

Introduction

This document provides installation and configuration instructions for the node power consumption web application. The web app provides a convenient way to view charts of power statistics for any calendar period for which data points are stored in the database. The app uses a round-robin database (RRD) that can be configured for any depth, limited only by file system storage capacity. The power data can be accessed by any client with a web browser that is on the mesh network. The instructions given in this document apply to most Linux distributions. The author has successfully run this app on a Raspberry Pi Zero running the Raspbian operating system.

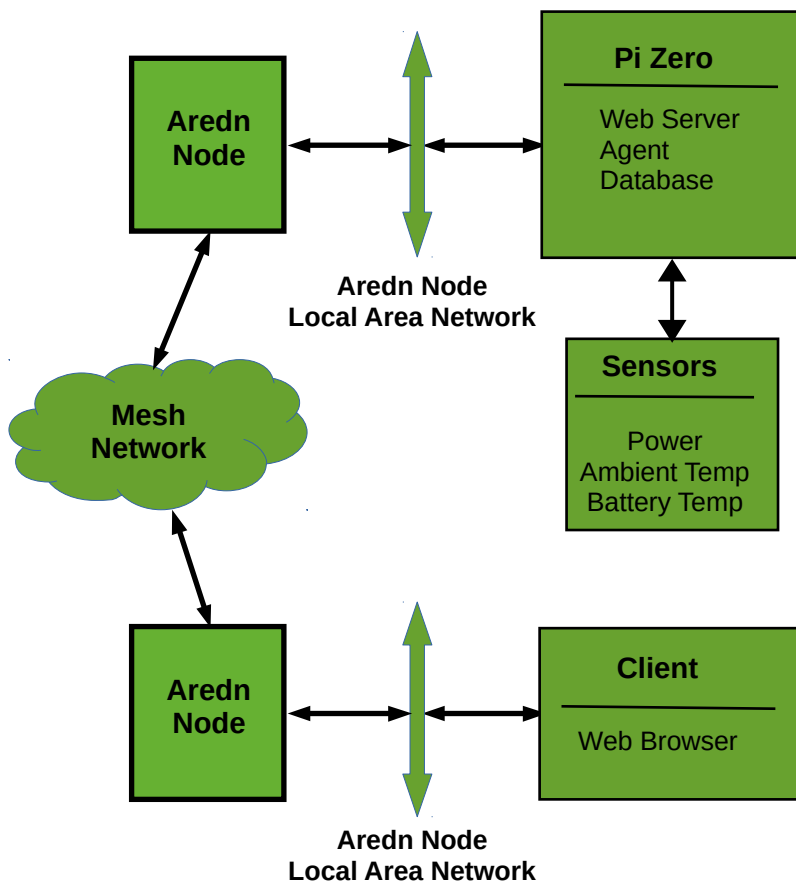


Figure 1. Overall conceptual view of node power sensor on AREDN mesh network.

AREDN Mesh System Overview

Figure 1 provides a conceptual overview of how the AREDN mesh provides network connectivity between client and node power sensors. Each node has two network interfaces: a wireless interface to other AREDN nodes, and a local area network interface to clients and servers. A node routes requests from clients on its LAN, via other nodes, to servers

Node Power Sensor – Software Installation

connected to the LAN's of other nodes. Referring to figure 1, a possible series of events goes as follows

1. The client sends, over the local node's LAN, a request for sensor data available on a server elsewhere on the mesh network.
2. The local node routes the request, via other nodes, to the remote node where the server is located.
3. The remote node routes the request to the appropriate server on its own LAN.
4. The server sends, over the remote node's LAN, the reply to the requesting client.
5. The remote node routes the reply back to the local node.
6. The local node routes, over its LAN, the reply back to the client.

Typically, the node's LAN is implemented by a Virtual Local Area Network (VLAN) capable switch, such as a Netgear GS105Ev2 switch. The node uses IEEE 802.Q VLAN identifiers to route traffic to devices on its own LAN, as well as routing external requests to an external network connect to a "WAN" port on the VLAN switch. An external request can be a request to any service on a host not on the LAN or somewhere else on the mesh network.

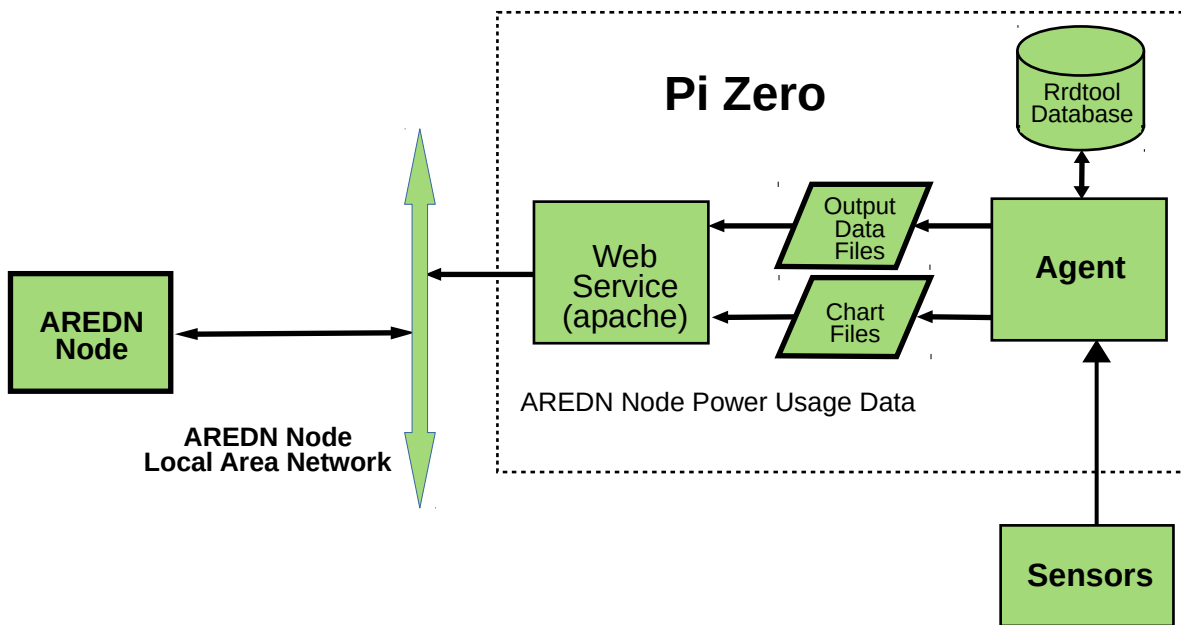


Figure 2. Block diagram of server software components showing data i/o, agent process, web service, and database elements.

Server Software

Referring to figure 2, the Pi Zero software consists essentially of two components: a web server component and an agent component. The web server component includes HTML documents containing embedded Javascript. The agent, a Python script, requests data from

the sensors, manages data conversion and re-formatting, updates a database, and generates stock charts. Events flow in the following manner.

The agent periodically sends a request for data to the sensors. The agent then converts specific data items to other formats where required. Selected data items get written to a round-robin database for permanent storage. Besides these functions, the agent manages generation of a stock set of charts for display in HTML documents. After formatting the node power consumption data, the agent writes the data to the Output Data File for use by HTML documents. When a client browser requests the HTML document, Javascript embedded in the document reads the output data file and displays this data in an HTML document. The agent, as mentioned earlier, generates graphic charts for display in HTML documents. The graphic charts are stored as image files which Javascript in the HTML documents can load and display in the web page.

Installing the Server Software

Important notes:

1. The NodePower application software should be installed on a Linux host which meets the following requirements:
 - The server software should be installed on a recent Linux distribution such as Debian, Ubuntu, or Raspbian. (The author has successfully developed and installed the software on a Raspberry Pi running the Raspbian operating system.)
 - Apache should be installed and configured to allow serving HTML documents from the user's public_html folder.
 - Rrdtool should be installed.
 - The Python I2C smbus library should be installed.
 - Python 2.7 usually comes pre-installed in virtually all Linux distributions. Type "python" at a command line prompt to verify Python has been installed.
2. See the Appendix for detailed notes on how to set up a Raspberry Pi server, and how to install and configure the components in note 1 above.

Software Inventory

The following software items in the install zip file need to be installed on the server

HTML folder:

- index.html
- power.html
- static/chalk.jpg

bin folder:

- createPowerRrd.py

Node Power Sensor – Software Installation

- npwAgent.py
- npwstart
- npwstop

Getting the NodePower Application Software

From the github repository:

1. On a computer connected to the Internet download the zip file from
<https://github.com/fractalxaos/ham/archive/master.zip>
2. From the downloaded file **ham-master.zip**, extract the **nodepower** folder to the desktop.

From the Willamette Valley AREDN mesh network:

1. On a computer connected mesh network download **nodepower.zip** from
<http://ka7jlo-raspi1.local.mesh/file-manager/files/KA7JLO/Apps/nodepower>
2. Save the zip file to the desktop or downloads folder.
3. From the downloaded zip file, extract all folders to the desktop.

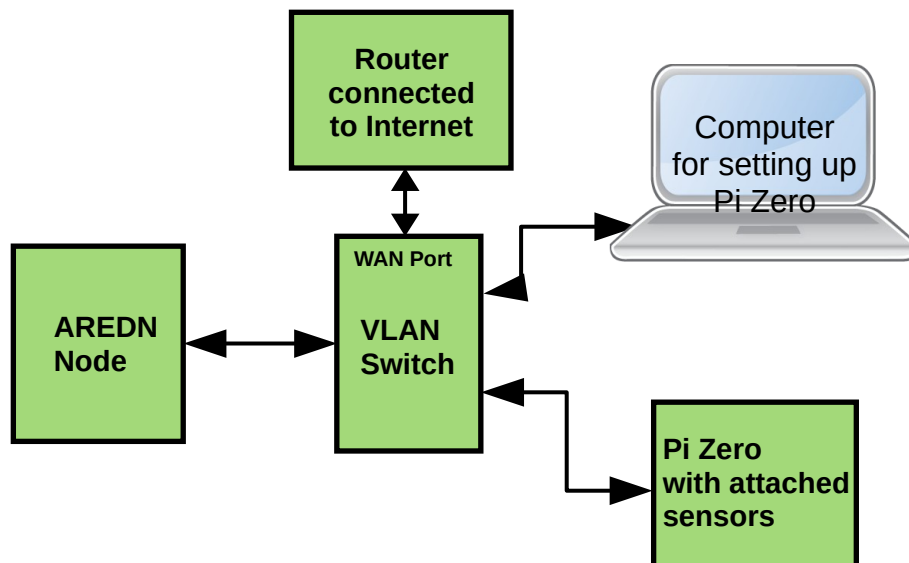


Figure 3. Pi Zero connected to VLAN switch in initial setup configuration for installing software on the Pi Zero. Note that the VLAN switch WAN port must be connected to a network that has Internet connectivity. An Internet connection is required in order to download the software required to set up the Pi Zero.

Installing on a Raspberry Pi

Figure 3 shows a block diagram of the Pi Zero connected to a VLAN switch. In the following notes the client computer used to set up the Raspberry Pi is also connected to the VLAN switch network. Note that the following installation procedure assumes that the document root for the NodePower HTML documents will be the user's `public_html` folder. Typically the full path name to this folder will be something like `/home/{user}/public_html`, where `{user}` is the name of the user account hosting node power. The following steps will assume the node power app is running under the `pi` user account. The `pi` account is the default account when Raspbian is first installed on a Raspberry Pi. If node power will running under another user account, then `pi` should be changed to that user name.

1. If necessary, turn on the I2C bus by running

```
sudo raspi-config
```

Select Interfaces. Select I2C and set to ON.

2. If it doesn't already exist, use `mkdir` to create a folder `public_html` in the `pi` user's home folder. The working folder should look like `/home/pi/public_html`.
3. In the `public_html` folder, use `mkdir` to create a folder `power` to contain the nodepower HTML files.
4. Move all the contents of the `html` subfolder in the `nodepower` folder to the `power` folder created in step 3. This includes the `static` subfolder and its contents.
5. The output data files (see figure 2) get overwritten frequently; similarly the chart files get overwritten frequently. On SD card systems, such as a Raspberry Pi, it is inadvisable to do frequent writes to any file system mounted on the SD card. To use the RAM based temporary file system (`tmpfs`) to store these files, continue with step 6. Otherwise, to write dynamic content to the disk drive, continue with step 11.
6. Assure that the server `/tmp` folder gets mounted to the temporary file system. This can be done by adding the following line to the `/etc/fstab` file

```
tmpfs /tmp tmpfs nodev,nosuid,size=10M 0 0
```

Reboot to make the above changes take effect.

7. Create a folder in the temporary file system for NodePower dynamic content, and modify the folder's ownership and permissions to allow the Apache `www-data` user to

have access.

```
mkdir /tmp/nodepower  
chmod g+w /tmp/nodepower  
sudo chown :www-data /tmp/nodepower
```

8. The commands in step 7 may be placed in a startup shell script and run at boot up time by launching the startup script with the **su** command from **/etc/rc.local**. For example, place the above commands in a script **/home/pi/bin/startup.sh** and place the following line in the **/etc/rc.local** file.

```
(su - pi -c "~/bin/startup.sh")&
```

Whenever the host boots up, the **startup.sh** script will run the commands in step 7. Be sure that you grant **startup.sh** execute permissions by running

```
chmod u+x ~/bin/startup.sh
```

9. In the **power** folder created in step 3 Create a symbolic link to **/tmp/nodepower** in the temporary file system by running

```
ln -s /tmp/nodepower dynamic
```

10. The web service cannot freely access files and folders outside of the **public_html** document root folder. To enable Apache to follow symbolic links to the **/tmp/nodepower** folder, back up and then modify the file **/lib/systemd/system/apache2.service**. Below the line **#PrivateTmp=true** insert the line **PrivateTmp=false**. Note that it is a good idea to use comments to indicate where and why a change has been made to a configuration file. For example,

```
#PrivateTmp=true  
# changed 2021-06-01 to enable apache to follow  
# symlinks to the /tmp folder in tmpfs  
PrivateTmp=false
```

Reload system deamons by running

```
sudo systemctl daemon-reload
```

Restart Apache by running

```
sudo systemctl restart apache2
```

11. If it does not already exist, use **mkdir** to create a folder named **bin** in the **pi** user's home folder. For example, the full path name should look like **/home/pi/bin**.
12. Move all the contents of the **bin** subfolder in the **nodepower** folder to the **bin** folder created in step 11. In most Linux installations the user's bash profile will automatically add the user's bin folder to the command search path. If such is the case, then the agent can be started up by simply typing **npwAgent.py** followed by ENTER. For example, typing **npwAgent.py -v** will launch the agent in verbose debug mode.
13. In the user home folder, create a folder named **database**. The full path to this folder should look like **/home/pi/database**. In the **bin** folder run the python script **createPowerRrd.py**. Running this script creates an empty round robin database file where the agent will periodically store the power data. This script should be run once and then kept in a secure place. Running it accidentally at some future date will result in *total loss of all previously stored data*.
14. In the user's home folder create a folder named **log**. This is where the agent will keep its error logs.
15. For convenience two scripts have been provided to make it easy to turn the agent on and off. The **npwstart** script starts up the agent and causes all diagnostic output and error messages to be written to a log file in the log folder. The **npwstop** script stops the agent from running. Start the NodePower agent by running **npwstart**. Alternatively the **npwstart** command can be placed in the **startup.sh** script mentioned in step 8 to automatically start the agent when the host boots up.

This completes installation of the server software.

References and Resources

The following resources describe how to configure a Raspberry Pi to be a server

- Using an SSH key pair:
<https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>
- Reducing wear on the SD card:
<https://www.zdnet.com/article/raspberry-pi-extending-the-life-of-the-sd-card/>
- Installing a web server:
<https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>
- Linux system administration and devops
<https://www.guru99.com/unix-linux-tutorial.html>

Node Power Sensor – Software Installation

<http://devopsbootcamp.osuosl.org/>
<http://lug.oregonstate.edu>

The following tutorials are useful for more in depth understanding of the server software:

- Javascript <http://www.w3schools.com/js/default.asp>
- PHP <http://www.w3schools.com/php/default.asp>
- HTML <http://www.w3schools.com/html/default.asp>
- Rrdtool <http://oss.oetiker.ch/rrdtool/>
- Python <http://greenteapress.com/thinkpython/thinkpython.html>

Appendix

The following steps describe how to build a web server on a Raspberry Pi. The steps below configure the Raspberry Pi to serve web pages from the user's document root folder. When user html folders are enabled, Apache looks for the files in the user's **public_html** folder. For example, a browser request **http://raspi.local/~pi/myDocument.html** would cause the host **raspi.local** to look for **myDocument.html** in the folder **/home/pi/public_html**. With the setup in the steps below, documents can also be served virtually, as if from the host's **/var/www** folder. For example, **http://raspi.local/myDocument.html** would also cause Apache to look for **myDocument.html** in the folder **/home/pi/public_html**.

1. If upgrading from a previous Raspbian version, first backup the user home folder by running

```
zip -r pi_bak.zip /home/pi/*
```

Copy the **pi_bak.zip** file to a thumb drive or some other external storage media.

2. Copy the new Raspbian OS disk image to the SD card. Follow raspbian instructions for copying the disk image to the SD card.
3. After copying the disk image to the SD card, mount the SD card **boot** partition on the computer you are using to set up the Raspberry Pi. Navigate to the root of the **boot** partition and create an empty file named **ssh**. This will enable setting up the Raspberry Pi without needing to attach a keyboard, mouse, and monitor.
4. If you are connecting to the Pi via wifi, also in the root of the **boot** partition create a file named **wpa_supplicant.conf** and add the following lines. If you are connecting via wired Ethernet skip this step.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US
network={
  ssid="your_wifi_network_name"
  psk="your_wifi_network_password"
  key_mgmt=WPA-PSK
}
```

5. Unmount and remove the SD card.

Node Power Sensor – Software Installation

6. Insert the SD card into the Pi and connect the Pi to its power source. The Pi will boot up.
7. Once the Pi has booted, on your computer open a terminal window and run the command

```
ssh pi@raspberrypi.local
```

The default password is **raspberry**.

8. Run the command

```
sudo raspi-config
```

and change the following items

```
hostname {your host name}  
password {your password}  
region US  
timezone {your timezone}  
keyboard US  
I2C Bus ON
```

9. Reboot the Pi by running

```
sudo reboot
```

10. Use ssh to login as user pi with the password set in step 8

```
ssh pi@{your host name}.local
```

11. Update the package database and install vim

```
sudo apt-get update  
sudo apt-get install vim
```

vim is an editor that makes it easy to make changes in configuration files. Alternatively you can use **nano** or some other terminal based editor of your choice.

12. [Optional] On the client computer, create an ssh key pair. In the Raspberry Pi user **pi** home folder, create a folder named **.ssh**. In the **.ssh** folder create a file named **authorized_keys**, and copy the public key to it. For more information on how to create and use key pairs see

<https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent> .

13. Note that all procedures below that involve editing a system file will have to be done acting as superuser. For example, to edit the file in step 14 run the command

```
sudo vi bootconfig.txt
```

14. If they will not be used, disable WiFi and Bluetooth. Backup and then modify **/boot/config.txt** by adding the following two lines to the end of the file:

```
dtoverlay=pi3-disable-wifi  
dtoverlay=pi3-disable-bt
```

and run the once off command:

```
sudo systemctl disable hciuart
```

15. [Optional] For added security turn off password authentication. You must have set up ssh keys as discribed in step 12. *Do not do this step unless you are familiar with ssh keys and how they work.* You will not be able to login to the Pi with a password. The above enhances security when using secure shell (ssh) to access the Raspberry Pi. Backup and then modify **/etc/ssh/sshd_config** as follows

```
#PasswordAuthentication yes  
PasswordAuthentication no
```

16. Setup the temporary file system (tmpfs) by backing up and then modifying **/etc/fstab**. Add the following lines to the bottom of the file.

```
# uncomment if needed for web appstial logs in ram to reduce  
# stress on the SD card due to frequent writes.  
tmpfs /tmp tmpfs nodev,nosuid,size=10m 0 0  
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=10m 0 0  
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=10m 0 0  
tmpfs /var/spool/mqueue tmpfs defaults,noatime,  
nosuid,mode=0700,gid=12,size=10m 0 0
```

Highly recommended, this step configures the Raspberry Pi to store all log files and temporary files in RAM. Unless configured to use an external hard drive, the Raspberry Pi mounts the root file system to the SD card. Log files and temporary files are frequently written to the file system, resulting in wear on the SD card. Storing these files in RAM saves the SD card from such wear. Note that all log files and temporary files are lost upon reboot or power down.

17. Reboot the Raspberry Pi.

18. Use ssh to login as user pi with the password set in step 8

```
ssh pi@{your host name}.local
```

19. [Optional] Run all software updates

```
sudo apt-get upgrade  
sudo reboot
```

20. Backup **/etc/rc.local** and then add the user start up script. For example add the following line to **/etc/rc.local**

```
(su - pi -c "bin/startup.sh")&
```

21. Install rrdtool

```
sudo apt-get install rrdtool
```

rrdtool maintains round-robin databases and generates charts for display in web pages. **rrdtool** can run on the same host as **Maria** (mySql). However they are not interoperable with each other.

22. Install LAMP. This is the standard Linux web server “stack”. For more information about installing LAMP see <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md> . Run the following commands

```
Apache  
=====  
sudo apt-get install apache2 -y  
sudo a2enmod rewrite  
sudo systemctl restart apache2
```

```
PHP  
===  
sudo apt-get install php libapache2-mod-php -y  
sudo systemctl restart apache2
```

23. Backup and then modify **/etc/apache2/mods-available/userdir.conf** to allow user **.htaccess** files. For example,

```
# changed 2020-01-15 by JLO to allow user .htaccess file  
#AllowOverride FileInfo AuthConfig Limit Indexes  
AllowOverride All
```

This enables apache to use the .htaccess file in the user's document root folder.

24. Enable user directories in Apache by running

```
a2enmod userdir
```

25. Enable php in Apache by running

```
a2enmod php7.3
```

26. Backup and then modify **/etc/apache2/mods-available/php7.3.conf** to allow PHP scripts to run in user directories by commenting the lines at bottom of file. For example,

```
# changed 2020-01-15 by JLO to enable user php scripts
#<IfModule mod_userdir.c>
# <Directory /home/*/public_html>
#     php_admin_flag engine Off
# </Directory>
#</IfModule>
```

27. Backup and then modify **/etc/apache2/sites-available/000-default.conf** to make the pi user **public_html** folder the web server document root. For example,

```
# changed 2020-01-15 by JLO to make user
# pi the html document root
#DocumentRoot /var/www/html
DocumentRoot /home/pi/public_html
```

This makes the pi user public_html folder the document root for the Raspberry Pi. For example a client request **http://raspi.local/myDocument.html** would cause Apache to look for the document in the folder **/home/pi/public_html**.

28. Backup and then modify **/etc/apache2/envvars** to create Apache logs in tmpfs. Add the following lines at the top of the file

```
# added 2020-01-15 by JLO to allow apache to use tmpfs
if [ ! -d /var/log/apache2 ]; then
    mkdir /var/log/apache2
fi
if [ ! -d /var/log/mysql ]; then
    mkdir /var/log/mysql
fi
```

Without these lines Apache will fail to start when the system reboots. When the system reboots the folders **/var/log/apache2** and **/var/log/mysql** will not exist if logs are using

the temporary file system in RAM. Apache will fail to start if these folders are not present. Adding the above lines causes Apache to create these folders when the system boots up.

29. Enable Apache to access files in the the tmpfs **/tmp** folder by backing up and then modifying **//lib/systemd/system/apache2.service** as follows:

```
# changed {date} by {name} to enable apache to follow
# symlinks to the /tmp folder in tmpfs
#PrivateTmp=true
PrivateTmp=false
```

This change enables Apache to follow symbolic links in the user document root (public_html) folder to folders and files in the temporary file system.

Reload system deamons

```
sudo systemctl daemon-reload
```

Restart Apache service

```
sudo systemctl restart apache2
```

This change enables Apache to follow symbolic links in the user document root (public_html) folder to folders and files in the temporary file system **/tmp**.

30. Install i2c smbus python library by running

```
sudo apt-get install python-smbus
```

31. Reboot the Raspberry Pi.

32. Copy the backup file pi_bak.zip from the thumb drive media or other external media to the /home/pi folder. Restore desired files and folders from backup archive by running

```
unzip pi_bak.zip
```

Use mv to move folders and files to their appropriate locations.

33. Test all the above modifications.