Raspberry Pi APRS Tracker

Version 1.1 – November 2014

The Linux version of Dire Wolf can be used as a tracker by enabling an optional feature to access GPS data. This is not enabled by default because most systems probably won't have the necessary files installed.

This document is targeted specifically for the Raspberry Pi. Other than the details about the serial port, everything else should be applicable to other Linux systems.

First install Dire Wolf as detailed in the other Raspberry Pi APRS document and make sure it is working properly before enabling this additional feature.

Step 1: Install the GPS support software.

There are three parts to this:

- **gpsd** daemon which communicates with the GPS receiver.

```
sudo apt-get install gpsd
```

- Test programs.

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

- Other files for application development. (e.g. /usr/include/gps.h)

```
sudo apt-get install libgps-dev
```

Step 2: Connect GPS receiver.

There are different ways to attach a GPS receiver.

- The easy way is to use one with a USB interface. Plug it in and it should show up with a device name like /dev/ttyUSB0.

- A GPS module with a 3.3 volt logic level interface can be connected directly to the Raspberry Pi's UART. Additional configuration steps must be taken to disable console output to the serial port. In this case, the device name is /dev/ttyAMA0.

  Complete instructions can be found here:

  http://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/introduction

  When using the UART, be sure to edit two files as described in the reference above.

  - **/boot/cmdline.txt** -- change from:

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

    to: *(remove part in red above mentioning ttyAMA0)*

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

  - **/etc/inittab** -- change from:

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

    to: *(comment out one line)*

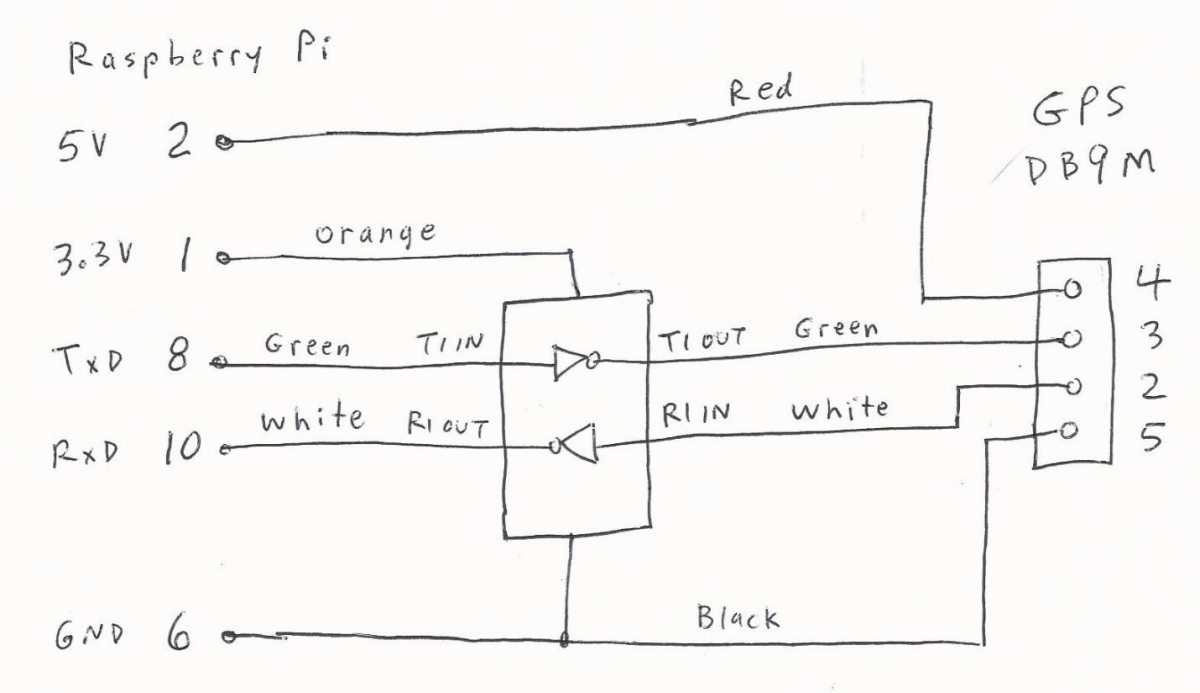
    #Spawn a getty on Raspberry Pi serial line
    #T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

- RS-232 interface

  This is similar to the previous step except you need to add a logic level to RS-232 level converter.

  WARNING: Do not attempt to connect a GPS receiver with RS-232 voltage levels directly to the Raspberry Pi GPIO pins. It will probably destroy your Raspberry Pi. An RS-232 to 3.3v logic converter, must be used.

  The SparkFun MAX3232 Breakout BOB-11189, https://www.sparkfun.com/products/11189, can be wired as shown below.



5 volts, to power the GPS, is supplied through pin 4. This is the same convention commonly used by other trackers such as OpenTrack and Tiny Trak4 so the same GPS receiver should be interchangeable between them.

It is also possible to obtain the RS-232 level converter and connector preassembled as a single unit. Something like the NKC Electronics 0609456348584 http://www.amazon.com/dp/B0088SNIOQ/ref=dra_a_cs_ss_hn_it_P1250_1000 looks like it would be suitable but I haven't actually tried it. Just be sure it has a male connector, is wired as DTE, and the level converter chip is capable of 3.3 volt operation. I'd like to hear from anyone who tried this specific item.

Reboot the system. (Type "sudo reboot.")

Start up the gpsd daemon with the appropriate device name:

```
sudo killall gpsd
sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
```

or

```
sudo killall gpsd
sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock
```

For testing you can run either "`cgps -s`" (text only) or "**`xgps`**" (with graphical display of satellite positions). Be sure that you are correctly receiving your location from the satellites.

If you run into problems, this troubleshooting information might be helpful.
http://catb.org/gpsd/troubleshooting.html

Step 3: Configure for automatic start up on reboot.

The one thing left out of the instructions from Ada Fruit is automatic start up on reboot. Run this command and answer the questions:

```
sudo dpkg-reconfigure gpsd
```

Use "-n" when it asks for options.

Step 4: Optionally set clock from GPS receiver.

The RPi does not have a battery powered real time clock. When it is powered up, it doesn't know the current time and date. This is usually obtained from an NTP server over the network. A mobile tracking device probably won't have reliable Internet access. The system time can be set from your GPS receiver.

Edit /etc/ntp.conf and add lines like this:

```
server 127.127.28.0 4
fudge 127.127.28.0 time1 0.340 refid GPS

server 127.127.28.1 prefer
fudge 127.127.28.1 refid GPS1
```

After rebooting, and waiting a little while, you can tell if it is working by using the **"ntpq -p"** command. When obtaining time from Internet NTP servers, you will see something resembling this:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*tock.usshc.com  .GPS.            1 u   46   64  377   47.884   18.724  64.903
+estatico.iloves 216.129.104.26   3 u   41   64  377   94.824   18.614  66.610
+79.132.237.6.st 193.190.230.65   2 u   10   64  377  117.697   17.355  50.584
+sisdb01.muskego 173.255.240.184  3 u    7   64  377   44.962   19.065  35.800
```

The lines with * and + at the beginning indicate time sources considered to be good.

When obtaining time from your GPS, it will look more like this:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+96.44.142.5     18.26.4.105      2 u   46   64   17   69.485   -0.346  61.541
+ec2-54-235-96-1 152.2.133.53     2 u   43   64   17   27.297    4.169  32.766
*lithium.constan 128.4.1.1        2 u   46   64   17   16.853   -1.192  24.474
-50.7.0.66       128.138.141.172  2 u   42   64   17   39.321    5.043   0.575
*SHM(0)          .GPS.            0 l   43   64  377    0.000    2.589   4.256
 SHM(1)          .GPS1.           0 l    -   64    0    0.000    0.000   0.000
```

Notice how the line with SHM(0) .GPS. has * in front of it.

More detailed information can be found here: http://www.satsignal.eu/ntp/Raspberry-Pi-NTP.html

Step 5: Rebuild Dire Wolf with support for GPS.

Go to the source file directory. The exact name will vary depending on version. For example:

```
cd /home/pi/direwolf-1.1
```

Edit Makefile.linux and look for a section similar to this:

```
# Uncomment following lines to enable GPS interface.
CFLAGS += -DENABLE_GPS
LDLIBS += -lgps
```

Remove any # characters from the beginning of the CFLAGS and LDLIBS lines. Then type:

```
make -f Makefile.linux clean
make -f Makefile.linux
sudo make -f Makefile.linux install
```

Step 6: Configure beaconing.

After rebuilding, as instructed above, a new beacon configuration command, TBEACON is available. It is much like PBEACON and OBEACON except it takes the location, altitude, speed, and direction from the GPS receiver.

```
TBEACON EVERY=4 SYMBOL=car
```

That example will transmit at a set interval, every 4 minutes. It might be too infrequent for a vehicle moving quickly. It might generate too many redundant transmissions for something moving slowly. SmartBeaconing™ can be used to adjust the timing based on speed and changes in direction. It's the same technique used by Kenwood, Yaesu/Standard, and in many other tracker applications. A configuration option like this will override any fixed interval in specified in TBEACON.

```
SMARTBEACONING  60 2:00  5 15:00  0:15 30 255
```

What do the numbers mean?

- For speeds above 60 MPH, a beacon will be sent every 2 minutes.
- For speeds below 5 MPH, a beacon will be sent every 15 minutes.
- For speeds in between, a rate proportionally in between will be used.

Additional beacons will be sent more frequently when direction changes significantly.

- Send no more frequently than 15 seconds apart.
- Send if direction has changed more than 30 degrees at high speed.
- Requires sharper turns at lower speeds.

More details can be found in these references or just Google for APRS SmartBeaconing to find many discussions and recommendations.

http://www.hamhud.net/hh2/smartbeacon.html

http://info.aprs.net/index.php?title=SmartBeaconing

Troubleshooting

1. Build errors.

If you see compiling or linking errors, mentioning missing **gps.h** or **libgps**, you didn't install the GPS software in Step 1.

2. Erratic GPS data.

Try running "cgps -s" (text output) or "xgps" to look at GPS data.

Use "-d t" option on direwolf command for tracker debugging information.