Introduction:

The Broadband Ham Net community has had a need for a stand alone time server for their meshes. This document will attempt to fill in this need.

Some caveats to consider before embark on a building project. While information is available on the Internet, that information is somewhat confusing/disorganized. Furthermore, the information is old and programs/ scripts mentioned are no longer available. And this document will get out of date as well.

I will try to keep the information in this document as generic as possible to hopefully extend the life of the information.

73 de N4FWD

Reference information: http://www.satsignal.eu/ntp/Raspberry-Pi-NTP.html

I want to properly thank David Taylor for taking the time to document all the information which he discovered while exploring the use of a Raspberry Pi as a time server. His articles can be read at the URL posted above.

I would also like to thank David Taylor for giving me permission to reference his work as well as the assistance which he rendered while I searched for answers about stand alone operation.

Table of Contents

List of Materials

Raspberry Pi computer – as of the writing of this document, the Raspberry Pi Foundation has released the Version 2 (with quad core CPU and 1 Gigabyte of RAM). This document used a Version 1.2 (with single core CPU and 512Megabytes of RAM).

GPS module – the market place has two versions of GPS module, a USB based version and a Serial port version. If you are going to build a serious time server, you need to avoid the USB based GPS module. The reason has to do with access to the timing pulse which is available from GPS. Any GPS module will give you position information, but without the timing pulse, you have no idea if the time information is valid.

The other item to look for when shopping for a GPS module is what voltage will the module require. To avoid voltage issues with the GPIO bus of the Raspberry Pi, shop for a module which will run on 3.3 Volts.

Interface kit – Unless the GPS module comes pre-wired for the Raspberry Pi GPIO connector, you will need an interface kit of some sort to assist you with interconnecting the GPS module to the GPIO connector. Try to avoid point to point wiring. Electronics do fail and it would be more advantageous to unplug the bad GPS module and plug in a new one than to dig out the soldering iron.

Wiring Guide

GPS Gnd → GPIO Gnd	Power
GPS Vdd → GPIO 3.3 Volts	Power
GPS T_P → GPIO GP18 / PWM	Timing Pulse
GPS Txd → GPIO GP15 / Rxd	Data receive

Use “Best Practices” when assembling your unit. Keep in mind that the GPS antenna must have a clear view of the sky.

Software

The Operating System chosen for this project is Raspbian Linux. You can use what ever Linux Distribution which you feel comfortable with, however, you will be on your own as far as software commands go.

Pre-Existing Raspbian OS

In a terminal window, update your Raspbian Linux with:

1. sudo apt-get update
2. sudo apt-get dist-upgrade
3. sudo rpi-update

New Raspbian OS

Start at the official Raspberry Pi website https://www.raspberrypi.org/downloads/ and follow the instructions for downloading, creating and configuring the Raspbian Linux.

Serial Port Check

You will need to check the boot options to ensure that the GPIO serial port is not in use. In a terminal window:

- sudo cat /boot/cmdline.txt

and check 'console=' It should read **console=tty1** and not **console=ttyAMA0**

If you need to fix this:

- sudo nano /boot/cmdline.txt

Next check:

- sudo cat /boot/config.txt

Again, there should be no references to **ttyAMA0**.

GPS Modifications

Now you will need to add some information to the /boot/config.txt for the boot time settings. In a terminal window:

- sudo nano /boot/config.txt

Now add the following lines to the end of the file and save:

- force_turbo=1
- dtoverlay=pps-gpio,gpiopin=18
- init_uart_baud=9600
- arm_freq=800

During the wiring phase, you connected the timing pulse signal to GPIO18. You are now telling the Raspbian Linux in software where to look for the timing signal. Furthermore, you are also telling Raspbian Linux what baud rate to set the hardware to. If the GPS module uses a 4800 baud rate, then modify 9600 to 4800. The **force_turbo** disables the dynamic clocking which can interfere with the time accuracy.

Software Packages

If you have not rebooted your Raspberry Pi yet, in a terminal window:

- sudo reboot

and log in again.

Now it is time to add in the software packages which will complete the project. The default ntp package which ships with Raspbian Linux does not support use of the timing pulse. As such, you will need to recompile the ntp package to enable that support. In a terminal window:

- sudo apt-get install gpsd gpsd-clients python-gps
- sudo apt-get install pps-tools

You will need to configure the gpsd program to use the GPIO serial port. In a terminal window:

- sudo dpkg-reconfigure gpsd

Configuring GPSD

- On the first screen, answer **Yes** to automatically starting gpsd and press **Enter**.
- Next screen, answer **No** to handling USB gps devices and press **Enter**.
- Next screen, change the device to **/dev/ttyAMA0** and press the **Tab** key to highlight the **Ok** and press **Enter**.
- On the next screen, enter **-n** for the options. (*This tells **gpsd** no-waiting before connecting to the GPS module.*) Press **Tab** and **Enter** to accept.
- You are now on the last screen. **Tab** and **Enter** to accept the default.

Reboot your system and log in again (see above).

Testing the GPS

(please excuse the pun) Now it is TIME to check the GPS. Your Raspberry Pi has rebooted and you have logged in. You will have to wait until the GPS module has acquired enough signals to get a sync lock. In a terminal window:

- sudo ppstest /dev/pps0

If the GPS module has a lock, you should see a parade of lines looking similar to:

source 0 - assert 1351501153.999956346, sequence: 47481 - clear 0.000000000, sequence: 0

Hit Control-C to stop the program.

Next enter in:

- sudo cgps -s

You should get a box in the upper left of the screen. After a couple of seconds, the box should populate with your GPS location information. If the program times out, two things to check:

- Is the GPS module synced / locked? You may have lost sync. Try repositioning the GPS module antenna.
- Is the baud rate correct for your module? Review the information in GPS Modifications.

Hit Control-C to stop the program.

Once you have verified that the GPS module works correctly, you will need to remove the gpsd program as it will interfere with the operation of the **ntpd** program. You can either remove the program like this:

- sudo apt-get remove gpsd

Or you can stop the gpsd program and edit the **/etc/rc.local** file to prevent gpsd from starting. If you choose this route, read further on to see what changes are needed.

NTP recompiling

As I stated earlier, the stock **ntp** module does not support using the timing pulse for accurate time information. So a recompile is necessary. Here are the instructions. Keep in mind that the author is using a terminal window to accomplish the recompile. Recompiling NTP

While the method is correct, the actual file names have changed over time. Here are the changes as of May 2015:

- wget http://archive.ntp.org/ntp4/ntp-4.2.6p5.tar.gz

will become

- wget http://archive.ntp.org/ntp4/ntp-4.2.8p2.tar.gz

- tar xvfz ntp-dev-4.2.7p397.tar.gz

will become

- tar xvfz ntp-4.2.8p2.tar.gz

- cd ntp-dev-4.2.7p397

will become

- cd ntp-4.2.8p2

Once you have finished making the replacement **ntp** programs, I suggest stopping the original **ntp** service. Next I recommend removing the original **ntp** as it is probably not a good thing to have two different copies of the program available. Then the replacement programs need to be copied over to the correct directories. And lastly, just reboot your Raspberry Pi. In the terminal window, the sequence will look like this:

- sudo service ntp stop
- sudo apt-get remove ntp
- sudo cp /usr/local/bin/ntp* /usr/bin/
- sudo cp /usr/local/sbin/ntp* /usr/sbin/
- sudo reboot

Testing the new NTP Service

At this point, you have installed all the bits and pieces to create a time server from a stock Raspberry Pi computer. You have already configured the GPS module and verified that it is functioning correctly. You have verified that the timing pulse is properly processed by Raspbian Linux. You have replaced the stock **ntp** programs with ones which properly utilize the timing pulse. All that is left to do is to configure the **ntp** service and verify that the time is correct.

Step 1 is to backup the original **ntp** configuration thus allowing you to experiment with different settings without getting lost. In a terminal window:

- sudo cp /etc/ntp.conf /etc/ntp-conf.orig

Next step is to edit the configuration file and tell the **ntp** service to prefer/use the GPS data being passed via the **NEMA** device.

- sudo nano ntp.conf

You will need to add the following lines to activate the use of the GPS data and the PPS timing pulse.

- server 127.127.20.0 mode 0x11 minpoll 4 maxpoll 4 prefer
- fudge 127.127.20.0 flag1 1 refid NEMA stratum 15
- server 127.127.22.0 minpoll 4 maxpoll 4
- fudge 127.127.22.0 refid PPS

Locate the following line and remove the '#' from the beginning of the line to uncomment it :

- restricted -4 default kod notrap nomodify nopeer noquery

Save the changes.

Now a link will be needed to allow the ntpd program to access the serial device.

- sudo ln -s /dev/ttyAMA0 /dev/gps0

And restart the **ntp** service.

- sudo service ntp restart

For more information on the mode setting of the NEMA driver for ntp service:

http://www.eecis.udel.edu/~mills/ntp/html/drivers/driver20.html

Now it's time to test the **ntp** time service.

- ntpq -c rl

In the resulting text from running the command, look for **precision**= . The negative number listed is the power of 2. So a listing like **precision=-18** means that your Raspberry Pi Time Server is accurate to 2^{-18} power (about 4 ms precision)

- ntpq -pn

An **X** will to the left of the NEMA server once the **ntpd** program has a lock.

Now to fix the situation so the link will be permanent. Edit the rc.local file:

- sudo nano /etc/rc.local

Add the following text above the line which reads **exit 0** :

- ln -s /dev/ttyAMA0 /dev/gps0
- service gpsd stop
- service ntp restart

And **save** the changes. Reboot the Raspberry Pi and log in.

Verify the settings:

- ntpd -pn <== gives you a locked indication (**X** to the left of the NEMA server line)
- ps ax | grep gps <== does not show a line with **gpsd**
- date <== shows the correct date and time

If all of the check pass, your Raspberry Pi is ready as a stand alone Stratum 1 time server.