Exact Time on Raspberry

Content

Configure Raspberry Pi as a GPS-PPS NTP Server with Chrony

This guide configures the **Grove - GPS (Air530)** module on a **Raspberry Pi 4 (64-bit OS)** to act as a **high-precision NTP server** using **GPS and PPS (Pulse Per Second)**.

Step 1: Connect the hardware

Air530 (GPS)	Raspberry Pi (GPIO)
VCC (3.3V)	3.3V (Pin 1)
GND	GND (Pin 6)
TX	GPIO 15 (RXD, Pin 10)
RX	GPIO 14 (TXD, Pin 8)
PPS	GPIO 18 (Pin 12)

Step 2: Enable UART and PPS in Raspberry Pi configuration

Edit the boot configuration:

```
sudo nano /boot/firmware/config.txt
```

Add these lines at the end:

```
enable_uart=1
dtoverlay=pps-gpio,gpiopin=18
```

Save (CTRL + X, then Y, then Enter).

```
sudo reboot
```

Disable Serial console and enable serial port

```
sudo raspi-config
```

Navigate to **"Interfacing Options" > "Serial Port"**:

- Disable **serial console** → Select **No**.
- Enable **serial hardware** → Select **Yes**.

```
sudo reboot
```

Step 3: Install required packages

```
sudo apt update && sudo apt upgrade -y && sudo rpi-update -y && sudo apt
install -y gpsd gpsd-clients chrony pps-tools
```

Step 4: Verify PPS signal

Check if the **PPS module** is loaded:

```
lsmod | grep pps
```

You should see:

```
pps_gpio               12288  0
```

```
sudo ppstest /dev/pps0
```

Expected output:

```
source 0 - assert 1740696218.993111372, sequence: 1091 - clear  0.000000000,
sequence: 0
source 0 - assert 1740696219.992820787, sequence: 1092 - clear  0.000000000,
sequence: 0
```

If PPS timestamps appear, the signal is working.

Step 5: Verify GPS data

Test raw GPS data:

```
sudo cat /dev/serial0
```

- You should see **NMEA sentences** ($GPGGA, $GPRMC, etc.).
- Press CTRL + C to stop.

Start gpsd manually for debugging:

```
sudo systemctl stop gpsd.socket
sudo gpsd -n /dev/serial0 /dev/pps0
```

Check GPS data using:

```
cgps -s
```

If you see satellite data, the GPS is working.

Step 6: Configure GPSD for automatic start

Edit GPSD settings:

```
sudo nano /etc/default/gpsd
```

Update the file:

```
GPSD_OPTIONS="-n"
DEVICES="/dev/serial0 /dev/pps0"
GPSD_SOCKET="/var/run/gpsd.sock"
```

You can keep

```
USBAUTO="true"
```

Save and restart gpsd:

```
sudo systemctl enable gpsd
sudo systemctl restart gpsd
```

Verify GPSD status:

```
systemctl status gpsd
```

Step 7: Configure chrony to use GPS and PPS

Edit the Chrony configuration file:

```
sudo nano /etc/chrony/chrony.conf
```

Add these lines at the end:

```
# Use GPS as a time source (from gpsd shared memory)
refclock SHM 0 offset 0.5 delay 0.2 refid GPS noselect

# Use PPS for precise synchronization
refclock PPS /dev/pps0 lock GPS refid PPS prefer
```

Save (CTRL + X, then Y, then Enter).

Restart Chrony:

```
sudo systemctl restart chrony
sudo systemctl enable chrony
```

Verify Chrony GPS & PPS Sync

```
chronyc sources -v
```

Wait for 30 seconds

Example output:

```
#? GPS                           0   4     7    24   -255ms[ -250ms] +/-  101ms
#* PPS                           0   4     7    23   +384ns[+4623us] +/-  417ns
```

GPS (#- GPS) is detected but not used.

PPS (# PPS) is the preferred time source for high accuracy.*

Check system time:

```
timedatectl status
```

Look for System clock synchronized: yes.

Step 8: Allow other devices to sync (optional)

Edit **Chrony configuration**:

```
sudo nano /etc/chrony/chrony.conf
```

Add this line (adjust subnet for your network):

```
allow 192.168.1.0/24
```

Restart Chrony:

```
sudo systemctl restart chrony
```

Sync time from other devices

Linux

```
ntpdate -q 192.168.0.47
```

Address of your Raspberry NTP server

Windows

Open a CMD window as administrator

```
net start w32time
w32tm /config /manualpeerlist:"192.168.0.47" /syncfromflags:manual /update
```

Address of your Raspberry NTP server

```
w32tm /query /status
```

Now you should see your Raspberry as a Source

Configure Raspberry Pi as an NTP Server Using a USB GPS (without PPS)

If your **USB GPS does not support PPS**, it will still provide accurate time, but with slightly lower precision than a GPS-PPS setup. We will configure **gpsd** and **chrony** to sync time from the GPS NMEA sentences. Configure Raspberry Pi as a GPS-PPS NTP Server with Chrony

This guide configures the **Grove - GPS (Air530)** module on a **Raspberry Pi 4 (64-bit OS)** to act as a **high-precision NTP server** using **GPS and PPS (Pulse Per Second)**.

 No GPIO wiring required – just plug in the USB GPS.

 Remove all PPS-related configurations:

- Don't add dtoverlay=pps-gpio,gpiopin=18 to /boot/firmware/config.txt.
- Remove pps-gpio from /etc/modules.
- Don't use refclock PPS in chrony.conf.

 Ensure GPSD uses /dev/ttyUSB0 or /dev/ttyACM0 instead of /dev/serial0 /dev/pps0.

 Modify Chrony to rely only on GPS (SHM 0), since there's no PPS for precise synchronization.

Step 1: Connect the hardware

Air530 GPS Module to Raspberry Pi Wiring Table

Air530 GPS Module	Raspberry Pi (GPIO Header)	Pin Number	Function
VCC (3.3V or 5V)	3.3V or 5V	Pin 1 or Pin 2 (5V)	Power
GND	GND	Pin 6, 9, 14, 20, or 25	Ground
TX	GPIO 15 (RXD)	Pin 10	GPS → Pi Serial RX
RX	GPIO 14 (TXD)	Pin 8	Pi Serial TX → GPS
PPS	GPIO 18	Pin 12	Pulse Per Second (PPS) Timing

Plugin GPS USB dongle and check if it works

```
ls -l /dev/serial/by-id/
```

You should get something like that

```
lrwxrwxrwx 1 root root 13 Feb 28 10:24 usb-u-blox_AG_-_www.u-blox.com_u-blox_7_-_GPS_GNSS_Receiver-if00 -> ../../ttyACM0
```

Step 2: Install required packages

```
sudo apt update && sudo apt upgrade -y && sudo rpi-update -y && sudo apt install -y gpsd gpsd-clients chrony
```

Step 3: Verify GPS data

Test raw GPS data (/dev/ttyACM0 from above):

```
sudo cat /dev/ttyACM0
```

You should see **NMEA sentences** ($GPGGA, $GPRMC, etc.).

Press CTRL + C to stop.

Start gpsd manually for debugging:

```
gpsd -n /dev/ttyACM0 -F /var/run/gpsd.sock
```

Check GPS data using:

```
cgps -s
```

If you see satellite data, the GPS is working.

Step 4: Configure gpsd for automatic start

Edit GPSD settings:

```
sudo nano /etc/default/gpsd
```

Update the file:

```
GPSD_OPTIONS="-n"
DEVICES="/dev/ttyACM0"
GPSD_SOCKET="/var/run/gpsd.sock"
USBAUTO="true"
```

Save and restart GPSD:

```
sudo systemctl enable gpsd
sudo systemctl restart gpsd
```

Verify GPSD status:

```
systemctl status gpsd
```

Step 5: Configure Chrony to use GPS

Edit the Chrony configuration file:

```
sudo nano /etc/chrony/chrony.conf
```

Add these lines at the end Add IP range if you want to distribute this time):

```
# Use GPS as a time source (from gpsd shared memory)
refclock SHM 0 offset 0.5 delay 0.2 refid GPS prefer

# Allow network clients to sync time
allow 192.168.0.0/24Save (CTRL + X, then Y, then Enter).
```

Restart Chrony:

```
sudo systemctl restart chrony
sudo systemctl enable chrony
```

Verify Chrony GPS & PPS Sync

```
chronyc sources
```

Wait for 30 seconds

Example output:

```
#* GPS                           0   4     7    24   -255ms[ -250ms] +/-  101ms
```

GPS (#- GPS) is detectedand used.

Check system time:

```
timedatectl status
```

Look for System clock synchronized: yes.