Power monitoring using the Raspberry Pi

Robin M Snyder
RobinSnyder.com
robin@robinsnyder.com
http://www.robinsnyder.com

Abstract

The Raspberry Pi is a credit card size low powered compute board with Ethernet connection, HDMI video output, audio, full Linux operating system run from an SD card, and more, all for $45. With cables, SD card, etc., the cost is about $70. Originally designed to help teach computer science principles to low income children and students, the Pi has taken on a life of its own, with many online resources and projects that cover most everything one would want to do with a small low monetary cost and low battery power computer. This paper/session will present the Raspberry Pi and provide ideas of what it can do and how one can get started using it. The author has used Pi's for Internet data acquisition/monitoring, cluster processing using more than one Pi, email notifications, and power monitoring. The particular focus example will be on UPS power monitoring and status using off the shelf Open Source software. The main software for the purpose of power monitoring is NUT, Network UPS Tools, whose goal is "to provide support for Power Devices, such as Uninterruptible Power Supplies, Power Distribution Units and Solar Controllers." Some background in the theory and practice of electrical power, surge protection, uninterruptible powers supplies, etc., will be provided.

Introduction

The Raspberry Pi is a credit card size low powered compute board with Ethernet connection, HDMI video output, audio, full Linux operating system run from an SD card, and more, all for $45. With cables, SD card, etc., the cost is about $70.

The Raspberry part comes from the name of a fruit not yet used so that the name could be trademarked. For example, Apple is already taken. The Pi part is the Greek letter pi as used in the ratio of circumference to diameter. Originally developed to promote computer science among lower income students, the Pi has taken on a life of its own with a community of users that have and continue to provide support a large number of areas.

By default, the Pi runs a version of Debian Linux called Raspian. In some cases, such as the Firefox web browser, technical/legal limitations/restrictions of the open source Firefox web browser have caused it to be renamed IceWeasel, but is otherwise the same.

One can run the Pi in GUI mode are in headless mode. That is, without a monitor/screen, being accessed via, say, SSH (Secure Shell Protocol). That is the primary way in which it will be used in this paper. There are many good intro books and/or online web sites dealing with getting started with the Raspberry Pi. This paper will only touch on some of the aspects related to the main topic of this paper - power monitoring and management using the Raspberry Pi.

Raspberry Pi alternatives

Here are some common alternatives to the Raspberry Pi.

Brick Pi

The Brick Pi is a KickStarter project that provides a board that can interface the Pi to Lego bricks and Lego sensor devices as found in the Lego MindStorms system.

Arduino

There are many add-on boards that can be used to interface the Raspberry Pi to Arduino boards.

The Arduino is a low cost open source hardware and software system designed to interface to the real would using an extensive collection of sensor (i.e., input) and actuator (i.e., output) devices. The Arduino, however, has a limited amount of memory and input/output capacity.

Thus, an Arduino combined with the Raspberry Pi can provide a useful system to interface sensors and actuators to the real world.

Raspberry Pi alternatives

There are many alternatives to the Raspberry Pi. Some of those used and/or investigated by the author include the following.

BeagleBone

The BeagleBone is faster, with more memory, no composite video (but with HDMI), and flash memory for the resident ARM Linux based system.

The author found it useful to use a SD card boot with Ubuntu rather than the suggested Linux OS.

Yun

The Yun is an Arduino system combined with a OpenWRT-based Linux system (as used in some routers). The Yun comes with a network connection and a wireless hotspot device.

3.3 Intel Galileo

The Intel Galileo is a more expensive and, in the authors opinion, not as well designed/realized system, that has an on-board Linux system and Arduino-like interface for connection to sensors and actuators.

Panda Board

The Panda Board is a dual ARM processor system, about $190, that has been used in a number of larger custom applications.

Power management

Every electronic device requires some type of power. Some devices require more power than other devices. Intel traditionally assumed that the processor would be always running. The ARM architecture, on the other hand, allows the process to almost stop completely - which helps a lot when a longer battery life is required. Intel's response to the demand for the competitor ARM's processors has been to develop an Atom (and Sandy Bridge, etc.) series of microprocessors to help with reducing power consumption and thereby increasing battery life.

Typical electrical power supply systems have fluctuations in the quality of power provided. Surges can increase the nominal voltage supplied. Sags can decrease the nominal voltage supplied. A brownout is a reduction in the voltage/current supplied. A blackout is a total decrease of voltage/current supplied. Spikes can pass though and induce currents in devices that can burn them out or otherwise cause the devices to malfunction. In the extreme case, such spikes are called EMP (Electromagnetic Pulses) that can be created by atomic/nuclear devices and can burn out (or cause to not function properly) sensitive electronic equipment such as microprocessors and associated memory, etc.

One solution to reasonable surges, sags, and/or spikes (i.e., not EMP) is to use a UPS (Uninterruptible Power Supply) which will help with sags, surges, and spikes, and can even be delivered by UPS (United Parcel Service)!

It is common to have computers, servers, etc., connected to UPS systems. But what if the power goes out for an extended period of time such that the connected servers, etc., will lose power. APC makes UPS systems that can use their PowerChute software to monitor and take actions if power is lost for an extended period of time. And even if power is not lost, one might want to know how the power systems have been functioning, the load on each, when they lose power, etc., and then shut down and restart the servers at an appropriate time.

The rest of this paper will look at how the Raspberry Pi can be used to monitor and manage UPS power in a network setting.

Security

The default user is user **`pi`** with default password **`raspberry`**. It is strongly suggested that the default password be changed. It is also suggested, if external access from the Internet is to be used, that the user **`pi`** have a very secure password and that another user be created to do the same things that would be done with the user **`pi`**.

Pi clusters

Multiprocessing using multiple Raspberry Pi boards, a Pi cluster, requires careful configuration to allow each Pi to be accessed while allowing each Pi in the cluster to be managed.

The Pi boots from an SD card. As one changes the configuration of the Raspian Linux, it is convenient to have a way to duplicate SD cards.

The hardware part of the copying can be done by a dedicated device. The author uses a $99 USB 2.0 flash drive copier, using USB 2.0 SD card adapters. Some SD cards will not be recognized but most are. One can copy one source SD card to two target SD cards simultaneously and then have them automatically verified by the copy device.

Each network device has a MAC (Media Access Control) address and an IP (Internet Protocol) address assigned by the local router. Using the static DHCP (Dynamic Host Control Protocol) of the router, each device (e.g., a Pi) can be assigned a specific IP address. For example, the author's Raspberry Pi's are assigned the IP addresses 192.168.100.161 for PX1, 192.168.100.162 for PX2, etc.

The software side of the copying is handled by a Bash/Python script combination that does the following using a hard-coded database of hardware MAC addresses and assigned IP addresses. If the Pi, as identified by the MAC address, does not match the desired IP address, the relevant files (e.g., the hosts file, hostname file, etc.) are modified and the message is displayed that the device needs to be rebooted, at which time, the Pi has been reset with all the desired configuration updates from the desired SD card and the software part has been appropriately modified.

External access

In order to do remote management of, for example, power monitoring, it is necessary to access the Pi from a remote location (i.e., from the Internet outside of the local network to which the Pi as connected). If the Pi is to be accessed from the Internet, it is suggested that another port, rather than standard SSH port of `22`, be used.

Provided the ISP (Internet Service Provider), such as ComCast, does not forbid it, the general method for accessing the Pi from the Internet is as follows.

The local setup is assumed to be as follows, using ComCast as an example.

- The Internet is provided via a Cable Modem using, for example, DOCS v3.
- A local router allows local sharing of the connection between machines.
- Assume the router has an IP address of 74.12.34.56, assigned by the ISP.
- The Pi is connected to the local router at IP address 192.168.100.161.

The general configuration is as follows.

- Assume the external SSH port is 2244.
- Set the port forwarding of the local router to take requests to IP 74.12.34.56 on port 2244 and redirect them to the local Pi at 192.168.100.161 on port 2244.
- Set up the Pi SSH Server to accept SSH requests on port 2244.
- Insure that local firewalls allow the desired traffic.

Power management

One use of a Raspberry Pi is the monitor and manage the power for the local network as provided by one or mare UPS (Uninterruptible Power Supply) systems that support the NUT (Network UPS Tools) collection of open source software tools.

The author uses APC (American Power Corporation) UPS systems, most of which support the NUT protocol for power monitoring. The NUT web site can be used to help determine if a given UPS is supported by NUT.

The setup of such monitoring can be a useful project that students in the computer science department could help develop for use by the IT (Information Technology) group at an academic institution.

Since the Pi is a complete Linux system running on the local network, email can be sent, SMS text messages can be sent, drives can be accessed, logs can be created, etc., just like any other computer/machine but with a small price (in case the device fails or is destroyed, for example, by a lightning strike) and small power usage (which can otherwise add up to a significant amount over time).

Power monitoring

"**:nut monitor ups device**"

An Internet search term for more information is "**Raspberry Pi NUT**". The author took some of the configuration and setup ideas from http://abakalidis.blogspot.com/2013/04/using-raspberry-pi-as-ups-server-with.html, which is one of the top results in a Google search of the topic.

One can check the NUT web site to see if a UPS is supported. Most APC UPS devices are supported and those are the only UPS devices used by the author.

The UPS is assumed to have a USB serial interface and not an older serial port interface. APC UPS systems that support NUT come with a special USB to serial cable for that particular UPS device.

Some Raspberry Pi Linux commands that are relevant to power monitoring are now covered. These are commands typed at the command line that can later be scripted to run automatically, if desired. The commands will serve as a gentile introduction to the Linux command line. Most commands here start with `sudo` which stands for "**Super User Do**" and for which there is no password to use this command on the Pi.

Note that the Linux way is to use text files for configuration so that only text editors are needed to configure, though important systems will have someone write a GUI to do the configuration. The alternative is the Windows way to make everything somewhat secret and proprietary and, in most respects, more difficult to change.

The USB utilities should be installed to get status on UPS devices.

```
sudo apt-get install usbutils
```

The USB utilities include the **lsusb** command. The command **lsusb** lists the known USB devices connected to the system. Here is part of a typical output.

```
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 005: ID 051d:0002 American Power Conversion Uninterruptible Pow-
er Supply
Bus 001 Device 004: ID 051d:0002 American Power Conversion Uninterruptible Pow-
er Supply
```

Each connected device has a bus (e.g., **001**), a device designation (e.g., **004**, **005**, etc.) and a 32-bit identifier (e.g., **051d:0002**, **051d:0002**) that identifies a specific type of device. That is, the identifier is like a UPC bar code that identifies a type of device and not an RFID tag that identifies a unique instance of a device.

To find the available/used USB devices, use the following command.

```
find /dev/bus/
```

The author had originally desired to connect more than one UPS device to a Pi. However, from the NUT web site:: *The driver ignores the "port" value in ups.conf. Unlike previous versions of this driver, it is now possible to control multiple UPS units simultaneously with this driver, provided they can be distinguished by setting some combination of the "vendor", "product", "serial", "vendorid", and "productid" options*. Since the author's UPS devices have the same information with respect to this limitation, the rest of this paper will use just one UPS device. Using one Pi per UPS device, however, would allow overcoming this restriction as a client machine (i.e., not the Pi connected to the UPS) could monitor the Pi machines that are each monitoring one UPS device.

```
lsusb -t

/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=dwc_otg/1p, 480M
      |__ Port 1: Dev 2, If 0, Class=hub, Driver=hub/5p, 480M
            |__ Port 1: Dev 3, If 0, Class=vend., Driver=smsc95xx, 480M
            |__ Port 2: Dev 6, If 0, Class=HID, Driver=usbhid, 12M
            |__ Port 3: Dev 5, If 0, Class=HID, Driver=usbfs, 12M
```

The NUT server

This section describes the NUT server setup, configuration, and usage.

Install the server

The following command can be used to install NUT, the NUT server, and the NUT client.

```
sudo apt-get -y install nut nut-client nut-server
```

Relevant files

Here are some relevant configuration files in the **/etc/ups** folder.

- **ups.conf** contains settings for UPS-specific drivers.

- **`upsd.conf`** contains settings for the main UPS daemon **`upsd`**.
- **`upsd.users`** contains user and access control for the UPS daemon **`upsd`**.
- **`upsmon.conf`** contains settings for the UPS monitoring daemon **`upsmon`**.
- **`upssched.conf`** contains settings for the **`upssched`** daemon.

Configure the server

Use the following command to edit the server configuration file **`/etc/nut/ups.conf`** using the nano editor.

```
sudo nano /etc/nut/ups.conf
```

Add the following at the end of this file. In this case, the author has two UPS devices to be added. The NUT web site has suggested drivers for known UPS devices.

```
[apc1200]
   driver = usbhid-ups
   port = auto
   desc = "APC Back UPS Pro 1200VA a"
```

A folder is needed for server operation. Here are some commands to do this.

```
sudo mkdir /var/run/nut
sudo chown root.nut /var/run/nut/
sudo chmod 770 /var/run/nut/
```

Start the server

The following command attempts to start the server and shows the resulting status. Any errors in the configuration file will be noted here.

```
sudo upsdrvctl start
```

Here is the output.

```
Network UPS Tools - UPS driver controller 2.6.4
Network UPS Tools - Generic HID driver 0.37 (2.6.4)
USB communication driver 0.32
Using subdriver: APC HID 0.95
Network UPS Tools - Generic HID driver 0.37 (2.6.4)
USB communication driver 0.32
Using subdriver: APC HID 0.95
```

Next some connections need to be made so that the client and server and UPS listen on the appropriate ports. Here is the command to edit the appropriate file.

```
sudo nano /etc/nut/upsd.conf
```

Add the following **`LISTEN`** directives, where **`192.168.100.160`** is the IP address of the Pi on the local network and the IP port being used is **`3493`**. Port **`127.0.0.1`** is the **`localhost`** port.

```
LISTEN 127.0.0.1 3493
LISTEN 192.168.100.160 3493
```

The **ifconfig** command can be used to determine the IP address of the Pi, although one will want to use static DHCP from the router so that the Pi always has the same local IP address. Note that the Windows command to see the IP address is **ipconfig** (starts with "**ip**") while the Linux command is **ifconfig** (starts with "**if**").

Edit the **upsd.users** file with the following command.

```
sudo nano /etc/nut/upsd.users
```

Here is the header of the file.

```
# Network UPS Tools: Example upsd.users
#
# This file sets the permissions for upsd - the UPS network daemon.
# Users are defined here, are given passwords, and their privileges are
# controlled here too. Since this file will contain passwords, keep it
# secure, with only enough permissions for upsd to read it.
```

The following should be already set but the users and passwords should be changed as desired for security, etc.

```
[admin]
   password = apcupc1
   actions = SET
   instcmds = ALL

[testuser]
   password = pass
   instcmds = test.battery.start
   instcmds = test.battery.stop

[upsmon_local]
   password = local_pass
   upsmon master

[upsmon_remote]
   password = remote_pass
   upsmon slave
```

Here, **local_pass** and **remote_pass** are used as the passwords, but one should use a secure password for such purposes. If only local access is desired, do not include the remote user.

Add commands such as the following in the **MONITOR** section.

```
MONITOR apc1200@PX0 upsmon_local local_pass master
```

In this case, **PX0** is the domain name set in the **hosts** file and assigned to **localhost** - **237.0.0.1**.

```
#MODE=none
MODE=netserver
```

The following command can be used to start the NUT server.

```
sudo service nut-server start
```

Here is a typical output when the server failed to start.

```
[ ok ] Starting NUT - power devices information server and drivers: driver(s).
(upsd failed).
```

Here is a typical output when the server starts successfully.

```
[ ok ] Starting NUT - power devices information server and drivers: driver(s).
upsd.
```

The following command can be used to insure that the services just configured will start when the system is started.

```
ps -ef | grep ups
```

The **ps** command lists all of the running processes while **grep** filters out all but those that contain the text "**ups**".

```
nut 2567 1 0 20:35 ? 00:00:00 /lib/nut/usbhid-ups -a apc1200a
nut 2569 1 0 20:35 ? 00:00:00 /lib/nut/usbhid-ups -a apc1200b
nut 2572 1 0 20:35 ? 00:00:00 /sbin/upsd
pi 2592 2412 0 20:37 pts/0 00:00:00 grep ups
```

NUT statistics

Once the server is running, statistics can be obtained using the **upsc** command as follows.

```
upsc apc1200@192.168.100.160
```

A typical output is as follows.

```
battery.charge: 100
battery.charge.low: 10
battery.charge.warning: 50
battery.date: 2001/09/25
battery.mfr.date: 2012/04/29
battery.runtime: 17676
battery.runtime.low: 120
battery.type: PbAc
battery.voltage: 27.2
battery.voltage.nominal: 24.0
device.mfr: American Power Conversion
device.model: Back-UPS RS 1500G
device.serial: 3B1217X31574
device.type: ups
driver.name: usbhid-ups
driver.parameter.pollfreq: 30
driver.parameter.pollinterval: 2
driver.parameter.port: auto
driver.version: 2.6.4
driver.version.data: APC HID 0.95
driver.version.internal: 0.37
input.sensitivity: medium
input.transfer.high: 147
input.transfer.low: 88
input.voltage: 121.0
input.voltage.nominal: 120
ups.beeper.status: enabled
ups.delay.shutdown: 20
ups.firmware: 865.L3 .D
ups.firmware.aux: L3
ups.load: 8
ups.mfr: American Power Conversion
ups.mfr.date: 2012/04/29
ups.model: Back-UPS RS 1500G
ups.productid: 0002
ups.realpower.nominal: 865
ups.serial: 3B1217X31574
ups.status: OL
```

```
    ups.test.result: No test initiated
    ups.timer.reboot: 0
    ups.timer.shutdown: -
```

But the following similar commands can also be issued from any computer on the network, not just the Raspberry Pi's.

```
    upsc apc1200b@192.168.100.160
    upsc apc1200b@192.168.100.162
    upsc apc1200b@192.168.100.163
    upsc apc1200b@192.168.100.164
```

In this case, the following Raspberry Pi's were set up on the network.

- `PX0`, at `192.168.100.160`
- `PX2`, at `192.168.100.162`
- `PX3`, at `192.168.100.163`
- `PX4`, at `192.168.100.164`

Note: The author was/is using `PX1`, at `192.168.100.161`, for other purposes (which is why the pattern is not uniform and contiguous)

Since all of the above information can now be obtained, parsed, processed, saved, etc., for each UPS on the network, or even from outside the network if the routers and port forwarding and firewalls are appropriately set, a larger power monitoring and management system can be created.

Obviously, some additional programming is needed to make use of this data (e.g, parse and store for later use), but the data is now available.

Starting at boot

To start NUT at boot time, edit the NUT configuration file using the nano editor with the following command.

```
    sudo nano /etc/default/nut
```

Add the following to this file, which is normally empty after a new install.

```
    # /etc/default/nut
    START_UPSD=yes
    START_UPSMON=yes
```

NUT client

The Raspberry Pi is being used as the NUT server. A client can be installed on some other machine (e.g., Linux client, Windows client, etc.) which can then access the Pi to get information about the power state. This access can be local or remote (provided the routers, firewalls, etc., are properly configured).

Additional configuration involves making decisions about what to do when the power fails.

There are GUI programs available to monitor and manage power. These programs automate and add a GUI interface to the commands presented above.

Ideas

Some ideas of usage are as follows.

- Monitor and report power usage.
- Determine if the load is too much for a given UPS.
- Determine when the battery should be replaced, or warn of the battery getting old.
- Send an email, or SMS message, when appropriate conditions are encountered.
- Send messages to have a server, which is monitoring the conditions, shut down, and then, when power is returned, start again. (This is non-trivial and requires careful planning and setup for everything to work).

Summary

This paper/session has discussed and/or demonstrated the Raspberry Pi with an emphasis on power monitoring and management. Some operational details were covered and some future ideas presented.

References

McGrath, M. (2013). Raspberry Pi in Easy Steps. Computer Step.

Monk, S. (2012). Programming the Raspberry Pi: Getting Started with Python. Mcgraw-hill.

Norris, D. (2013). Raspberry Pi Projects for the Evil Genius. McGraw-Hill Education.

Philbin, C.A. (2014). Adventures In Raspberry Pi. Wiley.

Richardson, M. and Wallace, S. (2012). Getting Started with Raspberry Pi. O'Reilly Media, Inc.

Upton, E. and Halfacree, G. (2012). Raspberry Pi User Guide. Wiley.