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

Presentation · March 2019

CITATIONS
0

READS
2,490

3 authors, including:

Richard Eugene Schmidt
U. S. Naval Observatory (retired)
28 PUBLICATIONS **56** CITATIONS

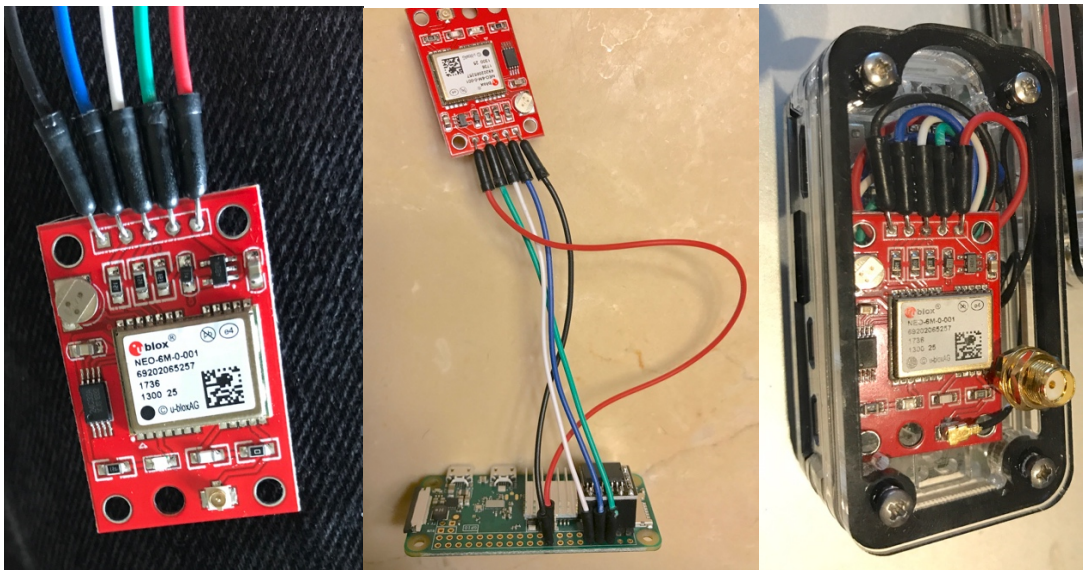
SEE PROFILE

BUILDING A RASPBERRY PI ZERO-W 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

The first stratum 1 NTP servers that I designed cost over $15K each in 1994. Today one can build a functional equivalent for under $50. Since publishing "Developing Low-cost NTP Servers with Linux PTP and GPS" (google it) 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 (adafruit.com) 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, a port of Debian Linux, runs from a MicroSD card. The Raspberry Pi Zero system board measures only 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.

In the following project, a Raspberry Pi running Raspbian Stretch Lite uses "gpsd" (*www.catb.org/gpsd/)* to transfer both GPS NEMA or receiver proprietary messages and 1PPS to NTP via the Pi GPIO header. We will wire the 1PPS output of the GPS to one of the GPIO pins (GPIO24) and the GPS RXD, TXD to the Pi TXD, RXD pins, GPIO14 and GPIO15. These are the hardware UART lines used by serial console /dev/ttyAMA0, which we must disable as outlined below.

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 (2019)
Raspberry Pi Zero, or Zero W	$5, $10
Ublox NEO-6 GPS Module	$4

SMA GPS external active antenna	$3
32GB Class 10 MicroSD card	$8
MicroUSB RJ45 Ethernet adapter	$12
5V 2A MicroUSB power supply	$4
Ufl – SMA antenna patch cable	$1
Total	$42

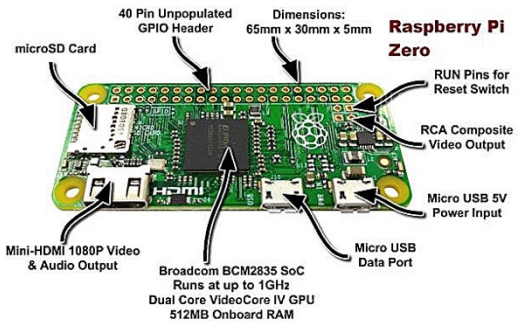


Fig. 1 Raspberry Pi Zero (Pi Zero-W = WiFi is slightly different).

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

```
GPS      RPI
---      ---
RX  --> TXD
TX  --> RXD
PPS --> GPIO #24 (Pin 18)
GND --> GND
VCC --> 3.3V
```

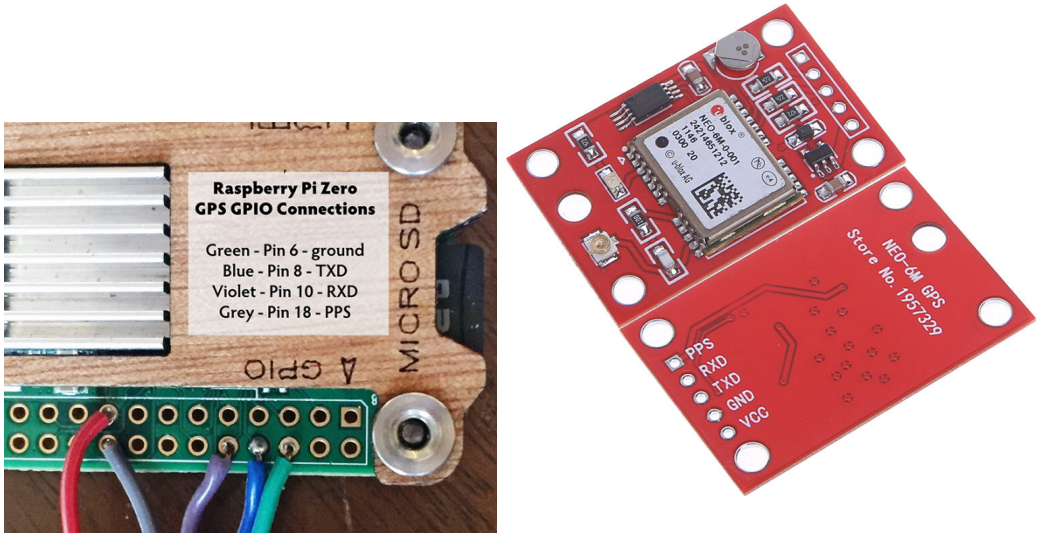


Fig. 2 GPIO wiring for GPS PPS

A very cost-effective GPS receiver is the *GYNEO6MV2 GPS NEO-6M GY-NEO6MV2 Board Module with Antenna,* each $3 (eBay, Feb. 2019), above right. This GPS provides a uFl connector for a straight (not RP) uFl to TNC cable antenna.

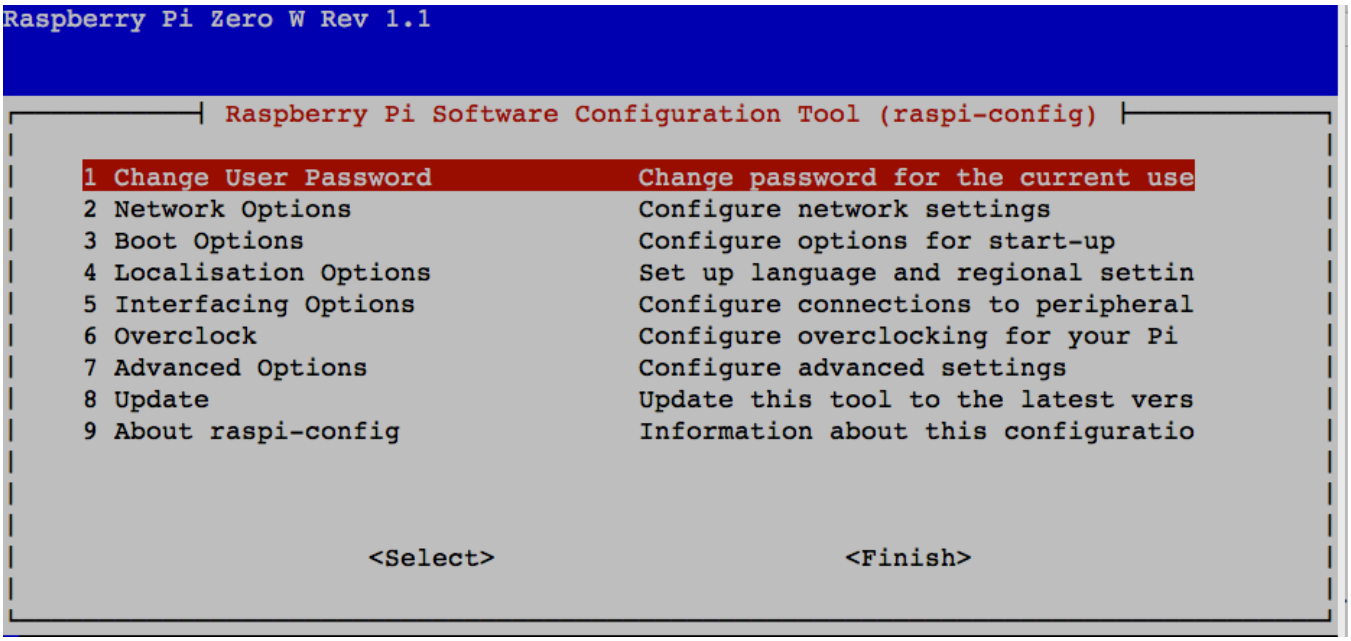
This version GPS/NTP server features the Raspberry Pi Zero W (WiFi). It obtains its IP address from DHCP, both for WiFi and if an Ethernet USB dongle is attached to the local router. You can configure WiFi using a USB keyboard and HDMI monitor if you do not have an Ethernet dongle. The file configuring WiFi access is:
`/etc/wpa_supplicant/wpa_supplicant.conf`

```
It looks like this:
# more /etc/wpa_supplicant/wpa_supplicant.conf
```

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="YourSSID"
    psk="YourPassword"
}
```

RASPBERRY PI CONFIGURATION
The utility "raspi-config" enables setting various system attributes.



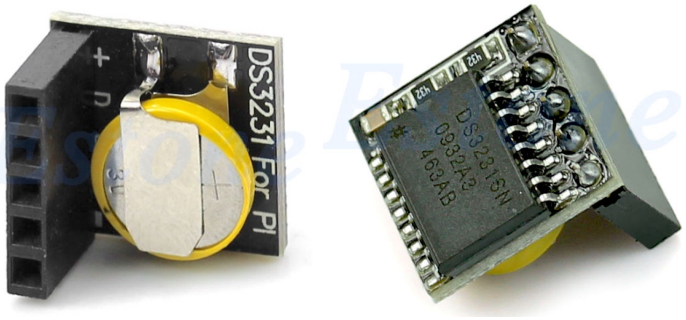
The 1PPS pulses from our GPS are input on GPIO24 (pin 18). We enable this for the Raspberry Pi in `/boot/config.txt`:

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

```
#For Pi3, RPi Zero W, overlay Bluetooth:
dtoverlay=pi3-miniuart-bt
enable_uart=1
```

ADD A DS3231 REAL_TIME CLOCK MODULE:

At right is a DS3231 RTC with a Raspbery Pi header soldered on which costs $3 (eBay,2019). This module uses a capacitor rather than a battery. It connects to GPIO pins 1=3.3V, 3=SDA, 5=SCL ,9=GND.
Edit `/boot/config.txt` – Add one line:
`dtoverlay=i2c-rtc,ds3231`



Use `raspi-config` under `Advanced` to select I2C and enable. Reboot. Install i2C utils:
`apt-get install i2c-tools.` Verify

the connection using
`i2cdetect -y 1`
to show UU at matrix 0x68.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Copy system time to RTC:

```
#hwclock -s
```

```
Check the time:
#hwclock -r -D
hwclock from util-linux 2.29.2
Using the /dev interface to the clock.
Last drift adjustment done at 1550956976 seconds after 1969
Last calibration done at 1550956976 seconds after 1969
Hardware clock is on UTC time
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
/dev/rtc does not have interrupt functions. Waiting in loop for time from /dev/rtc
to change
...got clock tick
Time read from Hardware Clock: 2019/03/05 20:39:26
Hw clock time : 2019/03/05 20:39:26 = 1551818366 seconds since 1969
Time since last adjustment is 861390 seconds
Calculated Hardware Clock drift is 0.000000 seconds
2019-03-05 20:39:25.633257+0000
```

Now remove the “fake-hwclock”:

```
apt-get -y remove fake-hwclock
update-rc.d -f fake-hwclock remove
systemctl disable fake-hwclock
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  fake-hwclock
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 32.8 kB disk space will be freed.
(Reading database ... 43583 files and directories currently installed.)
Removing fake-hwclock (0.11+rpt1) ...
Processing triggers for man-db (2.7.6.1-2) ...
root@rpz:~# update-rc.d -f fake-hwclock remove
root@rpz:~# systemctl disable fake-hwclock
Synchronizing state of fake-hwclock.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable fake-hwclock
```

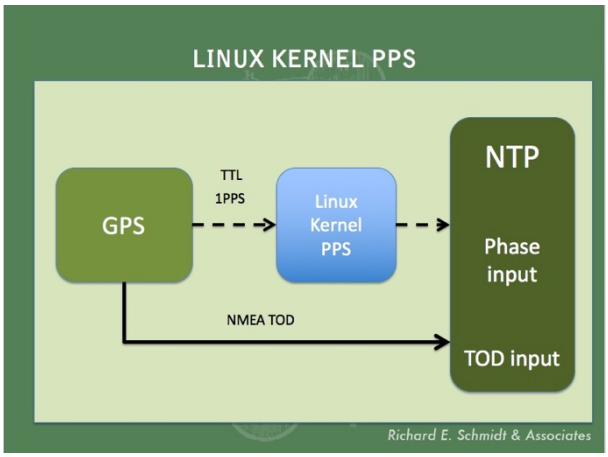
Edit /lib/udev/hwclock-set and comment out the lines:

```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi
```

LINUX KERNEL PPS

The Linux kernel Pulse-per-Second (PPS) API is found in current LINUX kernels including Raspbian, and kernel PPS framework is supported by the NTP software suite (www.ntp.org). Our GPS receiver will provide both 1PPS on-time pulses and time of day information via the NMEA and/or ublox protocol. The "gpsd" program provides NMEA to the NTP daemon via a shared memory interface as well as through a serial device; install it as:

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



The Raspberry PI will use the hardware UART /dev/ttyAMA0 and the kernel PPS device /dev/pps0.
Edit `/etc/cmdline.txt` to include this modification **on a single line** (remove `console=ttyAMA0` and `kgdboc=ttyAMA0`):

```
dwc_otg.lpm_enable=0 console=tty1 kgdboc=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline fsck.repair=yes rootwait ipv6.disable=1
```

Edit */etc/modules* – Add on a new line:

```
pps-gpio
```

Install setserial:

```
apt-get install setserial
```

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"
```

(If 25-gps-pps.rules exists remove it.) 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 package package "*pps-tools*" is used to verify the kernel PPS. (`apt-get install pps-tools`).

```
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
…

#ppswatch -a /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
timestamp: 1551153479, sequence: 868, offset:  84182
timestamp: 1551153480, sequence: 869, offset:  79249
…
Total number of PPS signals: 12
Maximum divergence: 84182
```

INSTALL NTP

Two ways to install

1. Repository install: `apt-get install ntp`

`- or install source code and compile, so that we can select options.`

2. Source code install:

Install git: `apt-get install git`

`cd /usr/src [or /usr/local  or other convenient spot].`

```
git clone https://github.com/ntp-project/ntp.git
Cloning into 'ntp'...
remote: Enumerating objects: 112428, done.
remote: Total 112428 (delta 0), reused 0 (delta 0), pack-reused 112428
Receiving objects: 100% (112428/112428), 69.99 MiB | 422.00 KiB/s, done.
Resolving deltas: 100% (95340/95340), done.
```

CONFIGURING NTP SOFTWARE

See the Raspberry Pi documentation for setting a static IP address for your NTP server. I am currently using NTP version 4.2.8p11@1.3728-o and 4.3.75 . 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…`. **If you do not have it, install `libcap-dev`:**

```
apt-get install libcap-dev libtool lynx autoconf
```

Here is an example compile script for NTP:

```
#CONFIG.sh
#
VER=4.2.8p6
#
#
#Prerequisites:
#apt-get install libcap-dev libtool lynx autoconf
#./bootstrap

./configure --prefix=/usr/local/ntp/${VER}  --disable-parse-clocks \
--disable-all-clocks --enable-CLOCK_GPSDJSON \
--enable-NMEA --enable-ATOM --enable-SHM --enable-debugging --
sysconfdir=/var/lib/ntp --with-sntp=no \
--without-openssl --disable-ipv6 --enable-linuxcaps \
--with-lineeditlibs=edit --without-ntpsnmpd --disable-local-libopts \
--disable-dependency-tracking && make -j5 install
```

```
# 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 second shared memory as "prefer" (because PPS requires one preferred server).

```
# Listen also on WiFi if you have it.
interface listen wlan0
interface listen eth0

# refclock 22 PPS
server 127.127.22.0 minpoll 4 maxpoll 4
fudge 127.127.22.0 time1 0 flag3 1 flag4 1 refid PPS
# refclock 28 SHM - with gpsd
server 127.127.28.0  minpoll 4 maxpoll 4
fudge 127.127.28.0 flag1 1 time1 0.1350 refid GPS stratum 1

# refclock 28 SHM - SHM2 source
server 127.127.28.2  minpoll 4 maxpoll 4 prefer
fudge 127.127.28.2 flag1 1  refid SHM2
#
```

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.

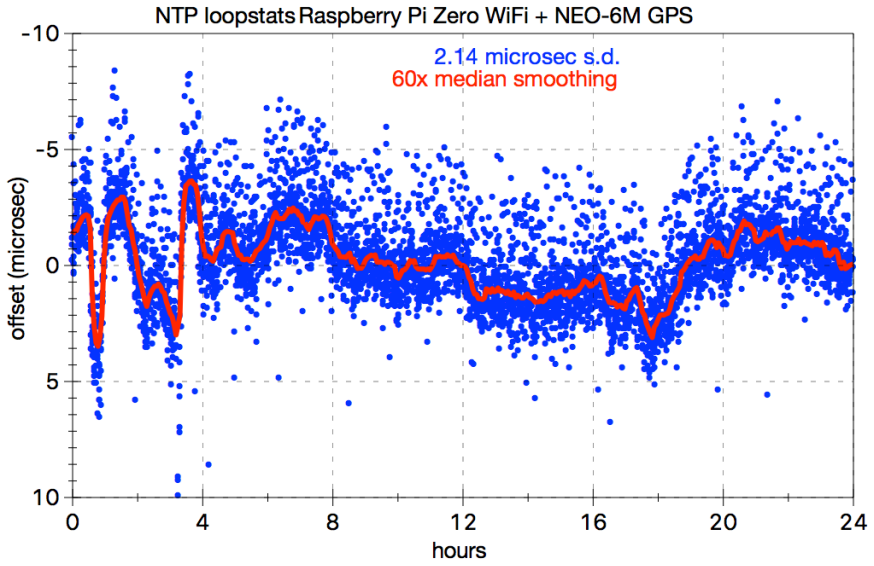
GPSMON

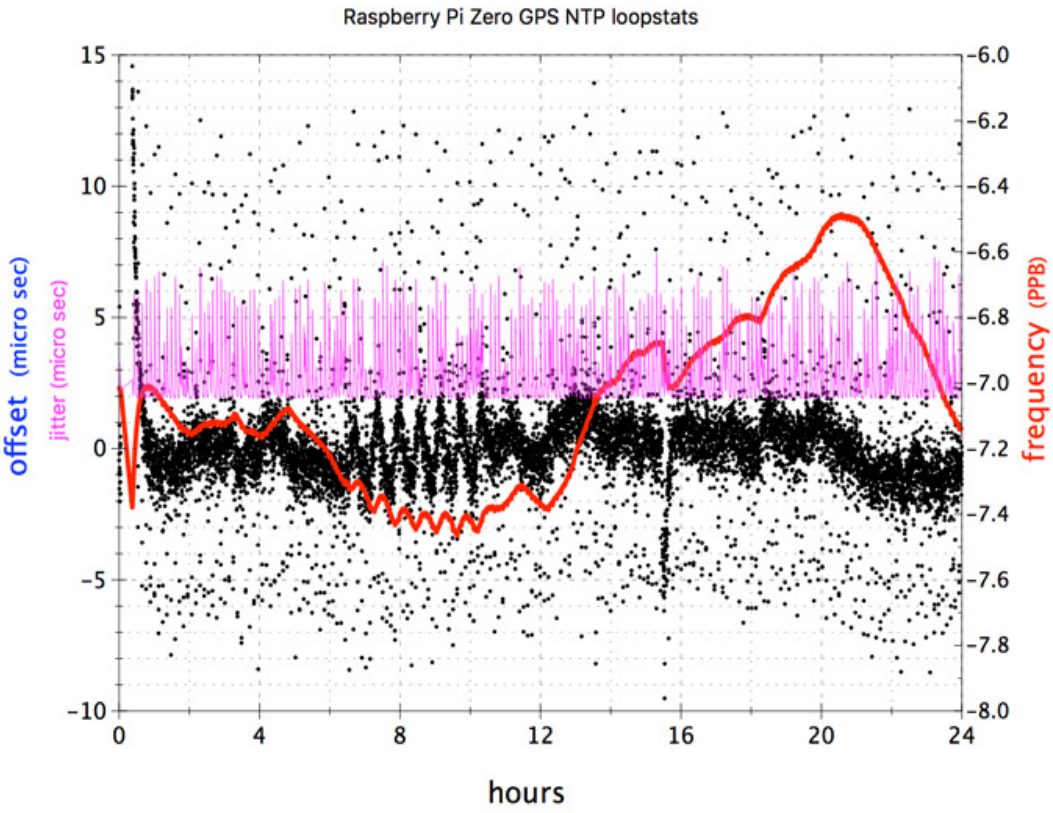
The gpsmon utility included with gpsd provides a monitor to show GPS acquisition:

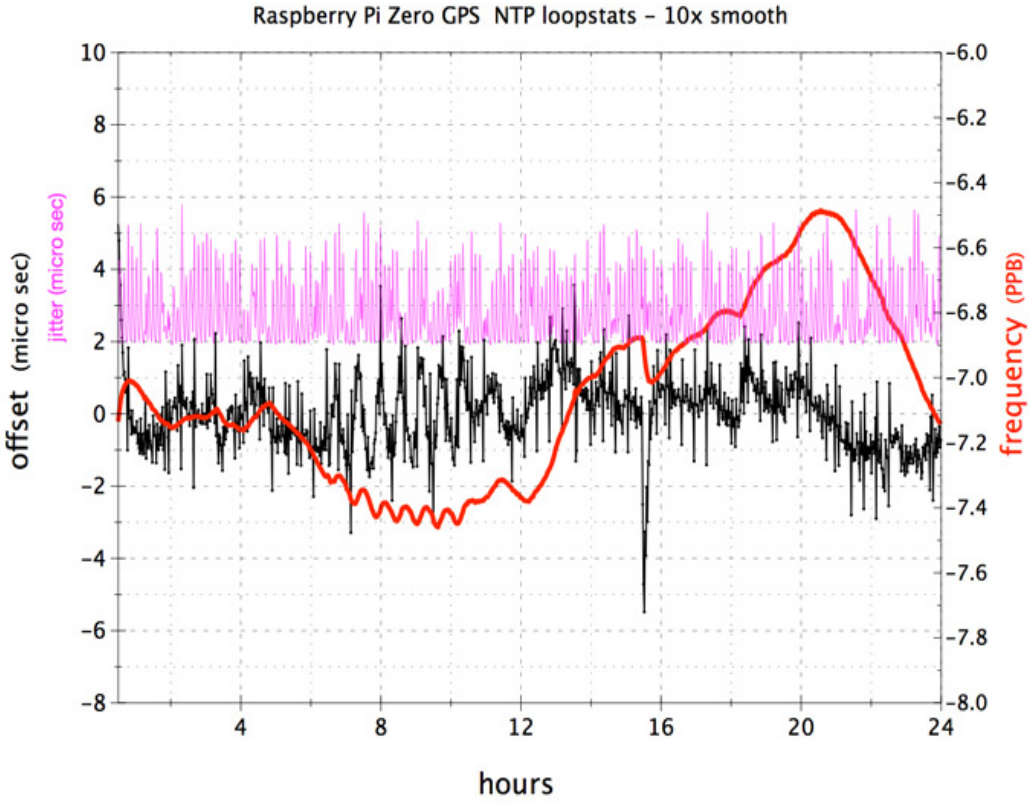
```
#gpsmon -n
```

```
localhost:2947:                Generic NMEA>
┌──────────────────────────────────────────────────────────────────────────────┐
|Time: 2016-02-10T21:54:22.000Z Lat:  38 54' 52.661" N Lon:  77 04' 10.025" W  |
└──────────────────────────── Cooked PVT ──────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
| GPRMC GPVTG GPGGA GPGSA GPGSV GPGLL                                          |
└──────────────────────────── Sentences ──────────────────────────────────────┘
┌──────────────────┐┌──────────────────────────┐┌──────────────────────────┐
|Ch PRN  Az El S/N ||Time:       215422.00      ||Time:       215421.00      |
| 0   3 253 34  34 ||Latitude:    3854.87770 N  ||Latitude:  3854.87804     |
| 1   4  47 66  27 ||Longitude:  07704.16710 W  ||Longitude: 07704.16747    |
| 2   9 315  6   0 ||Speed:      1.435          ||Altitude:  54.4           |
| 3  14 134  7  27 ||Course:     144.98         ||Quality:   2   Sats: 07   |
| 4  16 214 69  31 ||Status:     A       FAA: D ||HDOP:      1.30           |
| 5  22 184  4   0 ||MagVar:                   ||Geoid:     -34.7          |
| 6  23 311 38  27 |└──────────── RMC ──────────┘└──────────── GGA ──────────┘
| 7  26  53 78  32 |┌──────────────────────────┐┌──────────────────────────┐
| 8  27 170 12  24 ||Mode: A 3                ||UTC:           RMS:       |
| 9  29  46 16  23 ||Sats: 23 26 16 3 14 48 31||MAJ:           MIN:       |
|10  31  68 40  29 ||DOP: H=1.30  V=1.56  P=2.03||ORI:          LAT:       |
|11  46 212 41   0 |└──────────── GSA ──────────┘|LON:           ALT:       |
└──── GSV ─────────┘                           └──────────── GST ──────────┘
```

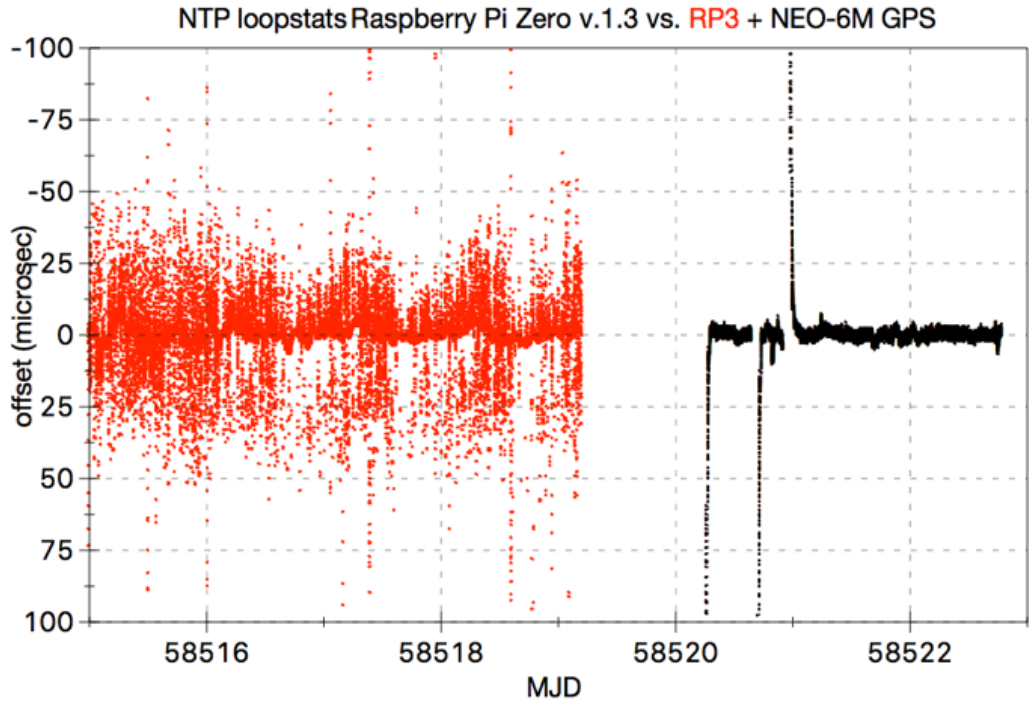
Using the NTP NMEA driver with PPS, the Raspberry Pi server is surprisingly stable with under 5 microseconds s.d. in loopstats logs. Below, we show NTP loopstats offsets for the Raspberry Pi Zero running NTP 4.3.75@1.2483.



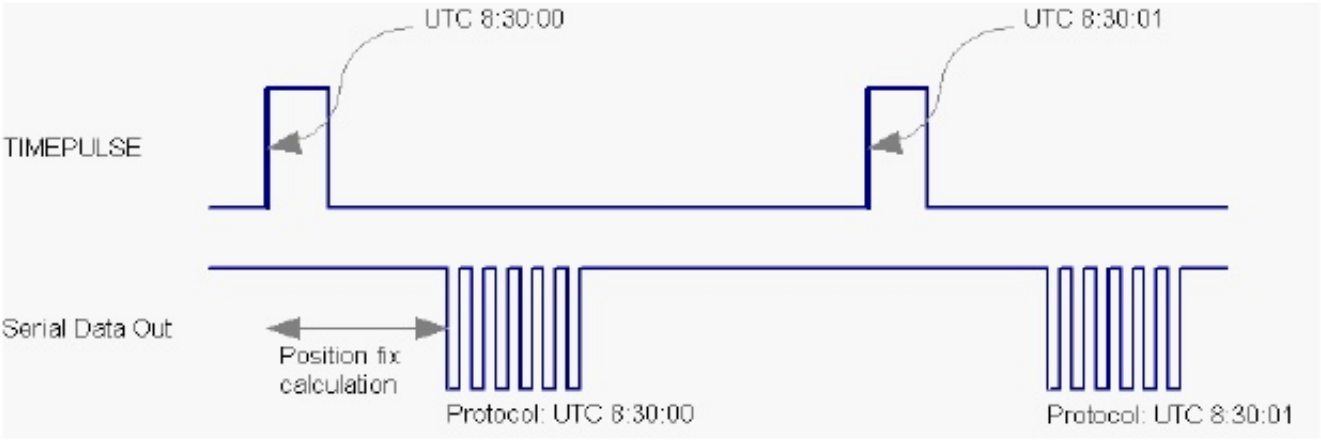

Raspberry Pi Zero GPS NTP loopstats
offset (micro sec)
jitter (micro sec)
frequency (PPB)
hours


Raspberry Pi Zero GPS NTP loopstats – 10x smooth
offset (micro sec)
jitter (micro sec)
frequency (PPB)
hours

More CPUs does not equate to better timing performance. Here we show the NTP loopstats offsets of the system clock from the GPS reference time, on the left in red for an RPi Model 3B quad-core, and on the right, in black, for the single core RPi Zero.



ALIGNING THE TIME PULSE TRIGGER TO LEADING EDGE (OPTIONAL)



NTP can sync to the NMEA GPS output, but as shown above this has a timing offset from the timepulse. But NTP can also sync to the time pulse. We want to select the leading edge.

We can use the sysfs file system to read and write configuration values to GPIO pins. In this case we want to set the pulse edge on GPIO24 to "rising". We will check the debug kernel to see that pps-gpio is using GPIO24:

```
#mount -t debugfs none /sys/kernel/debug
#cat /sys/kernel/debug/gpio
```

```
gpiochip0: GPIOs 0-53, parent: platform/20200000.gpio, pinctrl-bcm2835:
 gpio-24  (                    |pps-gpio           ) in  lo IRQ
 gpio-47  (                    |led0               ) out lo

#lsmod
Module                  Size  Used by
pps_gpio                3089  1
pps_core                8606  2 pps_gpio

#service gpsd stop
#service ntp stop
```

Now unload the pps-gpio module that is using GPIO24:

```
#rmmod pps_gpio

#cat /sys/kernel/debug/gpio

gpiochip0: GPIOs 0-53, parent: platform/20200000.gpio, pinctrl-bcm2835:
 gpio-47  (                    |led0               ) out lo
```

Run this script to set GPIO24 to “rising edge”:

```
#sh -v ./Setp.sh
#!/bin/bash
GNO=24

# Check if gpio is already exported
if [ ! -d /sys/class/gpio/gpio$GNO ]
then
  echo $GNO > /sys/class/gpio/export
  sleep 1 ;# Short delay while GPIO permissions are set up
fi
# Set to pulse edge
echo "rising" > /sys/class/gpio/gpio$GNO/edge
sleep 1
cat /sys/class/gpio/gpio$GNO/edge

echo $GNO > /sys/class/gpio/unexport
```

Then reboot.

TWEAKING GPS OFFSET IN NTP:

```
#ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
o127.127.22.0    .PPS.            0 l   11   16   17    0.000  307.844   3.250
*127.127.28.0    .GPS.            1 l   12   16   37    0.000   25.968  30.114
+127.127.28.2    .SHM2.           0 l   11   16   77    0.000  307.844 231.094
+192.5.41.40     .PTP.            1 u   23   64    3   18.365  311.067   8.291
 192.5.41.41     .IRIG.           1 u   23   64    3   60.054  309.788   9.400
 129.6.15.29     .NIST.           1 u   22   64    3   15.409  310.794   7.484
```

Change the value of time1 in /etc/ntp.conf:

```
# refclock 28 SHM - with gpsd
server 127.127.28.0  minpoll 4 maxpoll 4 prefer
```

```
fudge 127.127.28.0 flag1 1 time1 0.50 refid GPS stratum 1
```

Try

```
fudge 127.127.28.0 flag1 1 time1 0.75 refid GPS stratum 1
```

```
#ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
o127.127.22.0    .PPS.            0 l    5   16    1    0.000   52.344   0.002
*127.127.28.0    .GPS.            1 l    4   16    3    0.000   40.514  16.954
+127.127.28.2    .SHM2.           0 l    4   16    3    0.000   52.094   2.222
+192.5.41.40     .IRIG.           1 u   11   64    1   15.518   52.453   3.672
 192.5.41.41     .IRIG.           1 u   18   64    1   42.878   57.297   0.002
 129.6.15.29     .NIST.           1 u   19   64    1   12.908   56.457   0.002
```

This is better. Let it run for a while.

```
#ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
o127.127.22.0    .PPS.            0 l    6   16  377    0.000    0.009   0.002
+127.127.28.0    .GPS.            1 l    5   16  377    0.000    4.014   1.426
*127.127.28.2    .SHM2.           0 l    4   16  377    0.000    0.009   0.002
 10.0.0.160      .PPS.            1 u   23   64  377    5.217    1.815   0.221
 192.5.41.40     .INIT.          16 u    -  128    0    0.000    0.000   0.000
 192.5.41.41     .INIT.          16 u    -  128    0    0.000    0.000   0.000
 129.6.15.29     .NIST.           1 u   37   64  377   18.133    3.618   4.404
```