BUILDING A RASPBERRY PI ZERO GPS NETWORK TIME SERVER FOR UNDER $50

Richard E. Schmidt, *R. E. Schmidt & Associates*
schmidt.rich@gmail.com

BIOGRAPHY

Richard E. Schmidt is an independent consultant and former staff member of the Time Service Department, U.S. Naval Observatory, specializing since 1994 in the design of stratum-1 servers of Network Time Protocol (NTP).

INTRODUCTION

Since publishing "Developing Low-cost NTP Servers with Linux PTP and GPS" in December, 2014 the recipe for building a low-cost GPS NTP server has become significantly cheaper. The release of the "$5" Raspberry Pi Zero provides a very low-cost LINUX platform running on the 1GHz Broadcom BM2835 System-on-Chip with dual core VideoCore IV GPU, 512MB RAM, 1080P HDMI video output, a MicroUSB port and a 40-pin GPIO header socket. The Raspbian operating system runs from a microSD card. The Raspberry Pi Zero system board measures 65mm x 30mm x 5mm.

This platform makes an excellent small-format Network Time Protocol (NTP) stratum-1 server when connected to a timing GPS receiver. Stratum-1 NTP servers incorporate hardware clocks such as GPS or atomic or radio clocks. Most national timing labs operate stratum-1 servers to facilitate distribution of standard time at millisecond accuracy to public, private, and internal computer clients.

PARTS LIST

These are the parts used for this project. In addition an optional wood Zebra case is shown in the images. Prices for the Raspberry Pi Zero vary with the prevailing market when supplies are limited.

ITEM	COST (2016)
Raspberry Pi Zero	$5 - $30
Ublox NEO-6 GPS Module + antenna	$19
16GB Class 10 MicroSD card	$6
MicroUSB RJ45 Ethernet adapter	$12
5V 2A MicroUSB power supply	$5
patch cable	$1
Total	$49

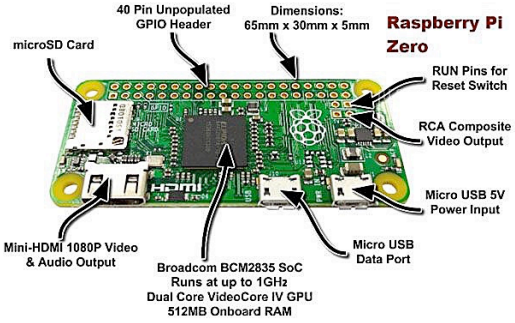


Fig. 1 Raspberry Pi Zero

The GPIO wiring of the Raspberry Pi (all models, including RP Zero) is shown in Figure 2 below. The connections between the GPS and the Raspberry Pi GPIO are:

```
GPS      RPI
---      ---
RX  --> TXD
TX  --> RXD
PPS --> GPIO #24
GND --> GND
VIN --> 3.3V0
```

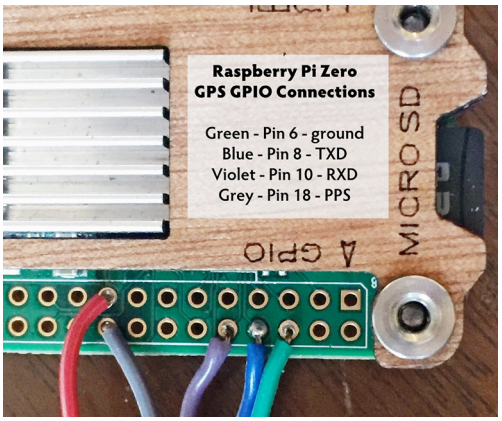


Fig. 2 GPIO wiring for GPS PPS

A very cost-effective GPS receiver is the *GPS Module w/ Ceramic Passive Antenna for Raspberry Pi / Arduino – Red*, Item #901384916 from *dxsoul.com*, each $18.74 (Feb. 2016). Figure 3 below shows the direct header

wiring for the five leads from the Raspberry Pi Zero. This GPS provides a uFl connector for a straight (not RP) uFl-TNC cable or antenna, however the attached ceramic patch antenna works quite well indoors about 6 feet from a window.

Fig. 3 GPS module header wiring

LINUX KERNEL PPS

The Linux kernel Pulse-per-Second (PPS) API is found in current LINUX kernels including Raspbian 4.1.13+ (Jan. 2016), and kernel PPS framework is supported by the NTP software suite (www.ntp.org; we are using NTP 4.3.75). Our GPS receiver will provide both 1PPS on-time pulses and time of day information via the NMEA protocol (Fig. 4) The gpsd program (*www.catb.org/gpsd/*) provides NMEA to the NTP daemon via a shared memory interface:

```
apt-get install gpsd gpsd-clients python-gps
```

SYSTEM CONFIGURATION

The 1PPS pulses from our GPS are input on GPIO24 (pin 18). We enable this for the Raspberry Pi in `/boot/config.txt`:

Edit `/boot/config.txt` – Add
`dtoverlay=pps-gpio,gpiopin=24` on a line.

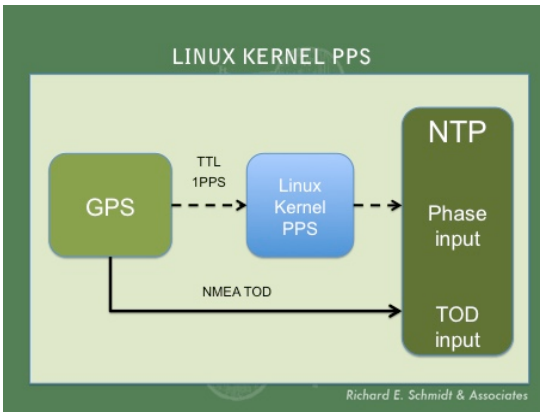


Fig. 4 Linux Kernel PPS

Edit `/etc/cmdline.txt` to include this modification on a single line (remove `console=tty1`):

```
dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
nohz=off ipv6.disable=1
```

Edit `/etc/inittab and comment out:`

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200
vt100
```

Edit */etc/modules* – Add *pps-gpio* on a new line.

At boot Linux PPS will create `/dev/pps0;` we must supply two symbolic links required by the NTP software, `/dev/gps0` and `/dev/gpspps0` by creating the new file:
`/etc/udev/rules.d/80-gps-pps.rules`

```
# Provide a symlink or two to /dev/ttyAMA0
KERNEL=="ttyAMA0",              SUBSYSTEM=="tty",
SYMLINK+="gps0", MODE="0666"
KERNEL=="ttyAMA0",  RUN+="/bin/setserial  /dev/%k
low_latency"
KERNEL=="pps0",   SUBSYSTEM=="pps",   DRIVER=="",
SYMLINK+="gpspps0", MODE="0666"
```

At boot the following messages confirm our correct setup:
`dmesg | grep –i pps`

```
pps pps0: new PPS source pps.-1
pps pps0: Registered IRQ 418 as PPS source
pps_ldisc: PPS line discipline registered
pps pps1: new PPS source ttyAMA0
pps pps1: source "/dev/ttyAMA0" added
```

The `ppstest and ppswatch` utilities in the Linux package "*pps-tools*" is used to verify the kernel PPS:

```
root@rpzero2:~# ppstest /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1455129612.999999541, sequence: 82981
- clear  0.000000000, sequence: 0
source 0 - assert 1455129614.000005521, sequence: 82982
- clear  0.000000000, sequence: 0
source 0 - assert 1455129615.000000500, sequence: 82983
- clear  0.000000000, sequence: 0
```

```
source 0 - assert 1455129615.999997477, sequence: 82984
- clear  0.000000000, sequence: 0

root@rpzero2:~# ppswatch -a /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
timestamp: 1455129399, sequence: 82767, offset:    141
timestamp: 1455129400, sequence: 82768, offset:   -930
timestamp: 1455129401, sequence: 82769, offset:      0
timestamp: 1455129402, sequence: 82770, offset:    -72
timestamp: 1455129403, sequence: 82771, offset:  -2143
timestamp: 1455129404, sequence: 82772, offset: -12220
timestamp: 1455129405, sequence: 82773, offset:  -1297
timestamp: 1455129406, sequence: 82774, offset:  -1374
timestamp: 1455129407, sequence: 82775, offset:  -1452
timestamp: 1455129408, sequence: 82776, offset:   -529
timestamp: 1455129409, sequence: 82777, offset:  -2606
timestamp: 1455129410, sequence: 82778, offset:    316
timestamp: 1455129411, sequence: 82779, offset:   -761
timestamp: 1455129412, sequence: 82780, offset:  -1799
timestamp: 1455129413, sequence: 82781, offset:  -1837

Total number of PPS signals: 15
Maximum divergence: 12220
```

CONFIGURING NTP SOFTWARE

See the Raspberry Pi documentation for setting a static IP address for your NTP server. No monitor is needed; connect via Secure Shell from another computer using the microUSB Ethernet adapter. NTP software can be dowloaded from www.ntp.org. First create the directory /usr/local/ntp and unpack the tar distribution into the location `/usr/local/ntp/ntp-dev-4.3.75`. If you do not have it, install `libcap-dev`:

```
apt-get install libcap-dev
```

Here is an example compile script for NTP version 4.3.75.

```
#CONFIG.sh
#
VER=4.3.75
#
#
./configure --prefix=/usr/local/ntp/4.3.75  --disable-
parse-clocks \
--disable-all-clocks \
--enable-ATOM --enable-SHM --enable-debugging --
sysconfdir=/var/lib/ntp --with-sntp=no \
--without-openssl \
--disable-ipv6 \
# be sure to apt-get libcap-dev
--enable-linuxcaps \
--with-lineeditlibs=edit --without-ntpsnmpd --disable-
local-libopts \
--disable-dependency-tracking && make install
make -j5 install
cd /usr/local/ntp/sbin/
strip ntpd
strip ntpdate
cd /usr/local/ntp/bin
strip ntpq
strip ntpdc
# end of script
```

In `/usr/local/ntp` create convenience symbolic links:

```
#Makelinks.sh
cd /usr/local/ntp
rm ./bin ./lib ./libexec ./share ./sbin
./include
NEW=4.3.75
for FILE in  ./$NEW/*
do
echo ln -s   $FILE $PWD/`basename $FILE`
 ln -s   $FILE $PWD/`basename $FILE`
done
#end of script
```

The following links are created:

```
bin -> ./4.3.75/bin
include -> ntp-dev-4.3.75/include
libexec -> ./4.3.75/libexec
sbin -> ./4.3.75/sbin
share -> ./4.3.75/share
```

To prevent NTP from being subsequently *downgraded* when you next update the operating system, do:

```
sudo apt-mark hold ntp
```

The `/etc/ntp.conf` file selects the NTP NMEA and SHM modules, and one external server:

```
# refclock 20 NMEA
server 127.127.22.0 minpoll 3 maxpoll 3
# refclock 28 SHM - with gpsd
server 127.127.28.0 prefer minpoll 3 maxpoll 3
server tick.usno.navy.mil  iburst
fudge 127.127.22.0 time1 0 flag3  1 refid PPS
fudge 127.127.28.0 time1 0.43 refid GPS
```

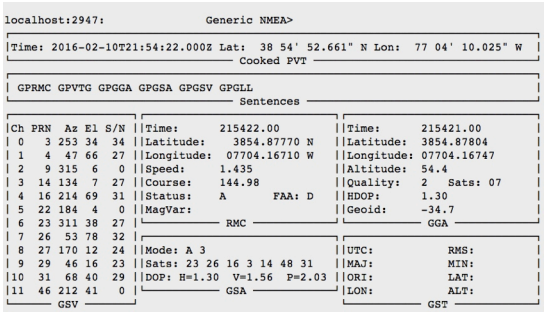
Refer to the NTP documentation for other configuration items, including logging performance. The runtime command line for NTP is:

```
/usr/local/ntp/sbin/ntpd -p /var/run/ntpd.pid -g
-4 -U 0 -l /var/log/ntpd.log -u 102:104
```

You should use the `ntpdate` utility or the `date` command to set the system time to within a few minutes prior to starting NTP.

GPSMON

The gpsmon utility included with gpsd provides a monitor to show GPS acquisition:



Using the NTP NMEA driver with PPS, the Raspberry Pi server is surprisingly stable with under 5 microseconds s.d. in loopstats logs. Figure 5 shows NTP loopstats offsets for the Raspberry Pi Zerorunning NTP 4.3.75@1.2483.

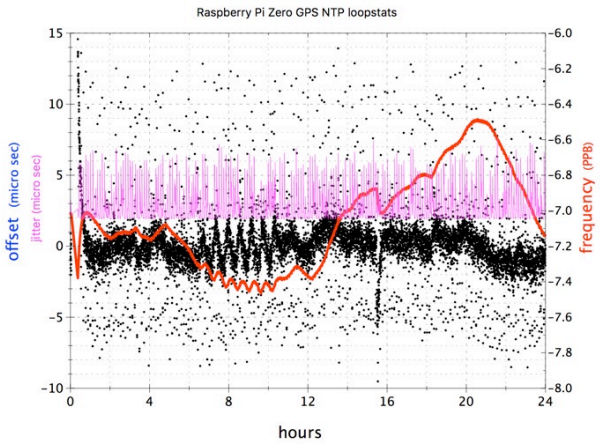


Fig. 5 Raspberry Pi NTP loopstats

Figure 6 shows the same data smoothed 10x.

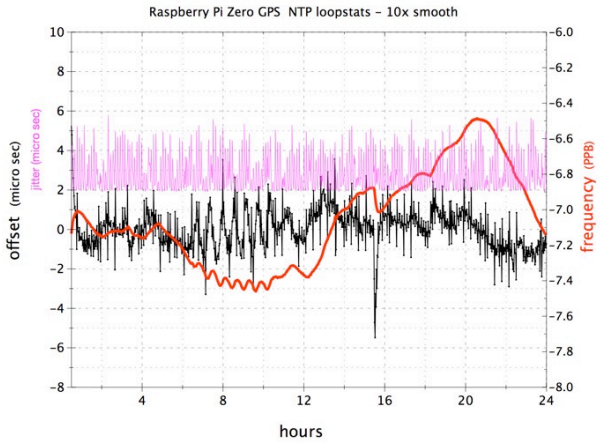


Fig. 6 Raspberry Pi NTP loopstats 10x smoothing

FOR A FEW DOLLARS MORE

For about $10 more, one may find the same ublox NEO-6M GPS with PPS wired on a board with a 40-pin header for the Raspberry Pi Zero and a convenient male TNC connector. (See Fig. 7 below). The software configuration is identical.

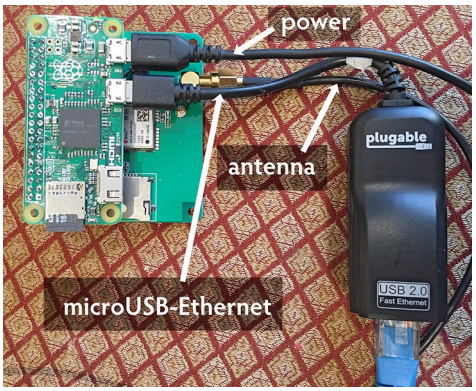


Fig. 7 NEO-6M GPS with 40-pin header and TNC

The Rpi-GPS V2.0 module (ebay.com) is shown in Fig. 8 below installed on a Raspberry Pi Model B.

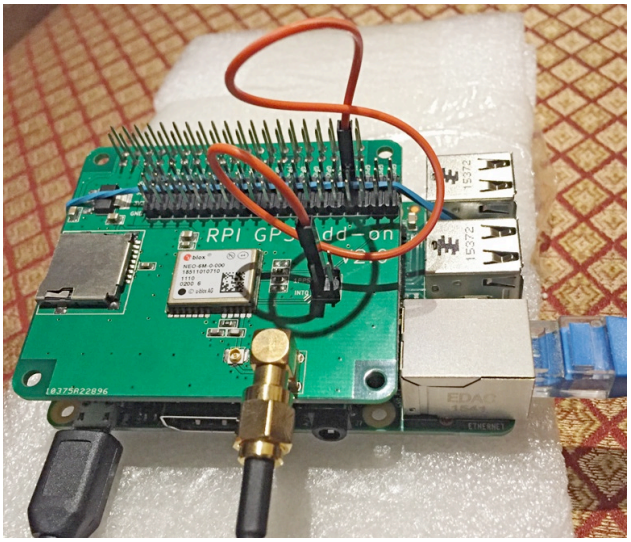


Fig. 8 RPI-GPS on Raspberry Pi, with output PPS jumpered to GPIO24.