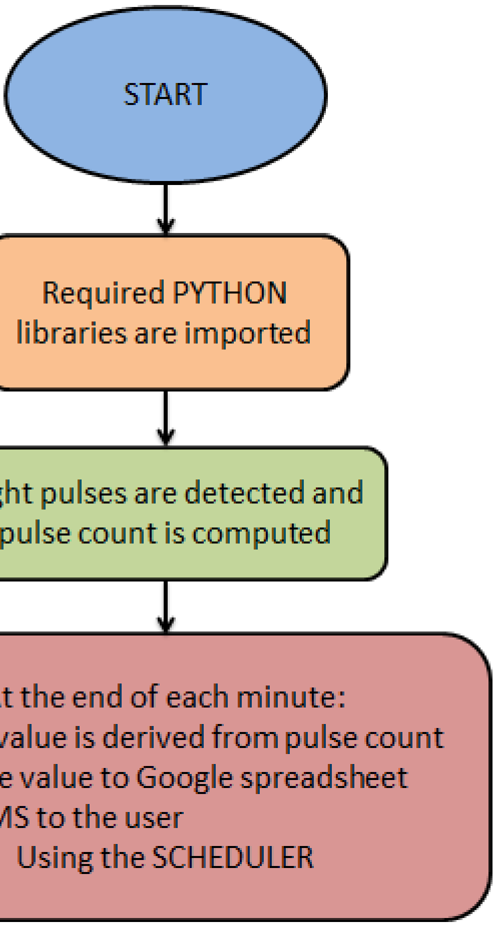

START
Required PYTHON libraries are imported
Light pulses are detected and pulse count is computed
At the end of each minute:
1.Energy value is derived from pulse count
2.Send the value to Google spreadsheet
3.Send SMS to the user
Using the SCHEDULER

Figure 4.3: block diagram

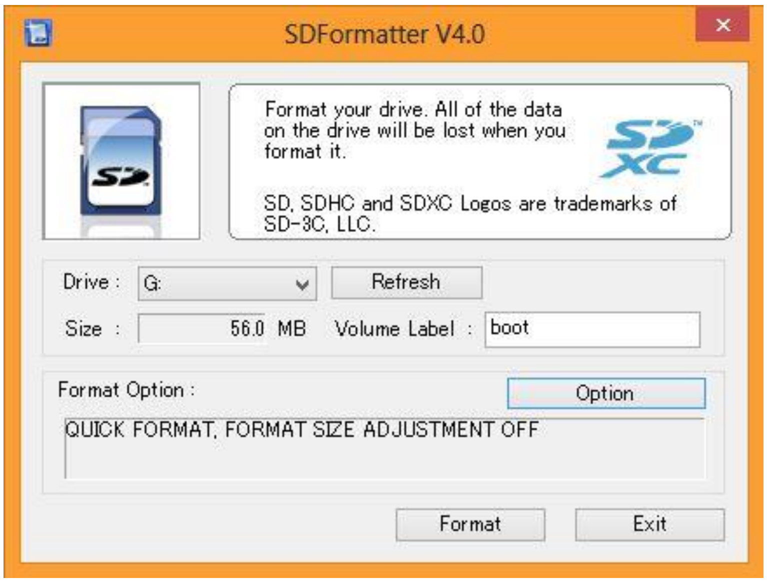
4.1. RPi and its Remote Access

An operating system is needed to get started with RPi. A minimum of 8 GB SD card is recommended for use by the RPi foundation. The most basic method is to install the operating system using NOOBS (New Out Of Box Software) which is an easy operating system installation manager. NOOBS pre-installed SD cards are available in the market which can be inserted directly into the card slot of the RPi and the external peripherals such as mouse , keyboard and monitor cables are also plugged into the RPi board.

Upon connecting the USB power cable to the RPi board , it will start booting and display a list of operating systems that can be installed such as Raspbian , Pidora , OpenELEC, OSMC, RISC OS etc. Raspbian OS which is Linux based distribution is the recommended one.

Alternatively, the required OS can be downloaded from the official website for free into a general microSD card. For our project, we have used the Raspbian Wheezy version for our RPi board. The latest version of Raspbian OS is the Jessie version but it has posed some trouble in connecting to the wireless networks using WiFi adapter. Hence the earlier version of Wheezy was used.

We used the second method for setting up our RPi board. The microSD card is formatted using SDFormatter and the downloaded image of Raspbian Wheezy is written into the SD card using win32 disk imager.


SDFormatter V4.0

Format your drive. All of the data on the drive will be lost when you format it.

SD, SDHC and SDXC Logos are trademarks of SD-3C, LLC.

Drive : G:

Size : 56.0 MB

Volume Label : boot

Format Option :

QUICK FORMAT, FORMAT SIZE ADJUSTMENT OFF

Figure 4.2: SDFormatter

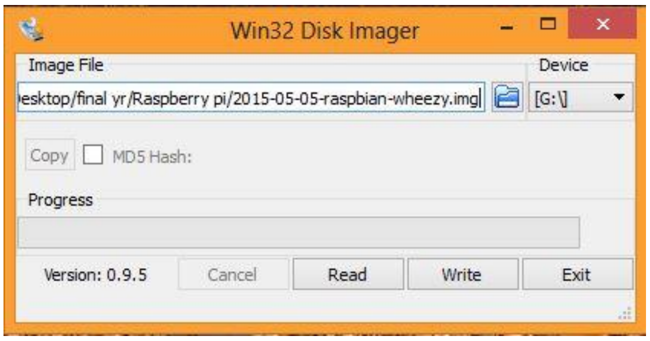

Win32 Disk Imager

Image File: esktop/final yr/Raspberry pi/2015-05-05-raspbian-wheezy.img

Device: [G:\]

MD5 Hash:

Progress

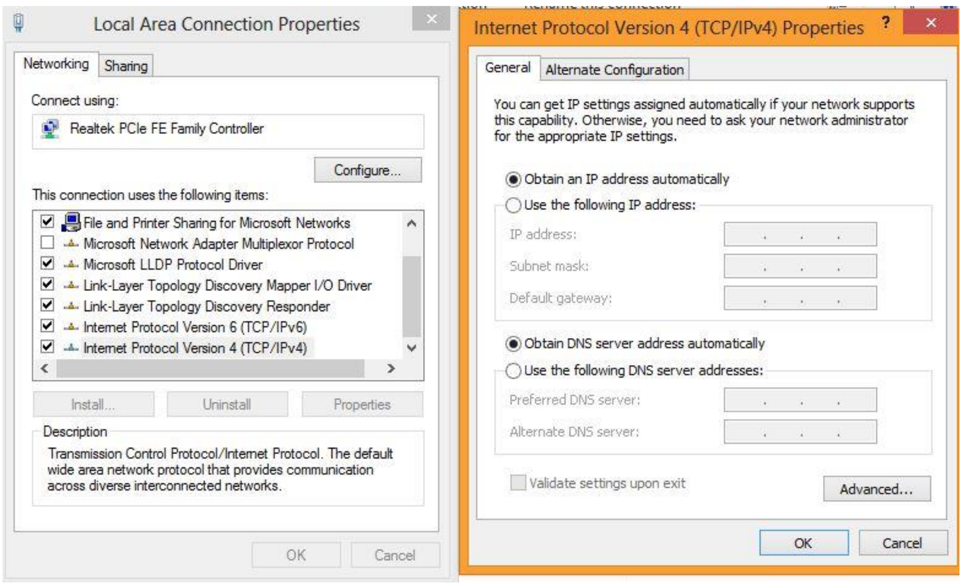
Version: 0.9.5

Figure 4.3: Win32 Disk imager to write Raspbian Wheezy OS to the SD card

After completing the above processes and inserting the microSD card into the RPi board, it boots up after powering on the board and it is ready to use. As it is more convenient to access the RPi board using a laptop rather than connecting the RPi board to external peripherals and using it like a regular desktop system. SSH (secure shell) can be used to connect to the RPi from a Windows computer by installing additional software known as PuTTY. By means of Ethernet LAN connection or wireless connection using WiFi adapter to the RPi board, the RPi can be operated without using a physical display monitor or screen, known as the headless mode.

The steps to access RPi shell and desktop directly on laptop using Ethernet LAN connection:

- The RPi and Laptop are connected via the Ethernet cable while the board is powered off.
- LAN properties in the network and adapter settings is opened and IPV4 properties is set to option of obtaining the IP address automatically.


Local Area Connection Properties

Networking | Sharing

Connect using:

Realtek PCIe FE Family Controller

Configure...

This connection uses the following items:

- [✓] File and Printer Sharing for Microsoft Networks
- [ ] Microsoft Network Adapter Multiplexor Protocol
- [✓] Microsoft LLDP Protocol Driver
- [✓] Link-Layer Topology Discovery Mapper I/O Driver
- [✓] Link-Layer Topology Discovery Responder
- [✓] Internet Protocol Version 6 (TCP/IPv6)
- [✓] Internet Protocol Version 4 (TCP/IPv4)

Install... | Uninstall | Properties

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK | Cancel

Internet Protocol Version 4 (TCP/IPv4) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

● Obtain an IP address automatically

○ Use the following IP address:

IP address: . . .

Subnet mask: . . .

Default gateway: . . .

● Obtain DNS server address automatically

○ Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

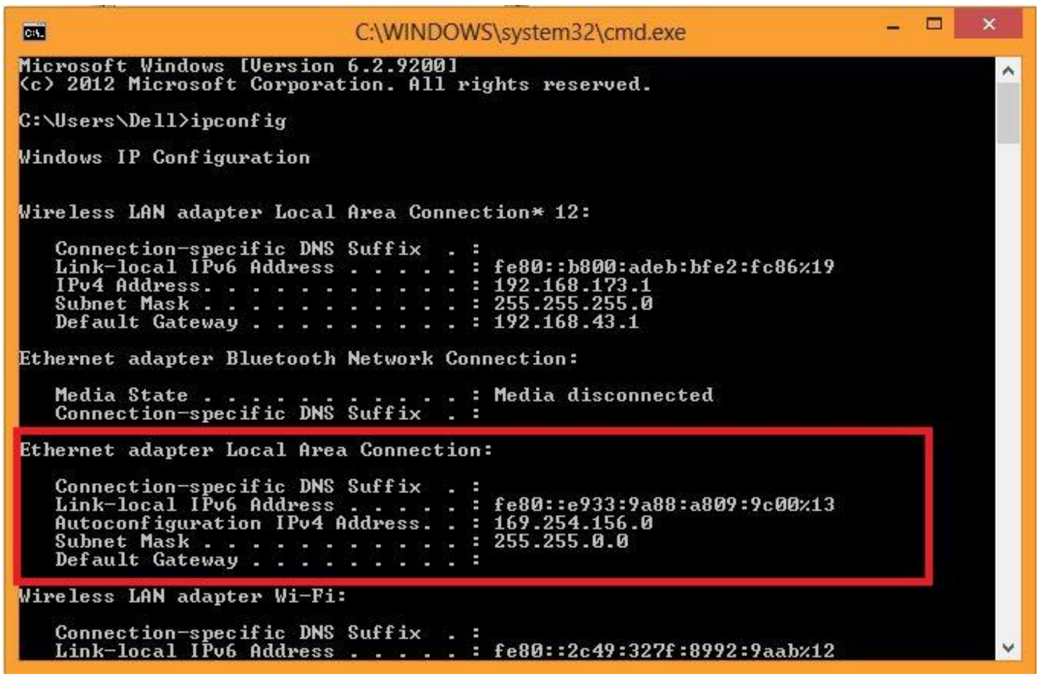
[ ] Validate settings upon exit

Advanced...

OK | Cancel

Figure 4.4: IPV4 properties window

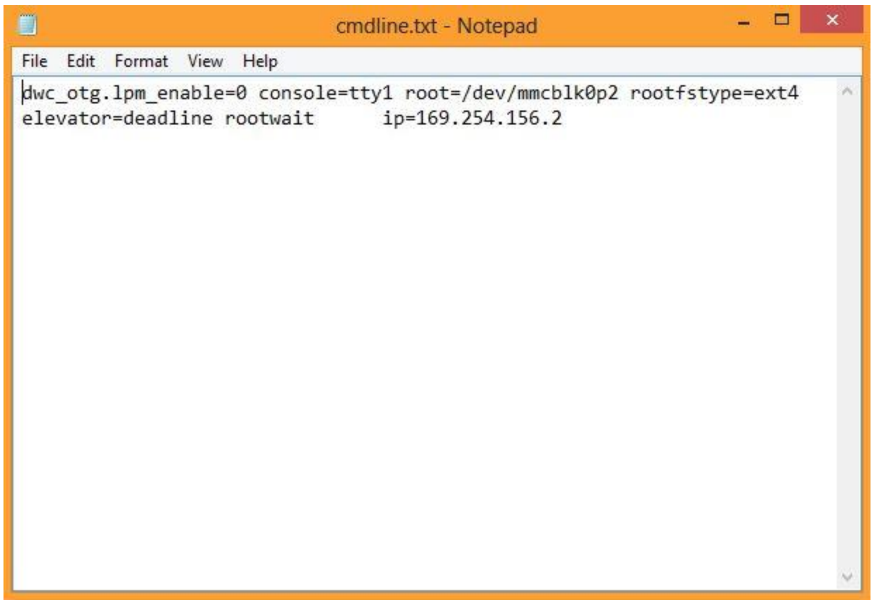
- After powering on the board, the presence of an unidentified network is indicated on the laptop. Open the command prompt and ipconfig command is typed and the result is as shown in figure 4.5. The IP address in the box is the IP address of laptop when connected to the RPi board.



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Dell>ipconfig

Windows IP Configuration


Wireless LAN adapter Local Area Connection* 12:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b800:adeb:bfe2:fc86%19
   IPv4 Address. . . . . . . . . . . : 192.168.173.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.43.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::e933:9a88:a809:9c00%13
   Autoconfiguration IPv4 Address. . : 169.254.156.0
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::2c49:327f:8992:9aab%12
```

Figure 4.5: Command prompt window displaying the IP address of laptop

- Now the RPi board is powered off and the microSD card is taken out and the contents of cmdline.txt file is modified as shown in the figure 4.6.

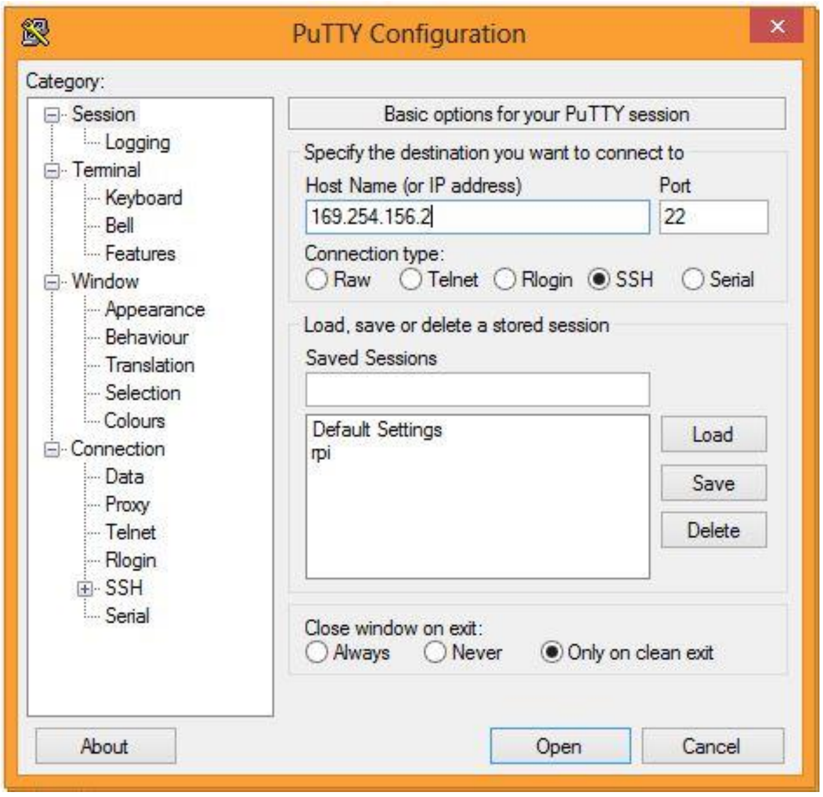


```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait      ip=169.254.156.2
```

Figure 4.6: cmdline.txt file in the microSD card

- The IP address is appended in the cmdline.txt file and it is saved. The IP address is chosen in accordance with the LAN IP address.

- After powering on the board and inserting the microSD card , the IP address is entered on the PuTTY software as shown in the figure 4.7.



Specify the destination you want to connect to

Host Name (or IP address): 169.254.156.2
Port: 22

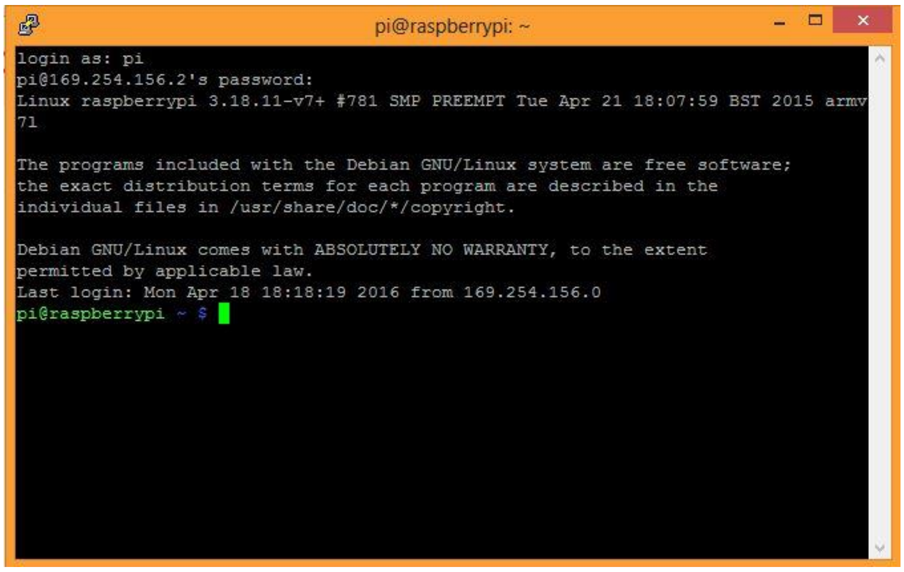
Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

Saved Sessions
Default Settings
rpi

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Figure 4.7: PuTTY software on laptop

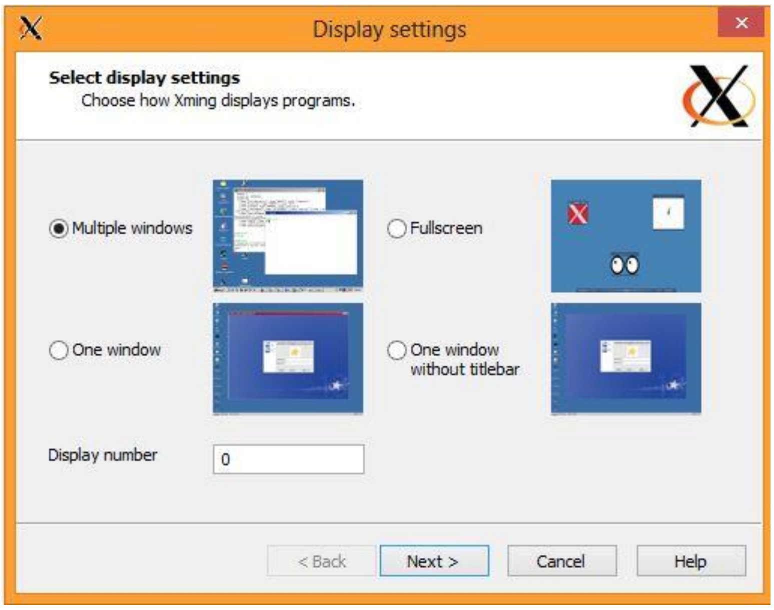


```
login as: pi
pi@169.254.156.2's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv
7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 18 18:18:19 2016 from 169.254.156.0
pi@raspberrypi ~ $
```

Figure 4.8: The terminal window

- After option open is pressed, the LXTerminal window appears as shown in the figure 4.8 and the default username is pi and password is raspberry on new RPi which can be changed by using the command raspi-config.

- In order to access the desktop of RPi , Xming Xserver needs to be installed and Xming should be opened such that it runs in the background. The X-forwarding should be enabled in PuTTY software before opening the terminal window.



Display settings

Select display settings
Choose how Xming displays programs.

(● Multiple windows) ○ Fullscreen ○ One window ○ One window without titlebar

Display number: 0

Figure 4.9: Xming display settings window

- The commands startx and /etc/X11/Xsession are executed one after the other in the terminal window, so as to obtain the desktop as shown in the figure 4.10.

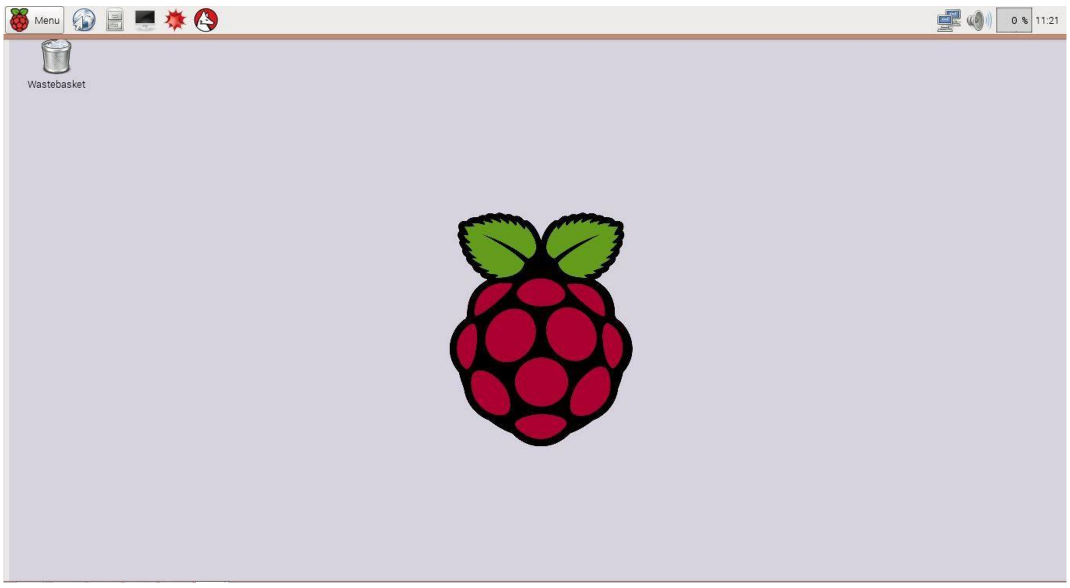


Figure 4.10: Desktop window

4.2. Wi-Fi Module Setup

Wi-Fi modules are readily available for purchase on e-commerce websites like amazon, but be sure to buy the one that supports Linux by default. Now plug the Wi-Fi module into one of the USB ports of the RPi and reboot the RPi. To find out if the module is recognized type the following command in the terminal dmesg | more. It lists out all the connected devices including the Wi-Fi module.

Now in order to configure the Wi-Fi network we use nano text editor. Type the following command
sudo /nano/network/interfaces and edit the file as follows:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
   wpa-ssid "Your Network SSID"
   wpa-psk "Your Password”
```

After editing the file save it to the disk by pressing Ctrl+O and exit the editor by pressing Ctrl+X. For the changes to take effect we have to reload the interfaces by using the command: sudo service networking reload. We can check the status by entering ifconfig command. If the module is connected to the network we should see a valid IP address under net addr field.

There is an easy method to connect to Wi-Fi using GUI but using this method ensures that the module connects to the preferred network by default at the time of startup.

4.3. Sending data to Google spreadsheets

The energy consumed values needed to be sent to the Google spreadsheets periodically. In order to achieve this, an API key is created to access Google services. The following steps have been adopted to do so:

- Create a Google account and go to Google Developers console

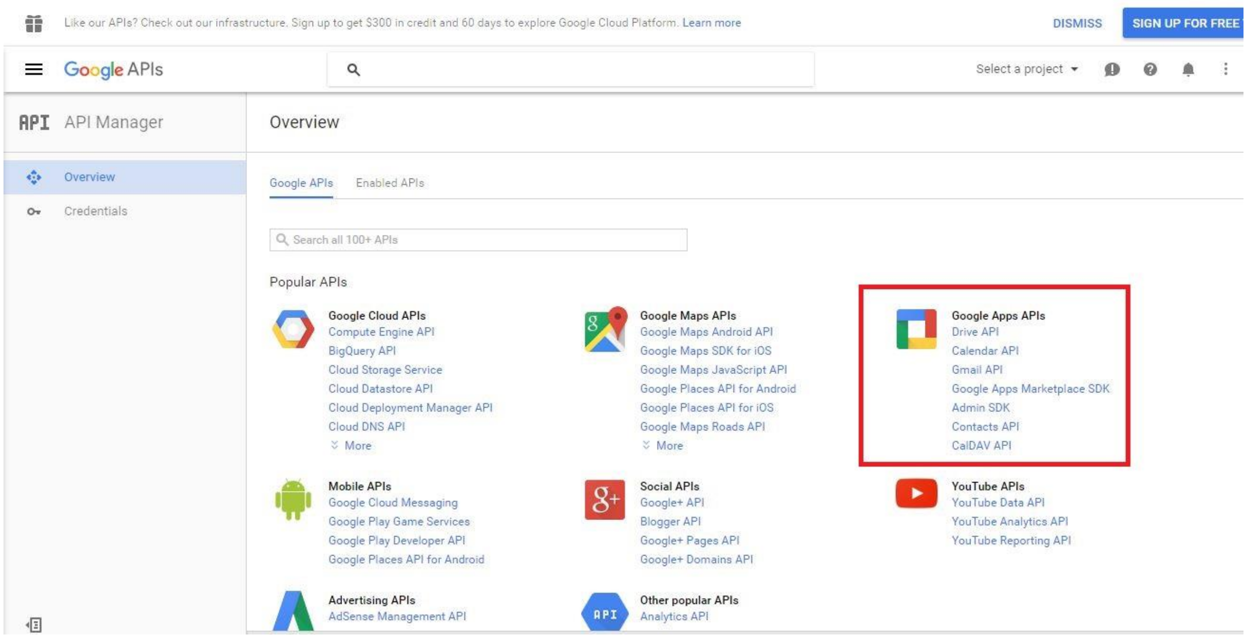
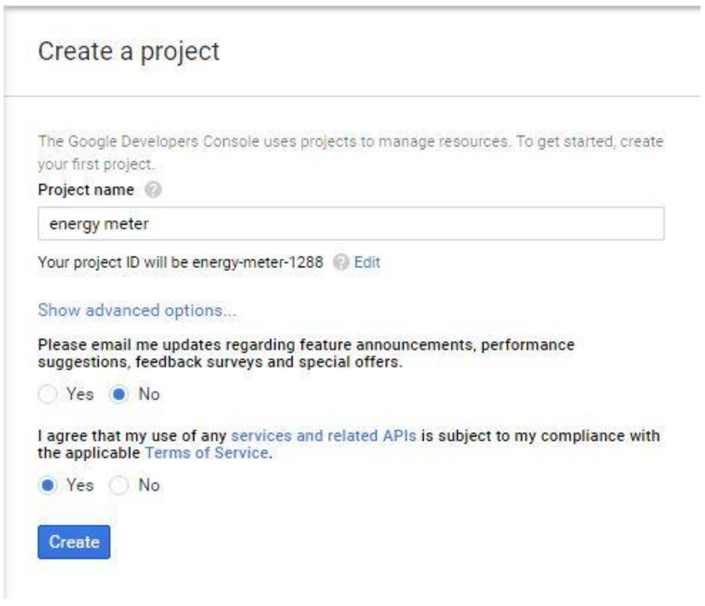

Like our APIs? Check out our infrastructure. Sign up to get $300 in credit and 60 days to explore Google Cloud Platform. Learn more

Figure 4.11: Google Developers console

- Go to Google drive API and create a new project as shown in the figure 4.12.


Create a project

The Google Developers Console uses projects to manage resources. To get started, create your first project.

Project name: energy meter

Your project ID will be energy-meter-1288 Edit

Show advanced options...

Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

Yes [✓ No]

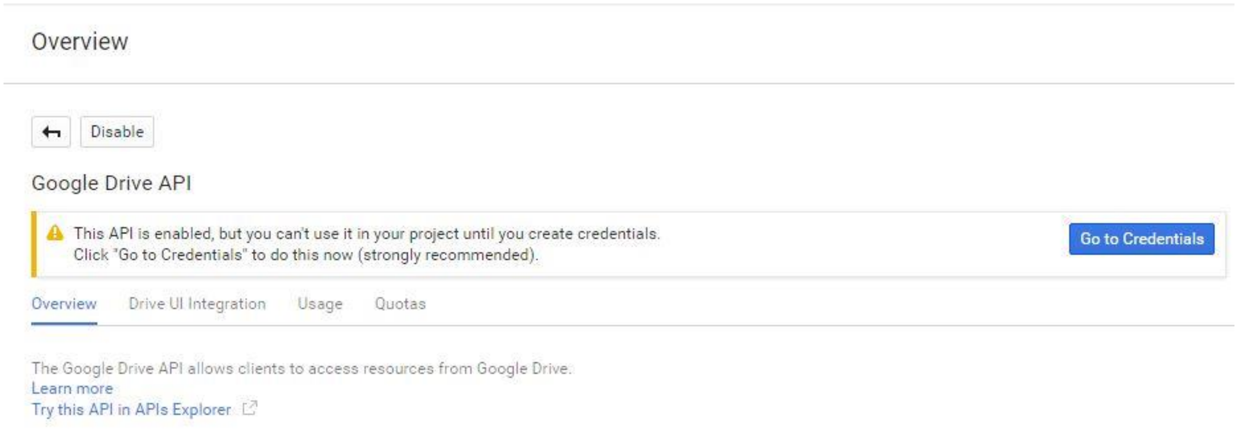
I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.

[✓ Yes] No

Create

Figure 4.12: Create a new project

- Google Drive API is enabled and the credentials need to be created.


Overview

Disable

Google Drive API

This API is enabled, but you can't use it in your project until you create credentials.
Click "Go to Credentials" to do this now (strongly recommended).

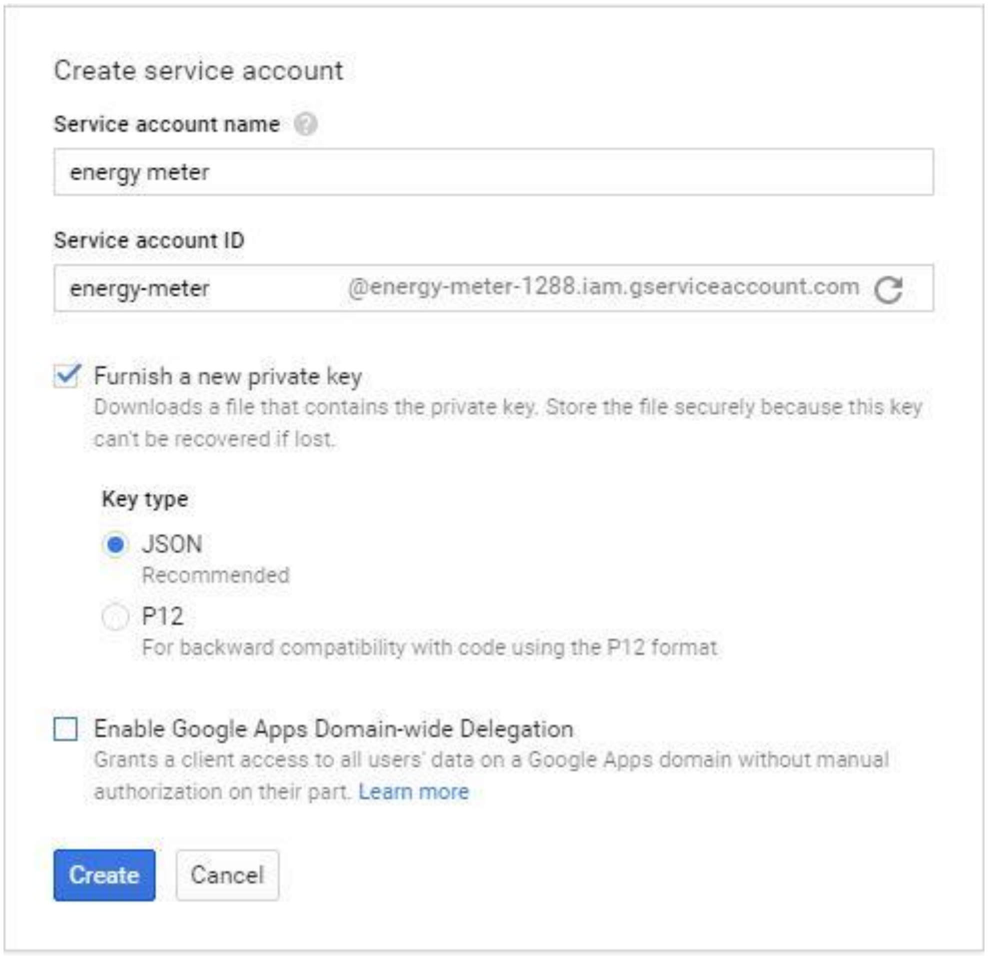
Go to Credentials

Overview · Drive UI Integration · Usage · Quotas

The Google Drive API allows clients to access resources from Google Drive.
Learn more
Try this API in APIs Explorer

Figure 4.13: Google Drive API

- After going to credentials, a service account should be created and choose JSON as shown in the figure 4.14.


Create service account

Service account name

energy meter

Service account ID

energy-meter @energy-meter-1288.iam.gserviceaccount.com

[✓] Furnish a new private key
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

● JSON
Recommended

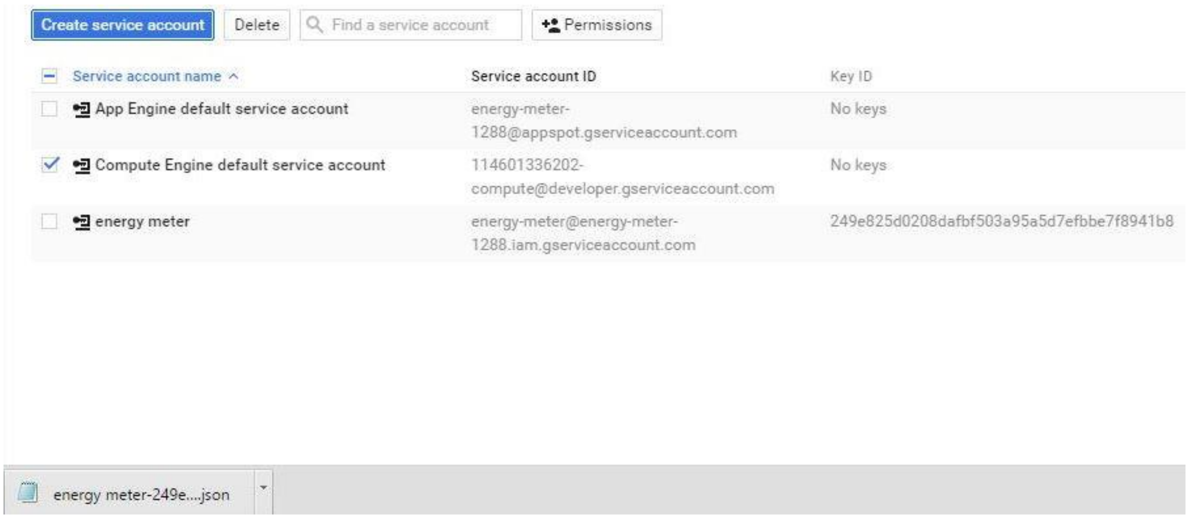
○ P12
For backward compatibility with code using the P12 format

[ ] Enable Google Apps Domain-wide Delegation
Grants a client access to all users' data on a Google Apps domain without manual authorization on their part. Learn more

Create · Cancel

Figure 4.14: Creation of private key

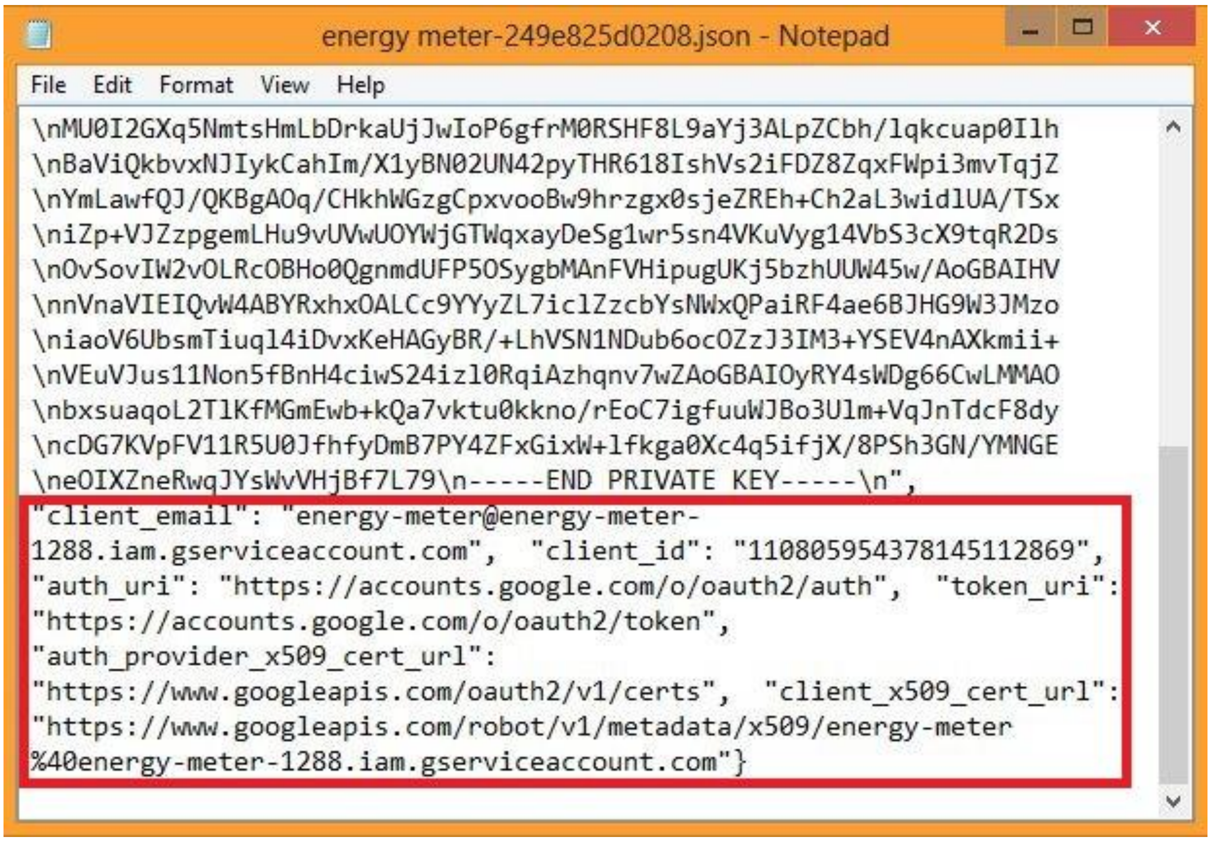
- After the creation of service account, JSON file is downloaded automatically which contains unique private key. The required credentials are in this JSON file. Therefore it should be stored in the same directory which contains the python script.



Service account name	Service account ID	Key ID
App Engine default service account	energy-meter-1288@appspot.gserviceaccount.com	No keys
Compute Engine default service account	114601336202-compute@developer.gserviceaccount.com	No keys
energy meter	energy-meter@energy-meter-1288.iam.gserviceaccount.com	249e825d0208dafbf503a95a5d7efbbe7f8941b8

Figure 4.15: Creation of JSON file

- The contents of the JSON file are as shown in the figure 4.16 and the text in the box will be used in the python script.


energy meter-249e825d0208.json - Notepad
File Edit Format View Help

```
\nMU0I2GXq5NmtsHmLbDrkaUjJwIoP6gfrM0RSHF8L9aYj3ALpZCbh/lqkcuap0Ilh
\nBaViQkbvxNJIykCahIm/X1yBN02UN42pyTHR618IshVs2iFDZ8ZqxFWpi3mvTqjZ
\nYmLawfQJ/QKBgAOq/CHkhWGzgCpxvooBw9hrzgx0sjeZREh+Ch2aL3widlUA/TSx
\niZp+VJZzpgemLHu9vUVwUOYWjGTWqxayDeSg1wr5sn4VKuVyg14VbS3cX9tqR2Ds
\nOvSovIW2vOLRcOBHo0QgnmdUFP5OSygbMAnFVHipugUKj5bzhUUW45w/AoGBAIHV
\nnVnaVIEIQvW4ABYRxhxOALCc9YYyZL7iclZzcbYsNWxQPaiRF4ae6BJHG9W3JMzo
\niaoV6UbsmTiuql4iDvxKeHAGyBR/+LhVSN1NDub6ocOZzJ3IM3+YSEV4nAXkmii+
\nVEuVJus11Non5fBnH4ciwS24izl0RqiAzhqnv7wZAoGBAIOyRY4sWDg66CwLMMAO
\nbxsuaqoL2TlKfMGmEwb+kQa7vktu0kkno/rEoC7igfuuWJBo3Ulm+VqJnTdcF8dy
\ncDG7KVpFV11R5U0JfhfyDmB7PY4ZFxGixW+lfkga0Xc4q5ifjX/8PSh3GN/YMNGE
\neOIXZneRwqJYsWvVHjBf7L79\n-----END PRIVATE KEY-----\n",
"client_email": "energy-meter@energy-meter-
1288.iam.gserviceaccount.com",  "client_id": "110805954378145112869",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",  "token_uri":
"https://accounts.google.com/o/oauth2/token",
"auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/energy-meter
%40energy-meter-1288.iam.gserviceaccount.com"}
```

Figure 4.16: Contents of JSON file

4.4. Python over C++

RPi can be programmed using Python and C++. Python and C++ are fundamentally different languages with some edge over the other one. Some of the advantages that made us choose Python over C++ are as follows:

Memory Management:

C++ doesn't have garbage collection which encourages use of raw pointers to manage and access memory. C++ differentiates between heap and stack which requires us to attend to values versus references. C++ requires much more attention to bookkeeping and storage details which is time wasting and generally not necessary.

Types:

While using a variable in C++ they have to be declared as a specific data type at the start of the program. During the execution the variables and their datatypes are inspected and execution stops if there is any error. In Python it is not required to declare data types.Python’s types are also an order of magnitude simpler. The simplicity and the lack of declarations help a lot of people move faster.

Language Complexity:

C++ is a very complex language. The specification document is 775 pages of language legalese, and even the expert C++ developers can be caught up short by unintended consequences in complex code. Python is much simpler and can be mastered with some effort.

Interpreted vs compiled (implementation):

C++ is almost always compiled. Compilation is not done for Python. It's common practice to develop in the interpreter in Python, which is great for rapid testing and exploration. C++ developers do not have the ease to do this. This was one of the most important advantages. Direct interpretation in the case of Python increases the latency which is very important for real time applications.

C++ provides many ways for developers to code a particular task while Python tries to keep it simple. This is one of the reasons why python is easy to get used to.

4.5. Methodology and Python libraries :

The following libraries are imported in the python code:

- RPi.GPIO :

  The GPIO pins are required to interface the physical devices to the RPi Board. In order to access and use these GPIO pins, RPi.GPIO package is imported in the python program which is usually pre-installed in the latest versions of the Raspbian Operating System. Generally, this package is imported as follows:

  ```
  import RPi.GPIO as GPIO
  ```

  This is done so in order to refer to the package as GPIO throughout the program.

  Numbering of IO pins can be done in two ways. The first method is called as BOARD numbering system in which the pin numbers are referred to by the P1 header of the RPi board. The advantage of this method is that it is independent of the board revision and will always work. The same code can be used in all the models and versions of RPi board. In our python program, we have used this type of numbering system.

  The other method of numbering is known as BCM numbering system. In this system, the channel numbers on the Broadcom SOC are referred. The mapping of channel number onto the pin number on the board needs to be worked out. So this numbering system, which is a lower level of working, is dependent on the board revision.

  In order to set the mode of numbering system to be followed:

  ```
  GPIO.setmode(GPIO.BOARD)
  ```

  or

  ```
  GPIO.setmode(GPIO.BCM)
  ```

  The GPIO channel needs to be configured as input or output pin. It is set up as follows:

  ```
  GPIO.setup(channel, GPIO.IN)
  ```

  The value of an input pin is read as follows:

  ```
  value = GPIO.input(channel)
  ```

  where the channel is as per one of the two above mentioned numbering systems and this will return 0 or 1 (Or true/GPIO.HIGH , false/GPIO.LOW).

In our project, we configured the GPIO pin 7 as input channel with numbering system as BOARD as follows:

```
GPIO.setmode(BOARD)
GPIO.setup(7, GPIO.IN)
value = GPIO.input(7)
```

To find out the information about RPi :

```
GPIO.RPI_INFO
```

In our project, the input pin 7 is read continuously and the returned value is stored in a variable(say value).When the light is incident on the light sensor module which feeds its digital output to the pin 7, a string of zeros is returned upon using the input function, which corresponds to the detection of light pulses. A string of ones is returned when no light is incident upon the light sensor module.

The current value of the variable value is compared with previous value stored in another variable(say prevval). If both the values in these variables do not match and current value variable is equal to 0, then the variable containing the value of pulse count(say elvalue) should be incremented. The python code snippet corresponding to this algorithm of reading the input and storage of pulse count is as follows:

```
while(True):
        prevval = value
        value = GPIO.input(7)
        if(value == 0 and value != prevval):
                elvalue = elvalue +1
        else
                pass
```

The above code is crucial to compute the value of energy consumed in kWh.

This also can be illustrated by a flowchart in the figure 4.17:

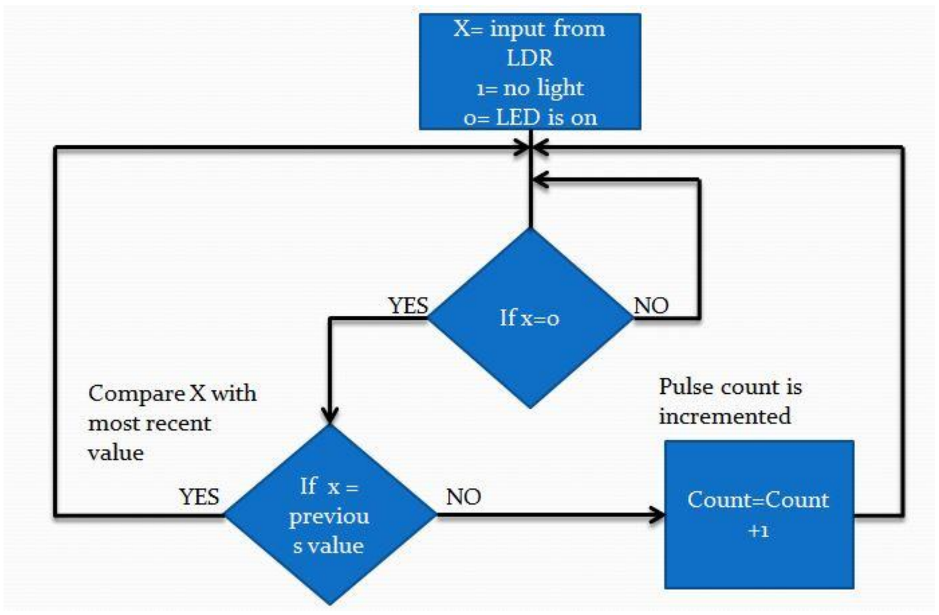


Figure 4.17: Flowchart to compute pulse count

- **Scheduler:**

  The APscheduler doesn't come pre-installed in the Raspbian OS so it can be installed by the following command in the LXTerminal window:

```
sudo pip install apscheduler
```

Certain new python packages which are not available in the Raspbian archives can be installed by using the pip tool from Python Package Index(PyPI).
The scheduler is imported in the python code as follows:

```
from apscheduler.scheduler  import Scheduler
```

This is the simplest way to schedule a job. Using this scheduler, the job can be executed at a specified date and time or at fixed intervals. For our project, the energy value needs to be exported to the Google spreadsheet periodically.

The code snippet to start the scheduler and schedule the job is as follows:

```
#start the scheduler
sched =  Scheduler()
```

```
sched.daemonic = False
sched.start()
```

The second line sched.daemonic = False is to tell the scheduler not to run as daemon or background process.

```
#schedule the job once every minute
@sched.interval_schedule(minutes=1,misfire_grace_time=15)
def   job_function():
        """"  Code to perform the following:
        i)Compute the pulse count for current month or minute
        by subtracting cumulative count with previous one
        ii)Compute the energy consumed value and print it  to
        spreadsheets and append a row in the spreadsheet.
        iii)Sending SMS to the user through serial UART
        communication.
        """"
```

The job is scheduled to be performed after every 1 minute. For the purpose of our project, the duration of 1 minute has been taken instead of 1 month.

In order to avoid any fatal errors due to time mismanagement, we can specify a grace time which gives extra time for the scheduled tasks to complete. For this purpose, misfire_grace_time is assigned as 15 seconds in our project.

- gspread , oauth2client.client , json :

  The special python library to utilize the Google docs spreadsheets is gspread. OAuth 2.0 protocol is used for authentication and authorization. Google API such as Google cloud storage can act on behalf of the application and the application needs to prove its identity to API in such situations so that access can be delegated to resources for a particular application. For these types of server-to-server interactions, a service account is needed that belongs to the application instead of an individual end-user. Your application calls Google APIs on behalf of the service account, and in such cases, user consent is not required.

  The necessary softwares are installed by executing the following commands:

```
sudo apt-get install python-pip
sudo pip install gspread oauth2client
sudo apt-get install python-openssl
```

The python code snippet to export data to Google spreadsheets:

```
 #import necessary libraries
import gspread
import oauth2client.client
import json

#json filename for credentials
JSON_FILENAME = ‘Energy meter-1353d119396a.json’

#Google sheet to save to
GSHEET_NAME = ‘Energy meter readings’

#load credentials from json and open spreadsheet for writing
json_key = json.load(open(JSON_FILENAME))
creds = oauth2client.client.signedJwtAssertionCredentials(json_key(‘client_email’),
                         json_key[‘private_key’],[‘https://spreadsheets.google.com/feeds’])
client_inst = gspread.authorize(creds)
gsheet = client_inst.open(GSHEET_NAME).sheet1
```

The credentials of service account, which are obtained from the Google Developers console, include a unique generated email address, a client ID and one private key pair. The Client ID and private key are used to create a signed JWT(JSON Web Token) and an access token request is produced. The application then sends the token request to Google authorization server, which gives back an access token.The token is used to access a Google API. The process is repeated in case of expiration of token. This process can be illustrated as shown in the figure 4.18.

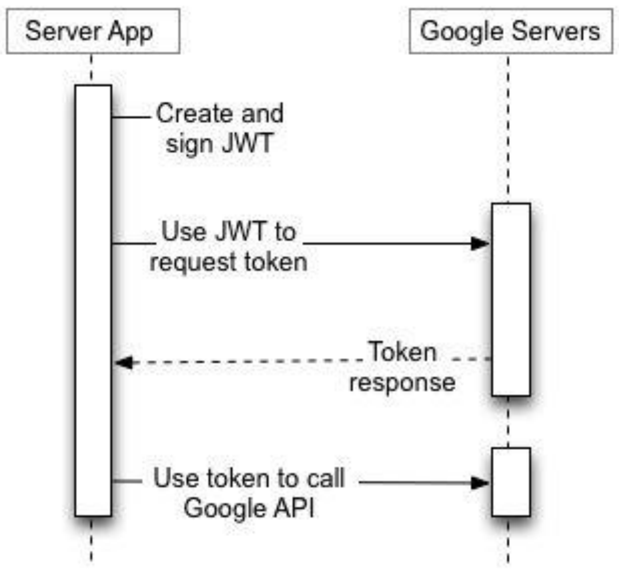


Figure 4.18: Authentication and Authorization

- **Serial Communication Library**

GSM module uses Universal Asynchronous Receiver Transmitter(UART) to transmit data serially to the RPi. The UART is a device that translates data between parallel and serial communication devices. UARTs are commonly used in along with communication standards like RS-232, RS-422 or RS-485. The data format and transmission speeds are configurable according to Universal designation. An external driver circuit handles the electrical signalling levels and methods.

A UART is an individual integrated circuit (IC) which is used for serial communications in a computer or peripheral device serial port. Many modern micro controllers come with a UART device by default . Some devices require more than one UART chip. Two UART chips together used are called Dual UART and eight UART chips used together is called OCTA UART.

Process:

The UART accepts bytes of data and send the individual bits in a serial fashion. The bits are then re assembled into complete bytes by a second UART at the receiver. The fundamental part of every UART is a shift register which is pivotal to convert data from parallel to serial form. Serial transmission is most preferred because it uses only one wire when compared to multiple wires in the case of parallel transmission.

The external signals from other devices are generated by other devices and merely processed by UART. In order to convert signals to logic levels of external signals to that of UART interfaces are used. There are many different forms of external signals. like RS-232, RS-422 and RS-485 from the EIA. There are different schemes which used current like telegraph circuits and Some signaling schemes do not use electrical wires like optical fiber, IrDA (infrared). There are other forms which use carriers to modulate the message like Amplitude modulation.

There are different types of communication between a transmitter and receiver. Simplex communication is when data can be transferred in one direction only at any given time. Full duplex communication is the most advanced one in which data can be sent in both directions. In Half duplex communication data can be sent in both ways but at a given point of time only one direction data transfer is possible.

Data Framing:

No data stage is high voltage or powered state. Before the actual data is sent an active low bit is sent as to indicate the start of communication followed by configurable number of data bits. At the end we can use a high stop bit and a parity bit to prevent errors. Depending on our convenience we can either send the least or most significant bit first.

Receiver:

A clock signal which runs at a multiple of the data rate usually 8 times the bit rate is used to control all the hardware operations. The receiver checks for the start bit at every clock pulse. The character received is valid if the start bit before it is half of the bit time if it is less than that it will be considered as random signal. The received bits are made available to the receiver if it receives all the bits. UART uses a flag or an interrupt to indicate the data. The communication signal is the only common thing between different communicating UARTs. The UARTs generally reconfigure their internal clock with that of the communicating devices to detect the data properly.

Most UARTs have a buffer to store the data which provides sufficient time to store, receive and transmit the data without losing valuable information.

Transmitter:

Transmission is a simple operation compared to receiving as the sender has no reason to worry about time synchronisation. After sending the data the UART shifts the data to the right and adds a start bit and sends the data to the receiver. A stop bit and parity bits can also be added if required.

We used AT commands to communicate with the GSM module (ATtention). Every command starts with AT. “AT+CMGS” is a command used to send message. We need to provide some delay after send command because the GSM module is slow or else it would result in “no response error”. The data to be sent in the SMS is given to the GSM module through UART Communication (Tx, Rx).

First serial communication is initiated by the following command:

ser=serial.serial(‘/dev/ttyacm0’,9600,timeout=1).

- ttyacm0 is the port at which serial communication takes place
- 9600 is the baud rate of serial communication
- timeout=1 is the time for which the RPi has to wait for data

The port is open for communication by using ser.open() command. Then the number to which SMS is to be sent is specified by using ser.write(‘AT+CMGS=“number is specified here”\r’). After this the message is given by using ser.write(“message here”) command. In order to specify the characters used ser.write(chr(26)) command is used. After every command it is advised to use time.sleep(3) command in order to allow enough time to complete the message transfer. At the end the serial port is to be closed using ser.close() command.

4.6. Codes and Standards

We have connected the Wi-Fi module to the Raspberry Pi through the USB port. Wi-Fi is a technology that allows electronic devices to connect to a Wireless Local Area Network(WLAN). We have used Wi-Fi module to export data to spreadsheet. Wi-Fi is IEEE 802.11 standard. Global System for Mobile Communication is a protocol developed for communication between different mobile devices. We have used GSM module to send user his consumption data. The GSM module is IEEE 802.21standard. We have used Python language for programming in Raspberry Pi. HTML and CSS have been used to design the website used.

4.7. Design Constraints, Alternatives, Trade-offs

4.7.1. Significant realistic design constraints

- **Power Consumption**: The RPi does not have any low power modes or interrupts. So the device has to be on continuously which adds to the power consumption.
- **Use of an Energy Meter with RS-485 port**: We initially planned to use a meter with RS-485 port, but were not able to get one even after repeated attempts. Using a meter with RS-485 port would have eliminated the use of LDR module.

4.7.2. Alternatives:

- We have used a LDR module to count the number of pulses and there by determining the energy usage according to the specification of the energy meter. Using a camera module to detect the energy usage is an alternative to this process. Even though it may be a little more accurate the cost of the module is 10 times that of the LDR module
- There are some energy meters with RS 232 from which we can extract the usage data with the help of a RS-485 to DB-9 connector. But many of the conventional meters do not have this option.

4.7.3 . Significant Trade-Offs

Considering our design and the available option, some of the significant trade-offs we faced and the solution chosen are listed below:

- **Use of RPi versus Arduino:** We have chosen to use a RPi because it can act as a single point for all future smart home applications. An Arduino on the other hand can be configured to perform a single task at any given time. The presence of Operating system facilitates multi-tasking which can be used for other applications such as water usage billing.

- **Use of Wi-Fi versus Zigbee protocol:** The ZigBee technology is designed to carry small amounts of data over a short distance while consuming very little power. On the other hand, WiFi is a local area network (LAN) that provides internet access within a limited range. In our application, WiFi seems to be a better option as the data uploaded to the internet is accessible everywhere if wireless networks are present.

5 . SCHEDULE, TASKS AND MILESTONES

There are four major milestones with respect to our project. The four milestones are:

1. Extracting data from LDR module
2. Exporting data from RPi to Google Spreadsheets
3. Designing a website which made the data available to the end user
4. Using the GSM module to send the user data periodically to his mobile phone

After the initial component survey we divided the work among the team allotting particular tasks to each member. Most of the tasks required the entire team effort and sometimes this helped to get a new perspective towards a challenging problem faced by a particular member.

Several online forums and open source libraries were referred and used to solve different bugs in the code. A usability testing was performed at regular intervals to make required enhancements. For example, while implementing the scheduler initially would result in task not executed error due to the multi-processing nature of the RPi. Later we used a misfire grace time function to ensure that enough time was provided for the RPi to perform the scheduled tasks.

Expert opinions and critique from the panel members helped us to improve our project significantly. A Gantt chart outlining important tasks and goals can be seen in Appendix A.

6. PROJECT DEMONSTRATION

Three 60 W bulbs were used as electrical loads connected to the energy meter and for the energy meter which we used, 1kWh corresponds to 3200 pulses. So the energy consumed by the load over a period can be derived from the pulse count. The time interval of 1 minute has been taken instead of 1 month for testing purposes. The complete system is as shown in the figure 6.1.

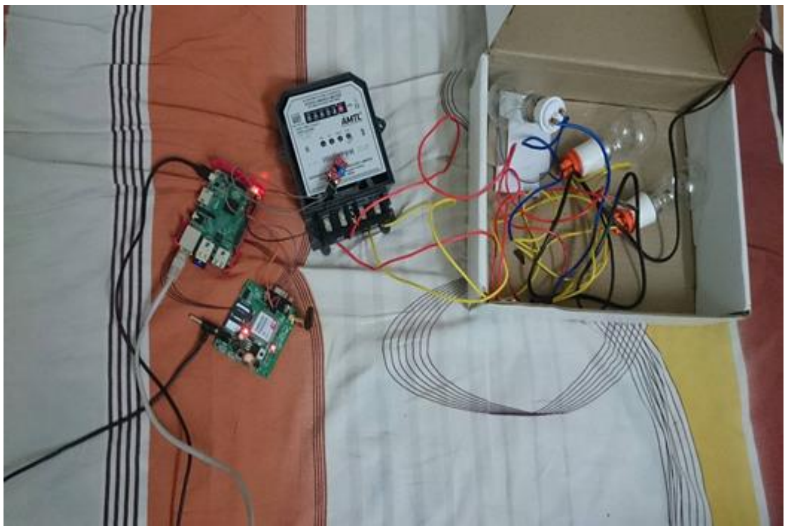
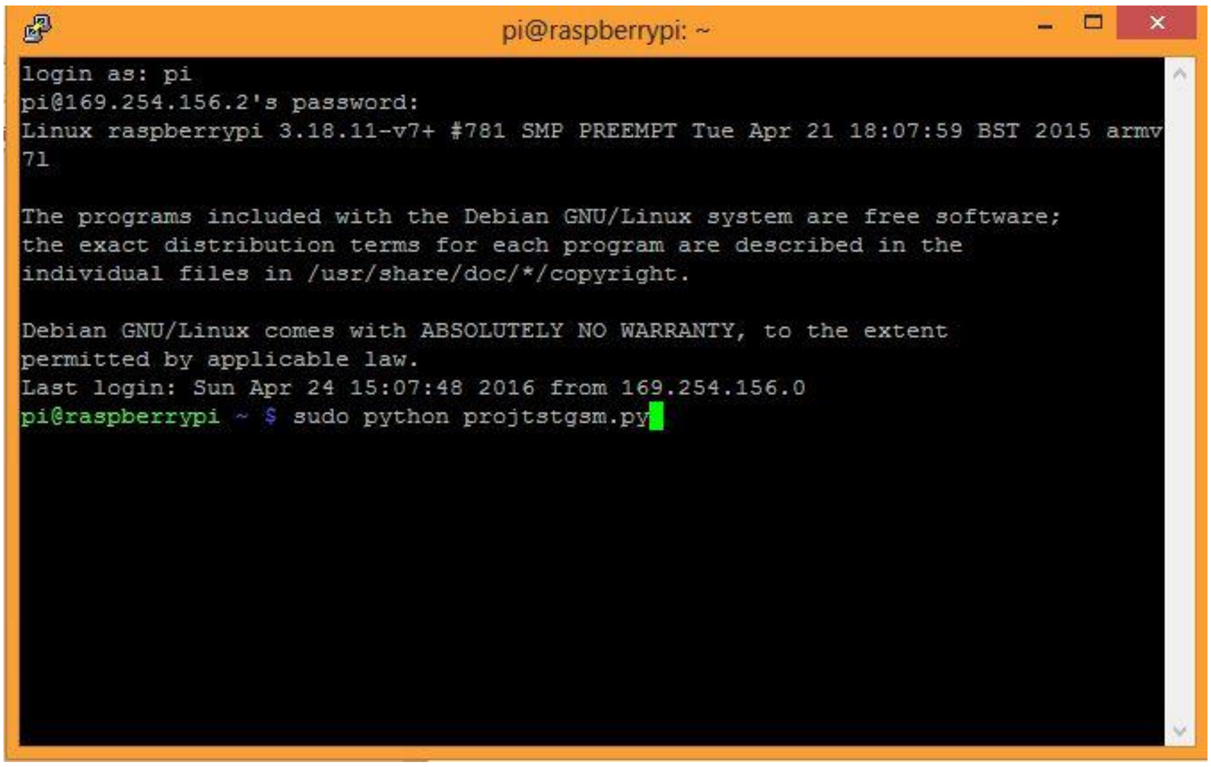


Figure 6.1: Complete system setup

Project Walkthrough:

1. Connect the three bulbs in parallel and power supply to the energy meter according to its connection diagram
2. Power on the GSM module and RPi board. Make sure that the Network LED on the GSM module is blinking and the power LED of RPi board and GSM module are solid red to ensure normal functioning.
3. Place the LDR module over the CAL LED of the energy meter such that the output indicator LED is ON whenever a pulse is detected.
4. Open the PuTTY software after connecting the RPi board to the laptop via Ethernet cable. Enter the IP address as explained in RPi setup section of earlier chapter.
5. After logging in with valid username and password, the python script needs to be executed by using the following command as shown in the figure 6.2.

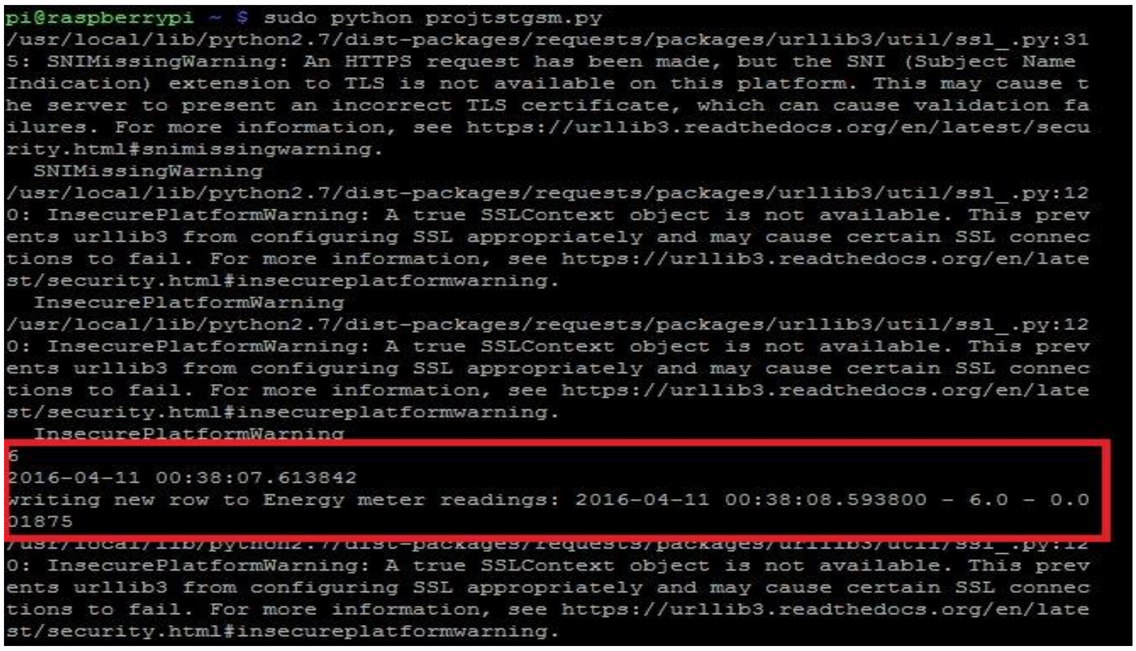
sudo python projtstgsm.py



```
login as: pi
pi@169.254.156.2's password:
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv
7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 24 15:07:48 2016 from 169.254.156.0
pi@raspberrypi ~ $ sudo python projtstgsm.py
```

Figure 6.2: Terminal window after login

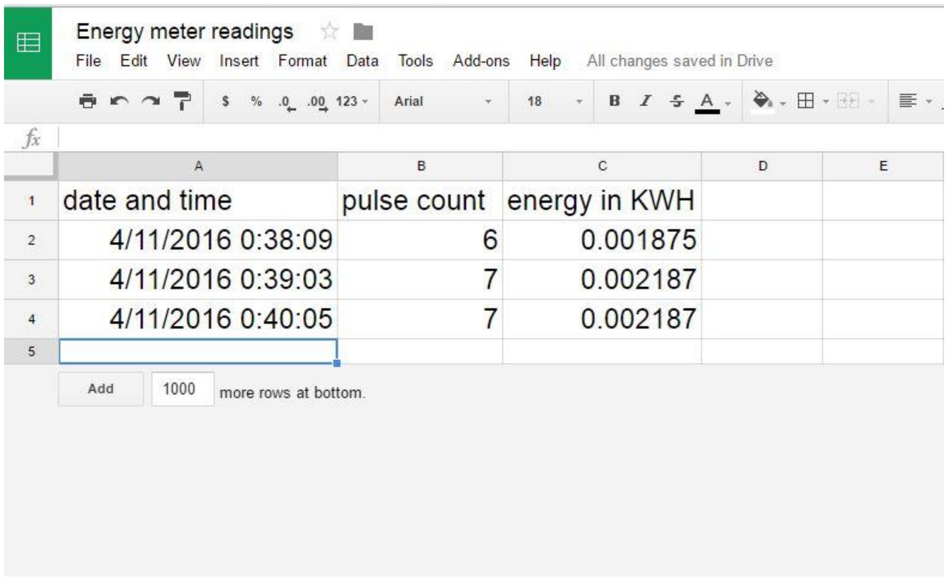
6. After executing the above command, the following result is observed. Some warnings are observed which can be ignored. The output is as shown in the figure 6.3



```
pi@raspberrypi ~ $ sudo python projtstgsm.py
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py:31
5: SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name
Indication) extension to TLS is not available on this platform. This may cause t
he server to present an incorrect TLS certificate, which can cause validation fa
ilures. For more information, see https://urllib3.readthedocs.org/en/latest/secu
rity.html#snimissingwarning.
  SNIMissingWarning
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py:12
0: InsecurePlatformWarning: A true SSLContext object is not available. This prev
ents urllib3 from configuring SSL appropriately and may cause certain SSL connec
tions to fail. For more information, see https://urllib3.readthedocs.org/en/late
st/security.html#insecureplatformwarning.
  InsecurePlatformWarning
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py:12
0: InsecurePlatformWarning: A true SSLContext object is not available. This prev
ents urllib3 from configuring SSL appropriately and may cause certain SSL connec
tions to fail. For more information, see https://urllib3.readthedocs.org/en/late
st/security.html#insecureplatformwarning.
  InsecurePlatformWarning
6
2016-04-11 00:38:07.613842
writing new row to Energy meter readings: 2016-04-11 00:38:08.593800 - 6.0 - 0.0
01875
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py:12
0: InsecurePlatformWarning: A true SSLContext object is not available. This prev
ents urllib3 from configuring SSL appropriately and may cause certain SSL connec
tions to fail. For more information, see https://urllib3.readthedocs.org/en/late
st/security.html#insecureplatformwarning.
```

Figure 6.3: Output on terminal window

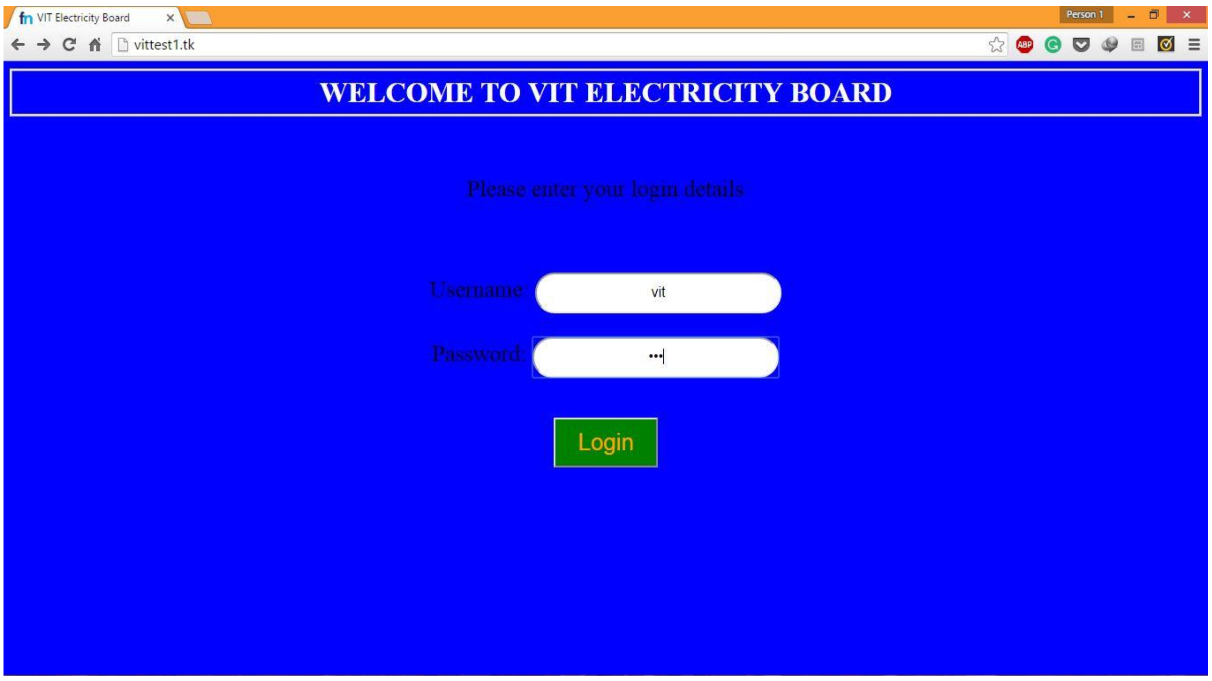
7. The data that is displayed on the webpage is as shown in the figure 6.4. The Google spreadsheet has been embedded into the webpage.


Energy meter readings
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive
Arial
18
Add
1000
more rows at bottom.

	A	B	C	D	E
1	date and time	pulse count	energy in KWH		
2	4/11/2016 0:38:09	6	0.001875		
3	4/11/2016 0:39:03	7	0.002187		
4	4/11/2016 0:40:05	7	0.002187		
5					

Figure 6.4: Output on Google spreadsheet

8. The website designed for users, named as vittest1.tk, is as shown in the figure 6.5 and figure 6.6. For our project, username and password has been taken as vit.



WELCOME TO VIT ELECTRICITY BOARD

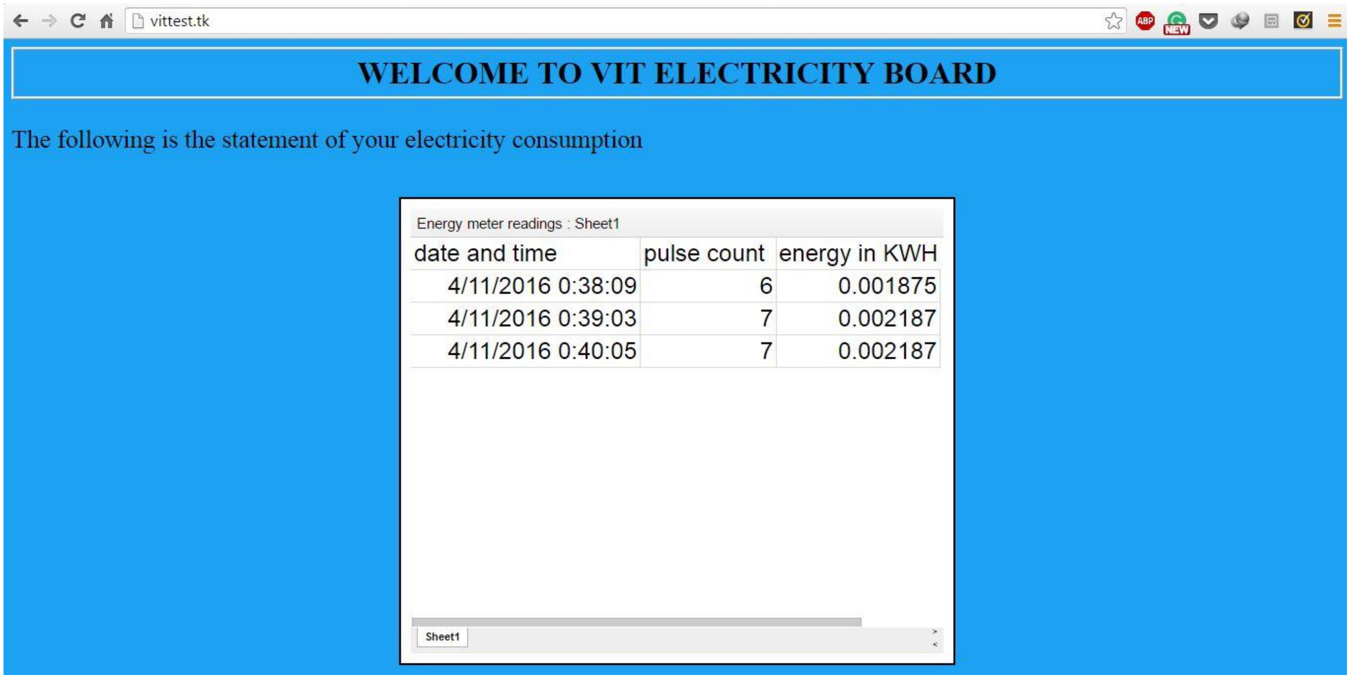
Please enter your login details

Username: vit

Password: •••

Login

Figure 6.5: Login page for our website – vittest1.tk


vittest.tk

WELCOME TO VIT ELECTRICITY BOARD

The following is the statement of your electricity consumption

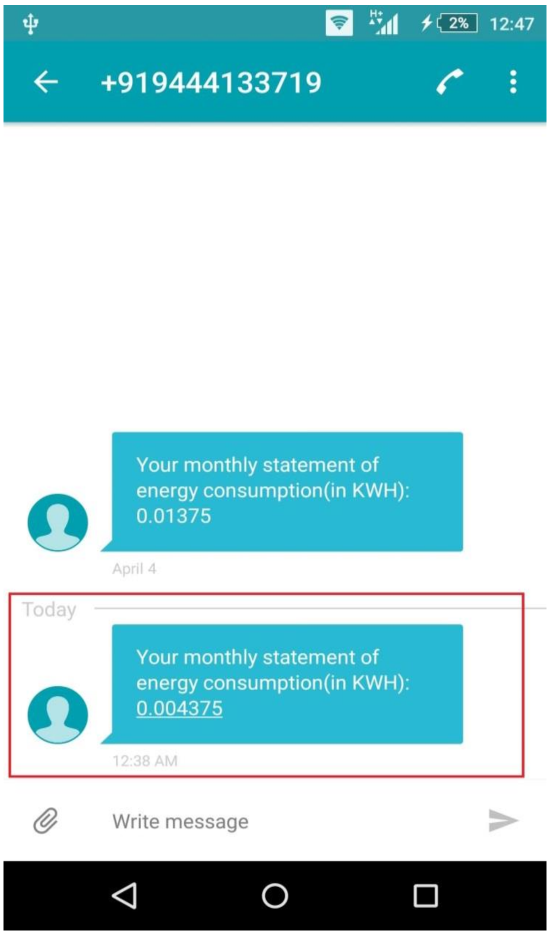
Energy meter readings : Sheet1

date and time	pulse count	energy in KWH
4/11/2016 0:38:09	6	0.001875
4/11/2016 0:39:03	7	0.002187
4/11/2016 0:40:05	7	0.002187

Sheet1

Figure 6.6: Statement of electricity consumption

9. An SMS was also received in the phone whose dial number has been mentioned in the program as shown in the figure 6.7.


+919444133719

Your monthly statement of energy consumption(in KWH): 0.01375

April 4

Today

Your monthly statement of energy consumption(in KWH): 0.004375

12:38 AM

Write message

Figure 6.7: SMS received on mobile phone

10. The figure 6.8 and figure 6.9 shows the output in Android App.

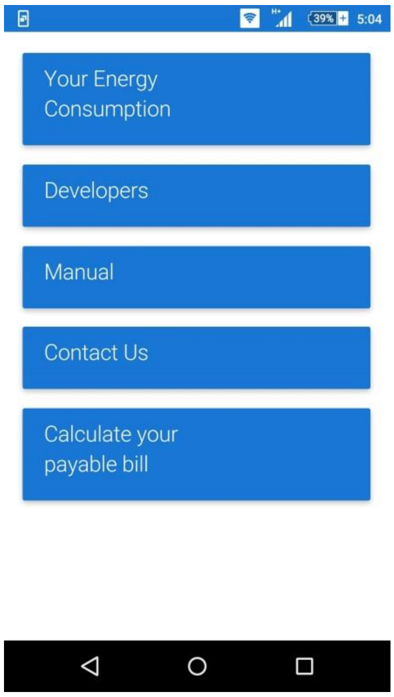
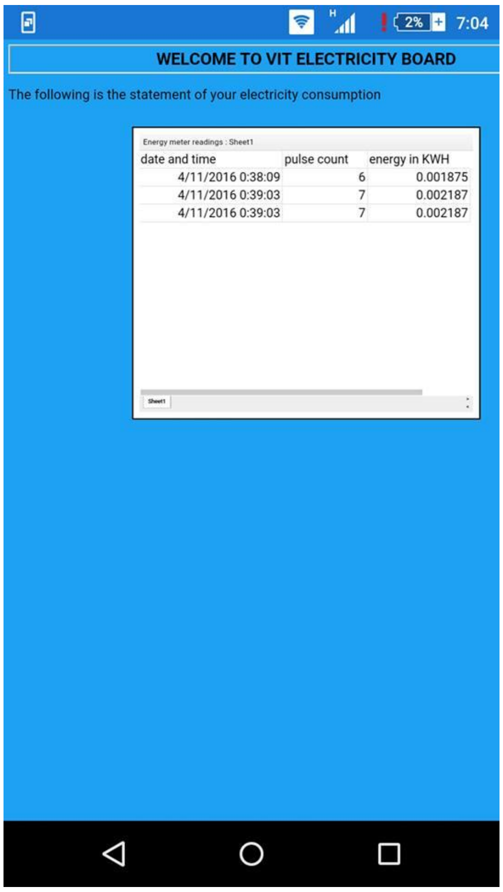


Figure 6.8: Home page of Android Application


WELCOME TO VIT ELECTRICITY BOARD

The following is the statement of your electricity consumption

Energy meter readings : Sheet1

date and time	pulse count	energy in KWH
4/11/2016 0:38:09	6	0.001875
4/11/2016 0:39:03	7	0.002187
4/11/2016 0:39:03	7	0.002187

Sheet1

Figure 6.9: Result on Android Application

7. COST ANALYSIS

While designing our project one of our objectives was to develop a system with as low cost as possible.

Table 7.1: Cost Analysis of the project

Serial number	Component	Price
1	RPi	Rs 2900
2	Electric Meter	Rs 400
3	LDR Module	Rs 200
4	Wi-Fi module	Rs 150
5	GSM module	Rs 950
	Total Cost	Rs 4600

8. SUMMARY

In this project we have designed a cheap and easy to install Automatic Metering System which can obtain the usage data using a LDR module and a RPi to transmit the acquired data. This project helps in eliminating human intervention for taking monthly readings, thereby reducing the costs associated with manual labor which is the chief advantage of our project.
The primary advantage is that this model can be installed in all household and industrial applications with no need of replacement of existing meter system as all the modern energy meters have LED indicator in them.

To summarize, we have calculated the pulse count using LDR module and converted the pulse count to energy consumed value. This value is exported periodically to Google spreadsheet which is embedded on a webpage for users to see. Additionally, they receive a SMS through the GSM module and they can view the results on an Android App as well.
In our project, we have obtained the results such that they are displayed in a website, Android App and SMS, keeping in mind the convenience of the user. The project can be used directly in real life scenario and it can be extended to display energy consumption bill amount in various forms by following the electricity bill calculations which change from one place to another.

REFERENCES

A. C. D. Bonganay, J. C. Magno, A. G. Marcellana, J. M. E. Morante and N. G. Perez(2014), "Automated electric meter reading and monitoring system using ZigBee-integrated RPi single board computer via Modbus," Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on, Bhopal, pp. 1-6

S. Shahidi, M. A. Gaffar and K. M. Salim(2013), "Design and implementation of digital energy meter with data sending capability using GSM network," Advances in Electrical Engineering (ICAEE), 2013 International Conference on, Dhaka, pp. 203-206

L. Quan-Xi and L. Gang(2010), "Design of remote automatic meter reading system based on ZigBee and GPRS" in Third International Symposium on Computer Science and Computational Technology, P.R China, pp. 186-189.

T. Ahmed et al(2010), "Automatic electric meter reading system: A cost-feasible alternative approach in meter reading for Bangladesh perspective using low-cost digital wattmeter and WiMax technology", M.S. thesis, Dept. Comp. Science, American International University, Dhaka, Bangladesh.

T. Maity and P. S. Das(2011), "A novel three phase energy meter model with wireless data reading and online billing solution," Computers & Informatics (ISCI), IEEE Symposium on, Kuala Lumpur, 2011, pp. 74-77.

Gokhan Kurt, "Vehicle Pi", RPi Android Projects, Packt Publishing Ltd. , 2015, pp.116-125

Video lectures on RPi Interfacing and Internet of Things offered by University of California, Irvine available at coursera.org

HTML and CSS tutorials available at w3schools.org

APScheduler documentation available at https://apscheduler.readthedocs.org/en/latest/#

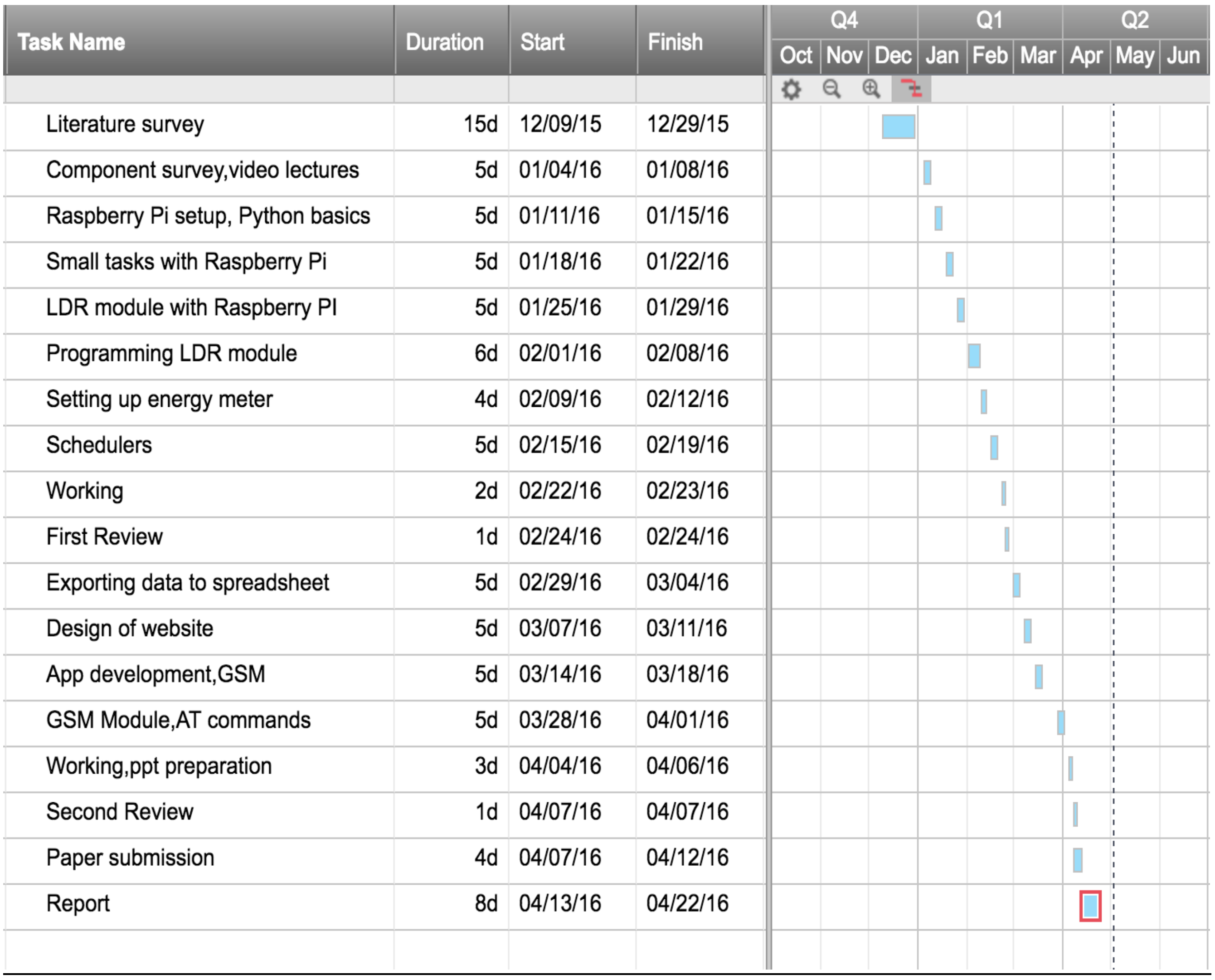
Using Google Spreadsheets available at
https://developers.google.com/identity/protocols/OAuth2#basicsteps

Video Tutorial for hosting a website in Google drive at
https://www.youtube.com/watch?v=oAx2sv9OyhE

Tutorial for setting up RPi at https://learn.sparkfun.com/tutorials/setting-up-raspbian-and-doom

APPENDIX A

(Gantt Chart)



Task Name	Duration	Start	Finish
Literature survey	15d	12/09/15	12/29/15
Component survey,video lectures	5d	01/04/16	01/08/16
Raspberry Pi setup, Python basics	5d	01/11/16	01/15/16
Small tasks with Raspberry Pi	5d	01/18/16	01/22/16
LDR module with Raspberry PI	5d	01/25/16	01/29/16
Programming LDR module	6d	02/01/16	02/08/16
Setting up energy meter	4d	02/09/16	02/12/16
Schedulers	5d	02/15/16	02/19/16
Working	2d	02/22/16	02/23/16
First Review	1d	02/24/16	02/24/16
Exporting data to spreadsheet	5d	02/29/16	03/04/16
Design of website	5d	03/07/16	03/11/16
App development,GSM	5d	03/14/16	03/18/16
GSM Module,AT commands	5d	03/28/16	04/01/16
Working,ppt preparation	3d	04/04/16	04/06/16
Second Review	1d	04/07/16	04/07/16
Paper submission	4d	04/07/16	04/12/16
Report	8d	04/13/16	04/22/16

Curriculum vitae

Name	:	Pendyala Arun Chandra
Father's name	:	Pendyala Ramaiah Chowdary
Date of Birth	:	16.10.1994
Nationality	:	Indian
Sex	:	Male
Company placed	:	Cognizant
Permanent Address	:	Flat no. 302, NRR Mansions, Old CBI Down, Vidyanagar, Visakhapatnam - 632014
Phone Number	:	0891-2529657
Mobile	:	9597435156
Email Id	:	arunchandrapendyala@yahoo.com

CGPA: 9.43 /10

Examinations taken:

1. GRE: 319/340
2. TOEFL: 110/120

Curriculum vitae

Name	:	Golla Mohith Vamsi
Father’s name	:	Golla Thyagarajulu Naidu
Date of Birth	:	14.12.1995
Nationality	:	Indian
Sex	:	Male
Company placed	:	Unplaced
Permanent Address	:	24-110/1A, B.C. Naidu Colony, Kongarareddypalli, Chittoor, Andhra Pradesh
Mobile	:	8500870498
Email Id	:	gollamohithvamsi@gmail.com

CGPA: 9.09/10

Examinations taken:

1. GRE: 317/340
2. TOEFL: 110/120

Curriculum vitae

Name	:	Yamparala Sri Manoj
Father’s name	:	Yamparala Seshu Babu
Date of Birth	:	03.06.1994
Nationality	:	Indian
Sex	:	Male
Company placed	:	Wipro
Permanent Address	:	40-25-17/4, 301, Sri Ratna Residency Netaji Street, Patamata Lanka, Vijayawada, Andhra Pradesh, 520010
Mobile	:	9159868583
Email Id	:	ysmanoj@ymail.com

CGPA: 8.58/10

Examinations taken:

1. GRE: 307/340
2. TOEFL: 102/120