

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

Updated: Monday, September 4, 17

I did not look for a long time to this article, until I recently got a request to help another radio ham.

Obviously, the old script and commands no longer work. And since there was a new Raspberry PI available, as well as a cheap 7inch tablet, I decided to give it a try over the week-end.

This document now describes the both the installation of the basic Dantracker from Dan Smith as well as the extended version by Basil, N7NIX. I have checked the latter, and may do so later also to Dan's basic version.

For N7NIX version, there will be a fully automatic installation script for the Raspian Stretch distro available on my website later on.

AUTO INSTALLATION



Login **as root** to your RPI, change into the /home/pi directory (create if you don't have it) and execute the following:

Install some prerequisite packages

```
sudo apt-get install build-essential libx11-dev zlib1g-dev libncurses5-dev autoconf autogen libtool sudo libgps-dev bridge-utils git-core libnl-3-dev libnl-genl-3-dev libjson0 libjson0-dev nodejs screen git python-serial libgtk2.0-dev gtk+-2.0 gcc pkg-config imagemagick automake autoconf libtool cvs curl libncurses-dev libssl-dev locate dnsmasq rfc2530
```

next install the auto install script:

```
wget http://www.pa0esh.com/svn/install\_dantracker.sh
```

Then execute

```
chmod +x install_dantracker.sh && ./install_dantracker.sh
```

Here you see my "test" station: Raspberry PI – 4 port powered hub with WiPi, GPS mouse and flash-disk for OS connected, TNC-X connected to the old Icom IC200H.

I had a small LCD screen to go along in case I used the original Dantracker (co-installed). However, the screen died on me and I only use the N7NIX version.

This should have worked and after you restart, the dantracker should be working.

If not follow the individual steps below and see where the installations crashes. From there on it is most of the time possible to detect the reason for the crash and to solve it.

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

FRESH INSTALLATION FOR RASPBERRY PI MOD II – RASPIAN STRETCH

Take (minimal) a **8Gb** sd disk of good quality.....

Install Raspbian Stretch (either version) as per Raspberry PI Download -

<http://www.raspberrypi.org/downloads/>

I advice for Mac and Windows users <https://etcher.io/#downloads> to put the Raspbian image on an SD card.

Or check the site <https://www.raspberrypi.org/downloads/raspbian/> and follow the instructions.

Now start your Raspberry. If you use a display attached to your Raspberry PI, follow the on-screen instructions and finish off.

If you use ssh, find the address of the raspberry (I use LanScan) and ssh into it (ssh pi@x.x.x.x). Please note that a virgin image does not have ssh enabled. So you must first use an attached screen en keyboard.

Make sure to change the PI password.... And create also a root password.
Then reboot

Log-in as user pi

```
✚ ssh pi@x.x.x.x -X (x.x.x.x is the ip address of my RPI)
```

If you go for the Dantracker N7NIX version, you do need the Raspberry to convert into an Access Point in your car, unless you have already an access point up and running inside.....

Using an iPhone hotspot utility, is one option, but not stable (so far for me)

Here is a solution to install **hostapd** with supporting **dhcp** en masquerading tools – **dnsmasq**.

You need internet access for this step so make sure that Ethernet connection is up!

SETUP THE PRE REQUISITES

To compile Dantracker and install supporting software, we are first installing all prerequisite packages.

```
✚ sudo apt-get install build-essential libx11-dev zlib1g-dev libncurses5-dev autoconf  
autogen libtool sudo libgps-dev bridge-utils git-core libnl-3-dev libnl-genl-3-dev libjson0  
libjson0-dev nodejs screen git python-serial libgtk2.0-dev gtk+-2.0 gcc pkg-config  
imagemagick automake autoconf libtool cvs curl libncurses-dev libssl-dev locate  
dnsmasq rfc2530
```

Note: since this document was written in 2015, some software has been updated and you may have to install a newer version. In that case the software will not install at all, and indicate to you which part is not working.

INSTALL HOSTAPD & DNSMASQ

PACKAGES

The first step is to install the required packages:

```
✚ sudo apt-get install dnsmasq hostapd
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

I'll go into a little detail about the two:

hostapd - This is the package that allows you to use the built in WiFi as an access point
dnsmasq - This is a combined DHCP and DNS server that's very easy to configure
If you want something a little more 'heavyweight', you can use the isc-dhcp-server and bind9 packages for DHCP and DNS respectively, but for our purposes, dnsmasq works just fine.

CONFIGURE YOUR INTERFACES

The first thing you'll need to do is to configure your wlan0 interface with a static IP. If you're connected to the Pi via WiFi, connect via Ethernet / serial / keyboard first.

In newer Raspian versions, interface configuration is handled by dhcpcd by default. We need to tell it to ignore wlan0, as we will be configuring it with a static IP address elsewhere. So open up the dhcpcd configuration file with

```
+ sudo nano /etc/dhcpcd.conf
and add the following line to the bottom of the file:
+ denyinterfaces wlan0
```

Note: This must be ABOVE any interface lines you may have added!

Now we need to configure our static IP. To do this open up the interface configuration file with `sudo nano /etc/network/interfaces` and edit the wlan0 section so that it looks like this:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 172.24.1.1
    netmask 255.255.255.0
    network 172.24.1.0
    broadcast 172.24.1.255
# wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Restart dhcpcd with `sudo service dhcpcd restart` and then reload the configuration for wlan0 with

```
+ sudo ifdown wlan0; sudo ifup wlan0.
```

CONFIGURE HOSTAPD

Next, we need to configure hostapd.

Create a new configuration file with

```
+ sudo nano /etc/hostapd/hostapd.conf
with the following contents:
```

```
# This is the name of the WiFi interface we configured above
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Pi3-AP
# Use the 2.4GHz band
hw_mode=g
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
ignore_broadcast_ssid=0
# Use WPA2
wpa=2
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=raspberry
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

We can check if it's working at this stage by running `sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf`. If it's all gone well thus far, you should be able to see to the network Pi3-AP! If you try connecting to it, you will see some output from the Pi, but you won't receive an IP address until we set up dnsmasq in the next step. Use Ctrl+C to stop it.

Note: with an old WiFi adapter you may have to delete the WMM lines. (I did require that with an early WiPi interface)

We aren't quite done yet, because we also need to tell hostapd where to look for the config file when it starts up on boot. Open up the default configuration file with

```
🔧 sudo nano /etc/default/hostapd
```

and find the line `#DAEMON_CONF=""` and replace it with `DAEMON_CONF="/etc/hostapd/hostapd.conf"`.

CONFIGURE DNSMASQ

The shipped dnsmasq config file contains a wealth of information on how to use it, but the majority of it is largely redundant for our purposes. I'd advise moving it (rather than deleting it), and creating a new one with

```
🔧 sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

```
🔧 sudo nano /etc/dnsmasq.conf
```

Paste the following into the new file:

```
interface=wlan0          # Use interface wlan0
listen-address=172.24.1.1 # Explicitly specify the address to listen on
bind-interfaces          # Bind to the interface to make sure we aren't sending
things elsewhere
server=8.8.8.8           # Forward DNS requests to Google DNS
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
domain-needed      # Don't forward short names
bogus-priv         # Never forward addresses in the non-routed address
spaces.
dhcp-range=172.24.1.50,172.24.1.150,12h # Assign IP addresses between
172.24.1.50 and 172.24.1.150 with a 12 hour lease time
```

SET UP IPV4 FORWARDING

One of the last things that we need to do before we send traffic anywhere is to enable packet forwarding.

To do this, open up the sysctl.conf file with

```
✚ sudo nano /etc/sysctl.conf
```

and remove the # from the beginning of the line containing net.ipv4.ip_forward=1. This will enable it on the next reboot, but because we are impatient, activate it immediately with :

```
✚ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

We also need to share our Pi's internet connection to our devices connected over WiFi by the configuring a NAT between our wlan0 interface and our eth0 interface. We can do this using the following commands:

```
✚ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
✚ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED
-j ACCEPT
✚ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

However, we need these rules to be applied every time we reboot the Pi, so run

```
✚ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

to save the rules to the file /etc/iptables.ipv4.nat. Now we need to run this after each reboot, so open the rc.local file with

```
✚ sudo nano /etc/rc.local
```

and just above the line **exit 0**, add the following line:

```
iptables-restore < /etc/iptables.ipv4.nat
```

WE'RE ALMOST THERE!

Now we just need to start our services:

```
✚ sudo service hostapd start
✚ sudo service dnsmasq start
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

And that's it! You should now be able to connect to the internet through your Pi, via the on-board WiFi!

To double check we have got everything configured correctly, reboot with
`sudo reboot.`

SET UP WLAN0 FOR STATIC IP

If you happen to have wlan0 active because you set it up, run

```
sudo ifdown wlan0
```

There's no harm in running it if you're not sure. Never mind the warning.

Next we will set up the wlan0 connection to be static and incoming. Run

```
sudo nano /etc/network/interfaces
```

to edit the file

Find the line `auto wlan0` and **add a # in front of the line**, and in front of every line afterwards. Depending on your existing setup/distribution there might be more or less text and it may vary a little bit

Add the lines:

```
iface wlan0 inet static
address 192.168.42.1
netmask 255.255.255.0
```

After ***allow hotplug wlan0***

Save the file

Assign a static IP address to the Wi-Fi adapter by running

```
apt-get install rftkill
sudo ifconfig wlan0 192.168.42.1
```

Now make sure that it does not lose this static ip number during startup.

Check that is installed

```
apt-get install ifplugd
```

In `/etc/default/ifplugd`, the default configuration is normally the following:

```
INTERFACES="auto"
HOTPLUG_INTERFACES="all"
ARGS="-q -f -u0 -d10 -w -I"
SUSPEND_ACTION="stop"
```

Simply change it to this (even if the file looked different as above):

```
INTERFACES="eth0"
HOTPLUG_INTERFACES="eth0"
ARGS="-q -f -u0 -d10 -w -I"
SUSPEND_ACTION="stop"
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

This will ensure that wlan0 will not lose its static IP address that's configured in /etc/network/interfaces when wlan0 goes up.

Now reboot the Raspberry PI and check if you can access the Wi-Fi and that you have internet connectivity available.

CONTINUE WITH INSTALLING DANTRACKER

Download either the original dantracker source

```
# cd /home/pi
# git clone https://github.com/kk7ds/dantracker
```

or the N7NIX branch

```
# git clone https://github.com/n7nix/dantracker
# cd dantracker
```

Download and install libfab

```
# sudo wget http://pakkettiradio.net/downloads/libfab/1.5/libfab-1.5.tar.gz
# tar xvzf libfab-1.5.tar.gz
# sudo cp libfab-1.5/src/fap.h /usr/local/include/
# cd libfab-1.5
```

Now apply the fap patch:

```
# patch -p2 < /<path_to_tracker_src>/fap_patch.n7nix
To find the path to the dantracker source, type pwd. Since you are in the libfab-1.5
directory (you should be !), the path_to_tracker_src is what is shown with pwd, less
libfab-15.
Example: pwd -> /home/pi/dantracker/libfab-1.5, then the patch command is
patch -p2 < /home/pi/dantracker/fap_patch.n7nix
# sudo cp src/fap.h /usr/local/include/
# ./configure
# make && sudo make install
# cd .. (go back to the dantracker directory)
```

Install iniparser - <http://ndevilla.free.fr/iniparser/>

```
# git clone http://github.com/ndevilla/iniparser.git
# cd iniparser
# cp src/iniparser.h /usr/local/include
# cp src/dictionary.h /usr/local/include
# make
# cp libiniparser.* /usr/local/lib
# ldconfig
# cd .. (back to the standard dantracker directory)
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

Note – The next part applies to the N7NIX branch only

===== BOF N7NIX INSTALLATION RASPBERRY PI – DEBIAN JESSIE =====

AX25 setup

The N7NIX branch of the Dantracker works best when ax25 is used.

This is a link to a very good ax25 installation instruction for the Raspberry PI.: <http://k4gbb.no-ip.org/docs/Raspberry.html>

Follow those instructions, change the axports and ax25d.conf files with your own data, as well as the IFUPD file, and you're good to go on ax25.

Execute the following commands: (they make take a looooong time to finish)

```
+ wget http://k4gbb.no-ip.org/docs/scripts/Instax25.new
+ sudo chmod +x Instax25.new
+ sudo ./Instax25.new
```

Now adapt the settings for your own applications, following the next guidelines:

AXPORTS

Let's start with the axports file. This is the base for the configuration of all of the ax.25 devices. Edit the axports file located in /etc/ax25/ with the command

```
+ sudo nano /etc/ax25/axports
```

Here is an example:

```
# /etc/ax25/axports
# please change the NOCAL INTO TOUR OWN CALL SIGN
# The format of this file is:
#
# name callsign speed paclen window description
#
0 NOCAL-13 1200 255 7 144.800 MHz (1200 bps)
1 NOCAL-15 38400 255 7 TNOS/Linux (38400 bps)
```

Note:

- Use Numbers for Port names. You can give them names like radio or UHF1. That means more key strokes when you need to specify a Port name.
- The Call-ssid does not have a valid HAM Call. The ssid should be in the range of 0 - 15. This is a default and OK for ports that will never send over a radio link.
- The PacLen and the Window or MaxFrame values are set at max. All of these values are Defaults, including the baud rate, and are usually modified by the using Applications.
- DO NOT leave any Blank Lines !!!

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

AX25D.CONF

Create or edit `/etc/ax25/ax25d.conf` using the command

```
sudo nano /etc/ax25/ax25d.conf
```

It should look something like this.

```
# /etc/ax25/ax25d.conf
[NOCAL-14 via 0 ]
NOCALL * * * * * L
default * * * * * - rmsgw /usr/local/bin/rmsgw rmsgw -P 0 %U
# End /etc/ax25/ax25d.conf
```

Create a port

Connect your TNC to the Raspberry PI and check what device it is given. I hooked up a TNC-X with USB and it showed me `/dev/ttyUSB0`.

If you have various usb devices attached, such as a gps, keyboard etc, do the following:

Execute `lsusb`

```
root@dantracker:/home/pi# lsusb
Bus 001 Device 007: ID 0461:0010 Primax Electronics, Ltd HP Keyboard
Bus 001 Device 006: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC
Bus 001 Device 005: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Bus 001 Device 006 is my TNC-X. Remove the connection, check again and if it disappeared, then you know it for sure.

Next execute `dmesg |grep USB Serial` to find what devices are created:

```
root@dantracker:/home/pi# dmesg|grep 'ttyUSB'
[ 9.073821] usb 1-1.3: cp210x converter now attached to ttyUSB0
[ 9.086930] usb 1-1.4: FTDI USB Serial Device converter now attached to ttyUSB1
```

The `cp210x` is my GPS mouse and the TNC is attached to `ttyUSB1` which uses the FTDI chip

Now execute:

```
sudo /usr/local/sbin/kissattach /dev/ttyUSB0 0 44.128.1.10
```

My result:

```
# /usr/local/sbin/kissattach /dev/ttyUSB1 0 44.128.1.10
AX.25 port 0 bound to device ax0
```

Software requires EACH ax25 device to have a unique ip address. Do not use the address of your server or the loop back address (127.0.0.1). It is safe to use Amprnet addresses in the range of 44.128.100.0 to 44.128.0.0.

If it does not return an error message, then the next step is to set the kiss parameters:

```
sudo /usr/local/sbin/kissparms -p 0 -r 128 -s 10 -t 250
```

Where :

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

-p is the Port name from axports,
-r is the Persistence value,
-t is the TxDelay

Testing Ax Ports

Stop a minute and test your device!

Be Sure that your TNC is in the KISS mode.

NOTE: The ax25-tools package from my source still uses the old file names for call and listen. They are the same as axcall and axlisten.

Calibrate

Use the adapted version by K4GBB for the Raspberry PI
You can use it to see if you have control of the TNC's PTT.

Download calibrate_pi and place it in /usr/local/sbin/.

```
+ wget http://k4gbb.no-ip.info/docs/rpi/calibrate_pi
+ cp calibrate_pi /usr/local/sbin/calibrate_pi
```

Use the portname from the axports file.

```
+ calibrate 0
```

This will send a full frame (256 chars) of alternating 1s & 0s to port 0.

Press Return to stop the test. The PTT will stay active until the TNC's transmit buffer is emptied.

If you get a error message about frame length being too long - check your axports file. The value for PaLen must be 256 and MaxFrames 7.

Listen

Use (ax)listen to check the receiver monitoring process.

```
+ listen -cart
```

The screen will go blank and print received frames as the are decoded by the TNC.

Use Ctl-C to close the listen screen.

Call

The (ax)call command can be used to check the port's ability to connect and pass data. Call is a simple Term program.

The syntax is call <port> <call>v<path>

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
+ call 0 pa0esh-1 v pa0esh-2 (use your own call – it's just for testing)
```

NODE JS & NPM INSTALLATION

Install node.js (npm will be installed at the same time for that matter)

Note: As described at the beginning of this article, this fsection related to the installation of Node.js requires a Pi system based on the newer ARMv7 or ARMv8 chip such as the Pi 2 or Pi 3. NodeSource provides Node.js binaries for these newer ARMv7+ architectures, but not for Raspberry Pi systems based on the older ARMv6 architecture such as the Raspberry Pi Model B/B+ or the Raspberry Pi Zero.

Read the writing carefully on your Raspberry Pi circuit board to confirm is says something like “Raspberry Pi 3 Model B” or “Raspberry Pi 2 Model B”. If in doubt, run the following command in the terminal:

```
+ uname -m
```

If the result returned starts with “armv6”, you are running a Raspberry Pi based on the older ARMv6 chipset and the next Node.js installation step will not work; otherwise, you are ready for the next step.

```
+ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -  
+ sudo apt-get install -y nodejs  
+ sudo ldconfig  
+ cd .. (back to the dantracker directory)  
+ Now test the version with node -v & npm -v
```

My results (September 2017):

```
v8.4.0  
5.3.0
```

INSTALL JSON-C.

```
+ git clone https://github.com/json-c/json-c.git  
+ cd json-c  
+ autoconf -i  
+ sh autogen.sh  
+ ./configure  
+ make && make install  
+ sudo ldconfig  
+ cd .. (back to the dantracker directory)
```

DOWNLOAD JQUERY

Check their website for the latest version - <http://jquery.com/>
At this date (September 2017), the latest version was jquery-3.2.2

```
+ npm install jquery  
+ cd jquery && npm run build
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

Since it is the latest revision of this jquery, you may have to adapt the html files in /usr/share/dantracker. See their website for more information.

Now install the following nodes for npm:

```
+ sudo npm install -g node-gyp
+ sudo npm install -g socket.io
+ sudo npm install -g ctype
+ sudo npm install -g iniparser
+ sudo npm install -g websocket
+ sudo npm install -g connect
```

Whenever you get the following warnings, they are just that, warnings, and the installation should pose no problem...

```
npm WARN package.json github-url-from-git@1.1.1 No repository field.
npm WARN package.json assert-plus@0.1.2 No repository field.
npm WARN package.json ctype@0.5.2 No repository field.
```

Change back into the dantracker directory.

```
+ cd ..
+ touch .revision # creates the .revision file required for make....
```

Compile the core programs of the Dantracker

```
+ make
```

if you get the following error:

```
aprs-ax25.c:24:33: fatal error: netax25/kernel_ax25.h: No such file or directory
compilation terminated. make: *** [aprs-ax25.o] Error 1
```

then change the aprs-ax25 file as follows

```
+ -> #include <netax25/kernel_ax25.h> becomes #include <netax25/ax25.h>
```

and do again:

```
+ make
+ sudo ./install.sh
+ cp *.js /usr/share/dantracker/
```

Now you copy the jquery file as well. Basil's install script does copy the file also, but I am not sure if he renames the file during copying – *to be investigated*....

adapt both /usr/share/dantracker tracker.html as well as spy.html for the correct library

```
</div>
<script src="jquery-1.9.1.min.js"></script>
<script src="spy-frontend.js"></script>
</body>
</html>
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

*Edit config files /etc/tracker/aprs_tracker.ini & /etc/tracker/aprs_spy.ini
change [station] mycall, put your range in, adapt the tier2 server name (**euro** for Europe and
noam for North America etc...*

Install iceweasel as a web socket compatible browser (Firefox branch) for local viewing if you view it on the Raspberry screen, other while use another pc on your local network and use the ip address of your Raspberry (<http://ip-of-raspberry:8080/tracker.html>)

Pre testing:

Reboot the Raspberry, login as root and change into the dantracker directory
Then execute aprs

If you see something like the following you're ok on step one.

```
root@aprs:/home/pi/dantracker# aprs
APRS v0.02.0429 ()
DEBUG: Calling iniparser for station call PA0ESH, ssid:0
Debug: aprsis filter string: r/52.259/6.755/100.0
Processed single key: Using dantracker v0.02.0429 (), key: ver, remaining: 0
Final str: Using dantracker v0.02.0429 ()
Processed single key: Software v0.02.0429 (), key: ver, remaining: 0
Final str: Software v0.02.0429 ()
Processed 2 lines, total keys: 2
```

Now we start the dantracker by executing /etc/tracker/tracker-up

You should see the following:

```
root@aprs:/home/pi/dantracker# /etc/tracker/tracker-up
Starting Tracker
Starting Spy
There are screens on:
    2279.Spy      (30/06/13 11:01:13)  (Detached)
    2275.Tracker (30/06/13 11:01:13)  (Detached)
2 Sockets in /var/run/screen/S-root.
```

finished starting tracker

Now check the status of the tracker by executing screen -r Tracker

Noted - you may have to carry out the following change to the tracker-up file...

On a raspberry it seems there is a problem with the creation of a log file.

Therefore modify the /etc/tracker/tracker-up file as follows

```
#!/bin/sh
echo "Starting Tracker"
touch /tmp/aprs_tracker.log # added in case of error
screen -dmS Tracker -c /etc/tracker/.screenrc.trk
echo "Starting Spy"
screen -dmS Spy -c /etc/tracker/.screenrc.spy
screen -ls
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

echo "finished starting tracker"

Restart the Raspberry PI again and continue...

If you see verbose messages like these, your ok.

```
DEBUG: aprsax25_connect(): Port 0 is using device ax0, callsign PA0ESH-7 PA0ESH-12
DEBUG: aprsax25_connect(): Connected to AX.25 stack on rx socket 4, tx socket 5
DEBUG: tnc type is AX25 try aprsax25_connect to APRS via WIDE1-1, socket=4
Waiting for node script to create a Unix socket
UI Socket /tmp/N7NIX_UI, connected with sock 7
.30.06.13 11:01:20: handle encap
third party parse ok: call 0x1c01270, dest (nil), dest call 0x1c01900, time (nil), type 0x1c01870, pkt
0x1c013c8
Processed thirdparty: PI1NYV-C->APJI23 @1101: type: 0 PI1NYV-C>APJI23,TCPIP,PI1TWE-
2*:I5222.69ND00628.48Ea NIJVERDAL JO32FJ
30.06.13 11:01:21: beacon reason: ATREST, req: 1200 delta: 1372582881
```

Leave this screen by Ctrl-a-d.

Now with the Acces Point software, you have given the wlan0 device an IP address of 192.168.42.1. So when in the car with no Ethernet connection, point your iPhone, iPad of Android to this address after having made a Wi-Fi connection. At home use the ip address from the eth0 interface.

Find them both by executing ifconfig.....

```
pi@aprs ~/dantracker $ ifconfig
eth0    Link encap:Ethernet HWaddr b8:27:eb:09:a3:c8
        inet addr:10.0.1.14 Bcast:10.0.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:119290 errors:0 dropped:0 overruns:0 frame:0
        TX packets:56030 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:79009703 (75.3 MiB) TX bytes:9305670 (8.8 MiB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

mon.wlan0 Link encap:UNSPEC HWaddr 00-0F-54-02-28-DA-00-00-00-00-00-00-00-00-00-00
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:156374 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:30029676 (28.6 MiB) TX bytes:0 (0.0 B)

wlan0   Link encap:Ethernet HWaddr 00:0f:54:02:28:da
        inet addr:192.168.42.1 Bcast:192.168.42.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:10613 errors:0 dropped:44 overruns:0 frame:0
        TX packets:9171 errors:0 dropped:0 overruns:0 carrier:0
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
collisions:0 txqueuelen:1000
RX bytes:1483359 (1.4 MiB) TX bytes:2680349 (2.5 MiB)
```

```
pi@aprs ~/dantracker $
```

Note:

The install script does not setup apps to start on a power up or reboot. To have the spy & tracker apps startup at boot time do the following.

As root copy tracker startup script to the init.d dir then use update-rc.d to add the daemon.

```
+ su
+ cd /etc/tracker
+ cp tracker /etc/init.d/
```

On a Raspberry the script header had to be adapted because the requirement for ax25 to be started did not fit ell. Quick fix:

Edit the file as follows

```
# Required-Start:  $remote_fs $syslog
# Required-Stop:   $remote_fs $syslog
```

Save and exit

Also the heartbeat is not available so comment it out

```
# echo "heartbeat" > /sys/class/leds/tiger\:green\:health/trigger
```

Add a daemon with sequence of 95

```
+ update-rc.d tracker defaults 95
```

The daemon script supports start, stop, restart & status

For example as root:

```
+ /etc/init.d/tracker status
```

To start & stop de Dantracker – N7NIX branch, use /etc/tracker/tracker-up (and down for that matter)

Make sure you have adapted the aprs_tracker.ini file (and the aprs_spy.ini file as well) and use ax25 to have all functions working.

For testing you can use NET, but the message fuctions will not work.

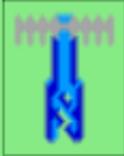
One handy feature I have done is to have the GPS connected, but use NET.

When I am driving in the car, the iPhone is setup as an Wi-Fi hotspot and connects to the Raspberry.

The range is set for the range of my VHF rigs with repeaters, approx. 60 km

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

Here is a screen picture of Sunday 26th of may 2013, after about 20 minutes of operation. In that mode.

PA0ESH 0.0km SW via TCPIP 

Xastir on iMac / when PC is on-line

1:PA3AZN-2 43 sec	5:DB0EEO 57km SW
2:PD4U-3 56km W	6:145.662- 59km SW
3:PI1DEV 41km W	7:PI1SGM-2 2m57s
4:DB0TVA 58km SW	8:PI1ZWL-11 53km N

Stationary, Alt 5 m
52.25860N 6.75795E 06:22:29 Locked: 8 sats
PA0ESH-7 : Every 20 min 

PA3DZX-13 9.7km E (12m18s ago)
Wind N 5/9m/s 10C 90%
Twente (bron: BuienRader.NL <http://www.buienradar.nl>) 

Messages
EL-DB0BOH->EL-DB0BOH@1248:UNIT.RX Erlang,TX Erlang,count/10m,count
/10m,none1,none2,logic
EL-DM0ZPK->EL-DM0ZPK@1250:UNIT.RX Erlang,TX Erlang,RXcount/10m,TXcount
/10m,none1,STxxxxxx,logic

ACK message From callsign: PA0ESH-7 To callsign:

Enter message:

Total	Messages	Weather	Encap Pkts	Error
105	2	1	0	48

Shutdown Confirm:

Debug window
12:43: URL: 10.0.1.5 Port num: 8080
12:43: Callsign: CF_CALL value: PA0ESH-7 myName: PA0ESH-7

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

===== **EOF N7NIX installation** =====

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

This is the part for the original dantracker again...

Change back into the dantracker directory

```
➤ cd ..
```

```
➤ $ touch .revision
```

Needed . You may have to add it later to the start script as well, as it did not stay on my Raspberry after a reboot, when I tried to execute make again
Note – this may be necessary for the N7NIX branch as well.

Now execute the compilation

```
➤ make
```

Do the same for the images (not required for N7NIX branch)

```
➤ cd images
```

```
➤ make
```

```
cd ..
```

This is all, and if you had no errors, then the test will show that it works.

Now the test

First check for the gps

```
➤ sudo ./detect_gps.py
```

In my case, I have a unknown GPS mouse and a TNC-X with FTDI USB to serial port
The output shows up as:

```
pi@raspberrypi ~/dantracker-ori $ sudo ./detect_gps.py  
lrwxrwxrwx 1 root root 12 Apr 23 19:40 /dev/gps -> /dev/ttyUSB1  
lrwxrwxrwx 1 root root 12 Apr 23 19:40 /dev/radio -> /dev/ttyUSB0
```

Copy the aprstest.ini file from the samples directory into your aprs directory and rename the call to your own call and change the position to some where near your home.

This will deliver a lot of data for the stress test

Now, run a screen session for the aprs function as follows

```
➤ screen -dmS aprs ./aprs -m /home/pi/dantracker/aprs.ini
```

The aprs job is now up and running on your raspberry in a (hidden) screen.

next start ui with the -i option

```
➤ xinit ./ui
```

That's it. On my screen is looks as in the youtube movie.

Now adapt the aprs.ini file from the samples directory to your real situation. The have fun with the dantracker, like I had.

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

In case you wish to autostart the whole thing do as follows:

Create a startup file in the directory /etc/init.d, called dantracker.

```
nano /etc/init.d/dantracker
```

Write the following into the file:

```
#!/bin/sh
# /etc/init.d/noip

### BEGIN INIT INFO
# Provides:      dantracker
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop:  0 1 6
# Short-Description: Simple script to start the original dantracker at boot
# Description:    A simple script by pa0esh to start the original dantracker programs at boot in
two screens. PA0ESH @21 April 2013
### END INIT INFO

# If you want a command to always run, put it here

# Carry out specific functions when asked to by the system
case "$1" in
  start)
    echo "Starting the Dantracker"
    # run application you want to start and change the working dir according to your system
    cd /home/pi/dantracker-ori
    screen -dmS aprs ./aprs -m
    screen -dmS ui xinit /home/pi/dantracker-ori/ui
    ;;
  stop)
    echo "Stopping the Dantracker"
    # kill application you want to stop
    killall screen
    killall ui
    ;;
  *)
    echo "Usage: /etc/init.d/dantracker {start|stop}"
    exit 1
    ;;
esac

exit 0

sudo chmod 755 /etc/init.d/dantracker
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

To register your script to be run at start-up and shutdown, run the following command:

```
+ sudo update-rc.d dantracker defaults
```

If you ever want to remove the script from start-up, run the following command:

```
+ sudo update-rc.d -f dantracker remove
```

73's

Erik, PA0ESH

Notes....

Somewhere in the file aprs-is.c I found data pertaining to the login for Dan I have changed that to my own data, and you should do so as well.

This is the part you are looking for.....

```
int sock;
double lat = 52.259;
double lon = 06.755;
double range = 1500;
int ret;
char buf[256];

sock = aprsis_connect("europe.aprs2.net", 14580, "PA0ESH-5",
                    lat, lon, range);
if (sock < 0) {
    printf("Sock %i: %m\n", sock);
    return 1;
}
```

After modifying execute make command...

If you use an additional Wi-Fi adapter, like I did, you may have to change your ip tables and flush them..

How to: Linux flush or remove all iptables rules

Here is small script that does this. Debian or Ubuntu GNU/Linux does not comes with any SYS V init script (located in /etc/init.d directory) .

You create a script as follows and use it to stop or flush the iptables rules.

Please don't type rules at command prompt. Use the script to speed up work.

Procedure for Debian / Ubuntu Linux

A) Create /root/fw.stop /etc/init.d/fw.stop script using a texteditor (nano)

```
#!/bin/sh
echo "Stopping firewall and allowing everyone..."
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
iptables -P FORWARD ACCEPT  
iptables -P OUTPUT ACCEPT
```

Make sure you can execute the script:
chmod +x /root/fw.stop

You can run the script:
/root/fw.stop

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

ORIGINAL README FILE FOR DANTRACKER –N7NIX

Pre-installation requirements

=====

Dependencies:

libfap:

```
http://pakkettiradio.net/libfap/  
wget http://pakkettiradio.net/downloads/libfap/1.3/libfap-1.3.tar.gz  
tar -zxvf libfap-1.3.tar.gz
```

```
cd libfap-1.3
```

apply fap patch

```
patch -p2 < ./<path_to_tracker_src>/fap_patch.n7nix  
sudo cp src/fap.h /usr/local/include/  
./configure  
make  
sudo make install
```

NOTE: libraries install to /usr/local/lib

libiniparser:

```
http://ndevilla.free.fr/iniparser/  
wget http://ndevilla.free.fr/iniparser/iniparser-3.1.tar.gz  
tar -zxvf iniparser-3.1.tar.gz
```

```
cd iniparser
```

```
sudo cp src/iniparser.h /usr/local/include  
sudo cp src/dictionary.h /usr/local/include  
make  
sudo cp libiniparser.* /usr/local/lib
```

GTKAPP: Only used by gtk app, see Makefile

```
gtk+-2.0  
libgtk2.0-dev
```

AX25: Only used with AX25 stack

```
Build from source http://www.linux-ax25.org/wiki/CVS  
libax25-dev
```

WEBAPP: Only used by web app, see Makefile

```
node.js  
requires openssl libssl-dev  
https://github.com/joyent/node  
git clone git://github.com/ry/node.git
```

```
cd node
```

```
./configure
```

```
make
```

```
sudo make install
```

```
verify by running: $ node -v && npm -v
```

json-c

```
https://github.com/json-c/json-c
```

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
git clone git://github.com/json-c/json-c.git
```

```
cd json-c
sh autogen.sh
./configure
make
sudo make install
ldconfig
```

javascript libraries

```
Needs version 1.8.3 or greater
wget http://bit.ly/jqsource -O jquery.js
sudo cp jquery.js /usr/share/dantracker
Above taken care of by install script.
```

node modules

```
ctype
iniparser
websocket
connect
```

Tools required:

```
build-essential (includes libc-dev, make, gcc, ... etc)
pkg-config
imagemagick
git
automake
autoconf
libtool
```

Basic Installation

```
`cd' to the directory containing the source code and type
`make' to compile, link & create png image files.
```

fap Installation

Before building the fap library run the fap patch in the tracker repository against its source tree. You need to supply the directory path to dantracker source.

```
cd libfap-1.3
patch -p2 < ./<path_to_tracker_src>/fap_patch.n7nix
```

For dantracker n7nix ONLY

=====

Web app install notes

=====

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

Once core apps, aprs for tracker & aprs-ax25 for spy are built need to do the following:

check that this package is installed:
screen

Install node.js & npm (package manager for node)
Install node.js modules globally:
npm -g install <module name>
npm list -g

```
ctype
iniparser
websocket
connect
```

As root run install.sh found in tracker dir. This will copy binary, javascript & config files to appropriate directories.

Confirm javascript library jquery-1.8.3.min.js exists here:
/usr/share/dantracker/

Edit config files /etc/tracker/aprs_tracker.ini & /etc/tracker/aprs_spy.ini

change [station] mycall

If you don't have a gps connected change [gps] type from nmea to static
If gps is connected set serial port and set type to nmea

For AX.25 operation change config for [ax25] port = xxx
This should be the same port name used in /etc/ax25/axports

As root run /etc/tracker/tracker-up

As root run screen -ls

should see something similar to:

```
There are screens on:
  30680.Tracker      (04/15/2013 06:10:32 PM) (Detached)
  21348.Spy         (04/15/2013 01:00:42 PM) (Detached)
2 Sockets in /var/run/screen/S-root.
```

Attach to a screen terminal

```
screen -r Spy
```

or

```
screen -r Tracker
```

If you are successful in getting these screen sessions running & you are attached use these 'screen' commands:

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

list windows:

ctrl a w

change windows

ctrl a 0

ctrl a 1

ctrl a 2

leave session by detaching:

ctrl a d

kill them:

ctrl a \

Open up your browser

Is your browser web socket enabled?

<http://caniuse.com/websockets>

web url for tracker

<your_machine_name>:8080/tracker.html

or

localhost:8080/tracker.html

web url for spy

<your_machine_name>:8081/spy.html

or

localhost:8081/spy.html

The install script does not setup apps to start on a power up or reboot. To have the spy & tracker apps startup at boot time do the following. As root copy tracker startup script to the init.d dir then use update-rc.d to add the daemon.

su

cd /etc/tracker

cp tracker /etc/init.d/

Add a daemon with sequence of 95

update-rc.d tracker defaults 95

The daemon script supports start, stop, restart & status

For example as root:

/etc/init.d/tracker status

Starting tracker or spy from command line without using script

Make sure tracker or spy is not already running

screen -ls

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

```
screen -S Tracker -c /etc/tracker/.screenrc.trk
```

or

```
screen -S Spy -c /etc/tracker/.screenrc.spy
```

Console Programs

=====

Stand alone programs displaying to console only.

```
./aprs-is
```

```
./aprs-is -c <path_to_ini_config_file>/aprs.ini
```

Display packets from an APRS server.

Need to specify in ini file:

[net] server_host_address, server_port and range

[static] lat and long

[gps] type=static

```
./aprs-ax25
```

```
./aprs-ax25 -c <path_to_ini_config_file>/aprs.ini
```

Display packets from AX.25 stack

Need to edit compile time define aprs-ax25.c and remove #define

PKT_SOCKET_ENABLE.

```
./fakegps
```

Continuously display gps GGA & RMC sentences that will be used if

[gps]:type=static is configured.

```
./faptest < file_containing_aprs_packet
```

Reads a packet from stdin and passes to fap library.

For dantracker kk7ds ONLY

=====

Configuration

=====

Copy one of the ini files from examples directory to same directory

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

as aprs executable and rename to aprs.ini

Running dantracker kk7dds edition

=====

Stress-test by using examples/aprstest.ini (modified as appropriate), which will attempt to eat the entire APRS-IS world feed without crashing.

Test Programs

Test stand alone programs displaying to console.

./aprs-is

./aprs-ax25

Test display interface.

Using socket connect with AF_INET

./ui -i with ./uiclient -i [NAME] [VALUE]

Using socket connect with AF_UNIX

./ui -w with ./uiclient -w [NAME] [VALUE]

Normal operation

UI socket connect via AF_INET

./ui -i with ./aprs -d 127.0.0.1

UI socket connect via AF_UNIX

./ui -w with ./aprs

DANTRACKER APRS on a RASPBERRY PI – VERSION 2017

TNC-X Raspberry & FTDI issues

TNC-X

I like the TNC-X with the USB option, but the FTDI driver (ftdi_sio) does not seem to work correctly on the ARMHF OS. Download the [AN 220 FTDI Drivers Installation Guide for Linux](#) from the FTDI website and follow the instructions in the PDF.

Simplified instructions

Change directories to /usr/local/src/

```
cd /usr/local/src/
```

Download the driver tarball

```
wget http://www.ftdichip.com/Drivers/D2XX/Linux/libftd2xx1.1.12.tar.gz
```

Unpack the archive,

creating the following directory structure:

```
build
```

```
arm926
```

```
i386
```

```
x86_64
```

```
examples
```

```
libusb
```

```
ftd2xx.h
```

```
WinTypes.h
```

```
tar xfvz libftd2xx1.1.12.tar.gz
```

Change to the arm926 directory

```
cd build/arm926
```

If you're already root, then not necessary.

```
sudo su
```

Copy the libraries to /usr/local/lib.

```
cp lib* /usr/local/lib
```

Change permissions.

```
chmod 0755 /usr/local/lib/libftd2xx.so.1.1.12
```

Create a symbolic link to the 1.1.12 version of the shared object.

```
ln -sf /usr/local/lib/libftd2xx.so.1.1.12 /usr/local/lib/libftd2xx.so
```