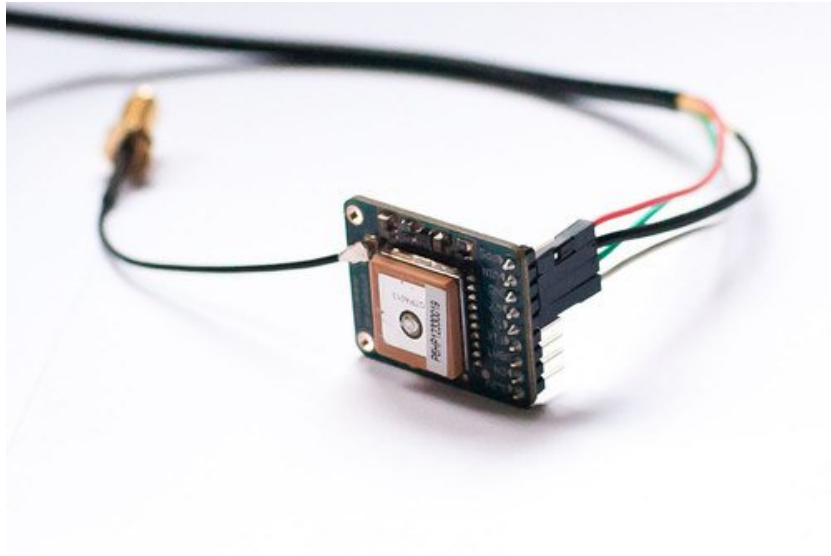




Adafruit Ultimate GPS on the Raspberry Pi

Created by Kevin Townsend

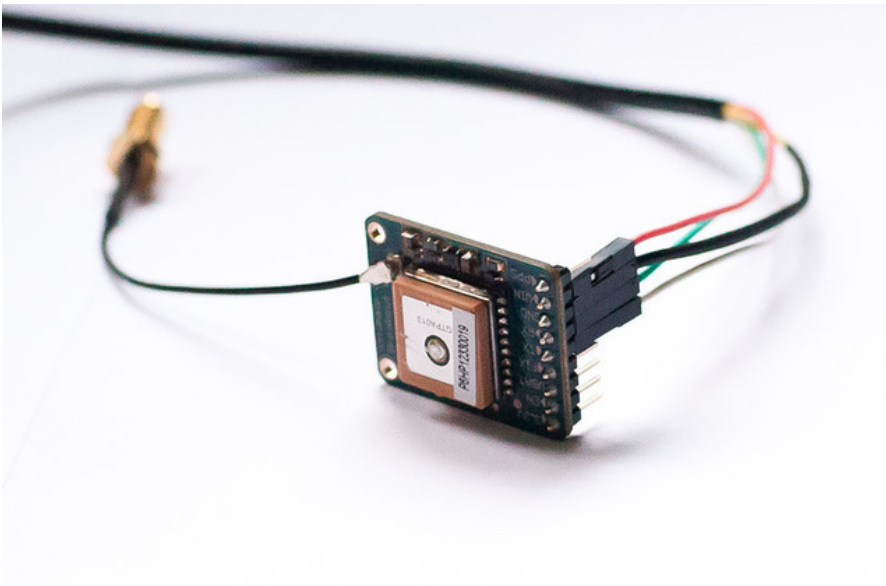


Last updated on 2016-07-08 01:22:35 AM EDT

Guide Contents

Guide Contents	2
Introduction	3
What you'll need:	3
Setting Everything Up	4
Hooking The Breakout Up	4
Setting up the USB Adapter	4
Installing a GPS Daemon (gpsd)	5
Raspbian Jessie systemd service fix	5
Testing it Out	5
Using your GPS	7
Using UART instead of USB	8
Step One: Edit /boot/cmdline.txt	8
Raspbian Wheezy only	8
Step Two: Edit /etc/inittab	8
Raspbian Jessie	9
Step Two:	9
Step Three: Raspberry Pi 3 Only	9
Step Four: Reboot your Pi	9
Step Five: Restart GPSD with HW UART	9
Further Resources	11

Introduction



How easy is it to get your Raspberry Pi eavesdropping on satellites 20,000 km up in the sky? Wonderfully easy thanks to Linux, and affordable thanks to Adafruit's [Ultimate GPS Breakout \(http://adafru.it/746\)](http://adafru.it/746)!

This quick learning guide will show you everything you need to do to add position tracking to your Pi project using the open source GPS daemon 'gpsd' and an inexpensive [USB to TTL adapter cable \(http://adafru.it/954\)](http://adafru.it/954) or via direct-wiring to the built-in Pi UART pins

What you'll need:

- A Raspberry Pi (<http://adafru.it/998>)
- An Ultimate GPS Breakout (<http://adafru.it/746>)
- A USB to TTL Adapter (<http://adafru.it/954>) Cable (or something compatible)

Don't forget to also read our Ultimate GPS tutorial which has a lot of information about this GPS module and datasheets/example code that you will find handy! (<http://adafru.it/aTI>)

Setting Everything Up

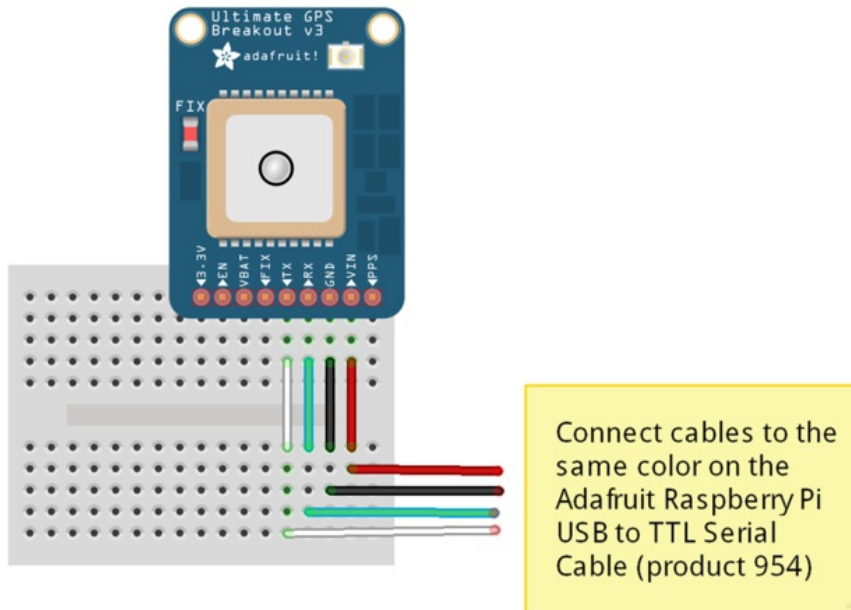
The easiest way to get start is to use an inexpensive [USB to TTL adapter cable](http://adafru.it/954) (<http://adafru.it/954>) with your GPS module.

You can of course use the HW UART directly on the Pi, but as you can see in this tutorial ([Freeing UART on the Pi](http://adafru.it/aWK) (<http://adafru.it/aWK>)) it's a bit more complicated, and there are no secondary consequences with the USB adapter.

This tutorial will assume that we are using the USB to TTL cable mentionned above, and that we are running on Occidentalis or Rasbian using the wonderfully painless [WebIDE](http://adafru.it/aQv) (<http://adafru.it/aQv>). Occidentalis & Rasbian already has the drivers for PL2303-based cables pre-installed, so you just need to plug it in and it should show up as `/dev/ttyUSB0`.

Hooking The Breakout Up

The first thing you'll need to do is to hook your Ultimate GPS Breakout up to the Pi with the adapter cable. The following diagram shows you what you need to know, essentially just connecting the cables of the same color together.



While the module on the Ultimate GPS Breakout has an exceptionally sensitive antenna and may work indoors as is, you may want to pick up an [external GPS Antenna](http://adafru.it/960) (<http://adafru.it/960>) and an [SMA to u.FL adapter cable](http://adafru.it/851) (<http://adafru.it/851>) if this is for indoor use. This will allow you to keep the Pi and GPS breakout indoors, but run the antenna out a window or at least near one for improved reliability and signal integrity.

Setting up the USB Adapter

Once you plug the USB cable into the Pi, the adapter should show up as `/dev/ttyUSB0` (though the '0' may be different if you have other ttyUSB adapters present).

You can see a list of all ttyUSB devices by entering the following into the console (I'm using the 'terminal' feature in Adafruit's browser-based WebIDE here for convenience sake!):



If you have any problems, you can enter the following command to see the USB devices on your Pi:



Which should show you the USB adapter (Prolific PL2303), as follows:



```
webide@raspberrypi /usr/share/adafruit/webide/repositories $ sudo lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 005: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
```

If you just want to do a quick check to see what data is coming out of the GPS, you can enter the following command, following by **CTRL+C** to quit:

Installing a GPS Daemon (gpsd)

The next step is installing some software on your Raspberry Pi that understands the serial data that your GPS module is providing via `/dev/ttyUSB0`. Thankfully other people have already done all the hard work for you of properly parsing the raw GPS data, and we can use (amongst other options) a nice little package named 'gpsd', which essentially acts as a layer between your applications and the actual GPS hardware, gracefully handling parsing errors, and providing a common, well-defined interfaces to any GPS module.

To install gpsd, simply run the following commands from the console:

... which will install the required packages (an internet connection will be required for this step!)

Raspbian Jessie systemd service fix

Note if you're using the Raspbian Jessie or later release you'll need to disable a systemd service that gpsd installs. This service has systemd listen on a local socket and run gpsd when clients connect to it, however it will also interfere with other gpsd instances that are manually run (like in this guide).

You will need to disable the gpsd systemd service by running the following commands:

Should you ever want to enable the default gpsd systemd service you can run these commands to restore it (but remember the rest of the steps in this guide won't work!):

After disabling the gpsd systemd service above you're ready to try running gpsd manually. Now run the following command to manually start gpsd and point it at the GPS breakout on the USB serial adapter port:

... which will point the gps daemon to our GPS device on the USB to TTY adapter cable (simply substitute `'/dev/ttyUSB0'` for another destination if required).

Testing it Out

After a few seconds, gpsd should open up the proper socket and if the GPS is locked we should be able to get some data from the GPS module.

To test this, we can use the following command:

If you have a fix, you'll see something like the following information in the terminal window:

```
Time: 2013-01-24T08:56:30.000Z || PRN: Elev: Azim: SNR: Used: |
Latitude: || 11 80 287 37 Y |
Longitude: || 1 59 288 26 Y |
Altitude: 215.6 ft || 32 53 207 29 Y |
Speed: 0.0 mph || 19 52 153 24 Y |
Heading: 127.3 deg (true) || 14 34 076 45 Y |
Climb: 0.0 ft/min || 39 29 150 30 Y |
Status: 3D FIX (7 secs) ||
```

If you have any problems and cgps always displays 'NO FIX' under status and then aborts after a few seconds, you may need to restart the gpsd service. You can do that via the following commands:

```
sudo systemctl restart cgpsd
sudo systemctl status cgpsd
```

Using your GPS

Now that you're GPS is up and running, and `gpsd` is playing nice with it, it's time to do something with the data!

The easiest way to get started is using the WebIDE and a bit of python code to access gpsd.

Create a new file in the [WebIDE \(http://adafru.it/aQv\)](http://adafru.it/aQv), and then add and run the following code:

```
from datetime import datetime

# Run on port 2047 (port of broadcast)
session = ops.ops.broadcast("2047")
session.streamops.WATCH_ENABLE |= ops.WATCH_NEWSTIME

while True:
    try:
        report = session.next()
        # Wait for a TTY report and display the current time
        # To see all report data, uncomment the line below
        # print report
        if report.class == TTY:
            if hasattr(report, "line"):
                print report.line
    except KeyboardInterrupt:
        pass
    except KeyboardInterrupt:
        quit()
    except KeyboardInterrupt:
        continue
except None:
```

This should give you something similar to the following (with an update every second or so):

```

Cwebide@raspberrypi /usr/share/adafruit/webide/repositories/my-pi-projects/gptest $ sudo python gpstest.py
2013-01-24T10:39:27.000Z
2013-01-24T10:39:28.000Z
2013-01-24T10:39:29.000Z
2013-01-24T10:39:30.000Z
2013-01-24T10:39:31.000Z

```

Looking for position data rather than just the timestamp? Essentially, all you have to do is parse the 'report' data following the example above.

To see what data is available, you can uncomment the 'print report' line, and then just look at the different values and class names and pick and choose whatever you want.

For example, you could use the following code to get the current speed using the TPV class:

```

    @property
    def report_speed(self):
        """
        pin/report_speed: gps.MPS_TO_KPH
        """

```

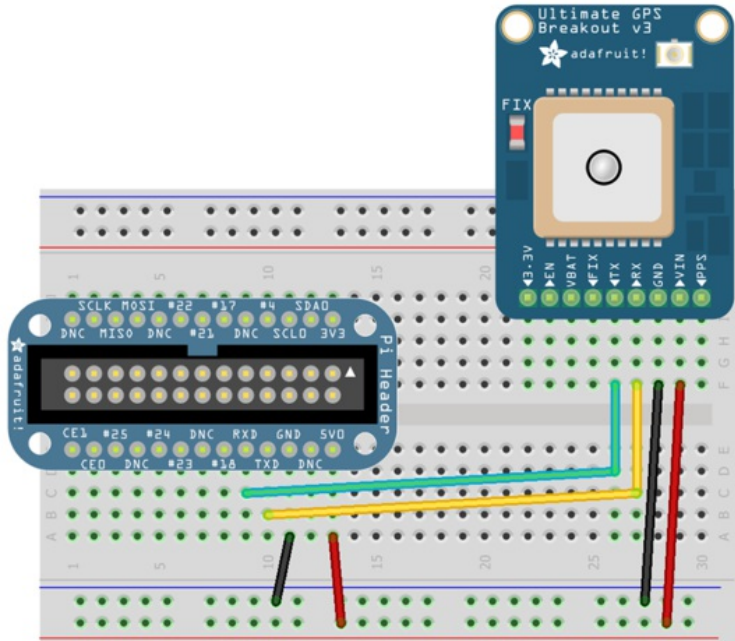
That's it! It's pretty painless, and now it's up to you to figure out what you want to do with you latitude, longitude, date and time, speed, altitude, etc.!

Using UART instead of USB

If you wish to use HW UART instead of the USB cable, it's perfectly possible ... you just need to do a bit more work to free the UART up on your Pi.

To get started, hook the GPS module up to your Pi as follows, cross-connecting the TX and RX pins (TX on one device goes to RX on the other and vice versa), and supply 5V from the Pi to the VIN pin on the GPS module:

We designed the Ultimate GPS with a built-in regulator, so even if it's powered with 5V, the signal levels are still 3.3V - safe for your Pi!



Step One: Edit /boot/cmdline.txt

Next, enter the following command from the command line:

```
sudo nano /boot/cmdline.txt
```

And change:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

to:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

(eg, remove `console=ttyAMA0,115200` and if there, `kgdboc=ttyAMA0,115200`)

Note you might see `console=serial0,115200` or `console=ttyS0,115200` and should remove those parts of the line if present.

Raspbian Wheezy only

Step Two: Edit /etc/inittab

From the command prompt enter the following command:

```
sudo nano /etc/inittab
```


And change:

```
#Spawn a getty on Raspberry Pi serial line  
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

to:

```
#Spawn a getty on Raspberry Pi serial line  
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

That is, **add a #** to the beginning of the line!

Raspbian Jessie

Step Two:

For the **Raspberry Pi 1 or 2** (but **NOT** the 3!) Run the following two commands to stop and disable the tty service:

```
sudo systemctl stop tty  
sudo systemctl disable tty
```

However for the **Raspberry Pi 3** you need to use the **/dev/ttyS0** port since that is what is normally connected to the GPIO serial port pins. Use these two commands instead:

```
sudo systemctl stop ttyS0  
sudo systemctl disable ttyS0
```

Step Three: Raspberry Pi 3 Only

For the Raspberry Pi 3 you need to explicitly enable the serial port on the GPIO pins. The reason for this is a change with the Pi 3 to use the hardware serial port for Bluetooth and instead use a slightly different software serial port for the GPIO pins. A side effect of this change is that the serial port will actually change speed as the Pi CPU clock throttles up and down--this will unfortunately cause problems for most serial devices like GPS receivers!

Luckily there's an easy fix detailed in [this excellent blog post \(http://adafru.it/oFE\)](http://adafru.it/oFE) to force the Pi CPU into a fixed frequency which prevents speed changes on the serial port. The Pi might not perform as well but it will have a stable serial port speed.

To make this change edit the **/boot/config.txt** file by running:

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

At the very bottom of the file add this on a new line:

```
enable_uart=1
```

Save the file (press **Ctrl-O**, then enter) and exit (press **Ctrl-X**). You're all set!

Step Four: Reboot your Pi

After rebooting the Pi for the above changes to take effect, you can proceed with running gpsd ...

Step Five: Restart GPSD with HW UART


Restart gpsd and redirect it to use HW UART instead of the USB port we pointed it to earlier. Simply entering the following two commands.

For the **Raspberry Pi 1 or 2** (but **NOT** the 3!) run these commands:


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

And for the **Raspberry Pi 3** run these commands to use the different serial port:

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



As with the USB example, you can test the output with:



Further Resources

Don't forget to also read our bigger tutorial on the Ultimate GPS which has lots more details, datasheets and examples for setting the sentences, update rate, etc! (<http://adafru.it/aTI>)

The following tutorials may be useful to you if you want to dig into this a bit further, and do something a bit more advanced with your GPS data:

- [GETTING GPS TO WORK ON A RASPBERRY PI](http://adafru.it/aWL) (<http://adafru.it/aWL>).
- [GPSD Client How-To](http://adafru.it/aWM) (<http://adafru.it/aWM>) ... including examples in C, C++ and [Python](http://adafru.it/aWM) (<http://adafru.it/aWM>)
- The official [GPSD project pages](http://adafru.it/aWN) (<http://adafru.it/aWN>)
- [A nice writeup of using GPSd with python using threads to make it faster](http://adafru.it/cGM) (<http://adafru.it/cGM>)

Doing something fun with GPS and tracking data? Be sure to post about it in the [Adafruit forums](http://adafru.it/forums) (<http://adafru.it/forums>) so everyone else can get inspired by it!