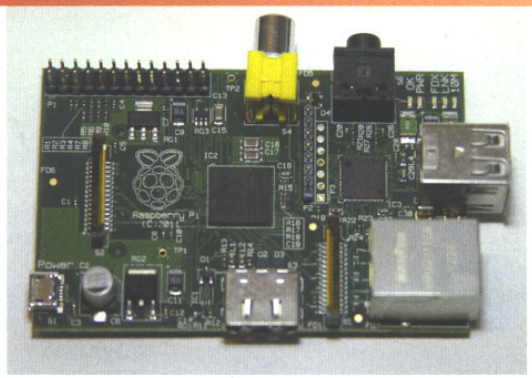
Lücken sparsam schließen

APRS-iGate mit dem Raspberry Pi



Stefan Hüpper, DH5FFL

Ganze 10 W aus dem Stromnetz gönnt sich dieser Anwendungsvorschlag auf Basis des Raspberry Pi's, bei dem im Endeffekt ein Empfangs-iGate für APRS entsteht. Der Konfigurationsaufwand hält sich ähnlich der Hardware in Grenzen.

Für einen Funkamateur, der seine Standortinformationen mobil aussendet, macht die Betriebsart APRS meist nur dann Spaß, sobald seine Datenpakete auch von einer Gegenstelle empfangen und ins Netzwerk weitergeleitet werden, beispielsweise zur Anzeige auf aprs.fi [1]. Deutschlandweit arbeiten bereits viele solcher Digis, aber es gibt eben noch genug weiße Flecken auf der Landkarte. Markus Heller, DL8RDS, beschrieb jüngst eine solche Empfangsstelle auf Basis eines Arduino-Mikrocontrollers [2]. Sein Konzept dürfte in Sachen Stromsparsamkeit auf Platz 1 liegen – wer will heute schon Desktop-PCs mit solchen Aufgaben betrauen, die sich meist über 50 W Leistungsaufnahme gönnen? Das Konzept in diesem Beitrag setzt auf die Verwendung eines Raspberry Pi's und ist somit ähnlich energiesparsam. Um diesen Beitrag zu verstehen, sollten Sie bereits den Anwendungsvorschlag [3] gelesen haben, bei dem dieser Linux-Kleinstrechner zu einem Echolink-Gateway konfiguriert wurde. Dort ist beschrieben, um was es sich bei dem Rechner handelt und wie man ihn grundsätzlich in Betrieb nimmt. Daher nehme ich hier nur Bezug auf die Schritte, die nach der grundlegenden Installation ansetzen.

Geringer Hardwareaufwand

Gleich vorweg: Auch dieser Systemvorschlag begnügt sich mit nur 10 W Leistungsaufnahme aus dem Stromnetz – und das trotz (im Laboraufbau) zweier Schaltnetzteile für 5 V/12 V und einem betagten IC-240 als Empfangstransceiver. Letzterer ist auf Flohmärkten oder bei Internet-Auktionshäusern zu finden, macht als 2-m-Monobander je nach Abgleich 10 W HF und ist aufgrund diskreter Bauteile sehr servicefreundlich. Allerdings muss man die APRS-Frequenz 144,800 MHz in Form einer Diodenmatrix fest „einlöten".

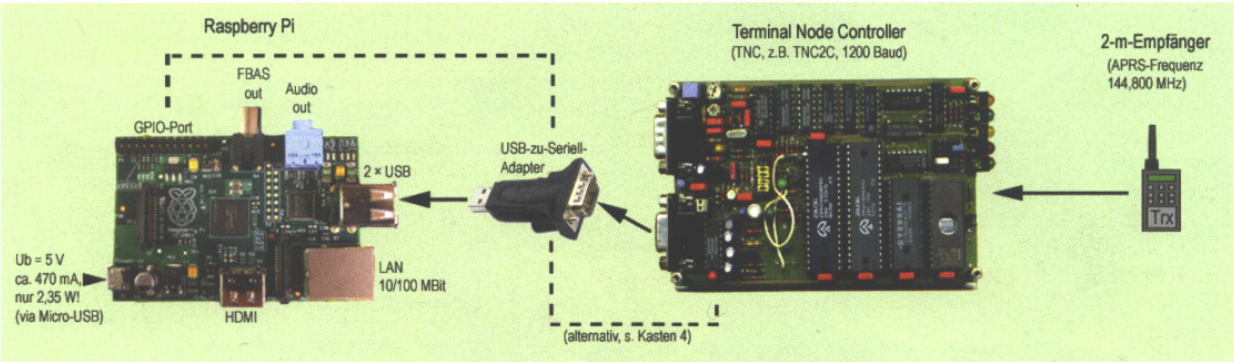
Zum Hardware-Konzept hinzu kommt ein TNC2C als AX.25-Modem sowie entweder ein USB-zu-Seriell-Wandler oder ein Pegelwandler **(s. Kasten 4)** zur Anbindung des Raspberry Pi's. Einen TNC zu nutzen ist wohl das einfachste. Zwar gibt es im Internet Anleitungen, bei denen eine USB-Soundkarte nebst Software („Soundmodem") die Decodierung der 1200 Baud AX.25-Signale für APRS übernimmt. Der Konfigurationsaufwand ist höher, und man muss eine funktionierende USB-Soundkarte finden. 1200-Baud-TNCs dagegen haben viele Funkamateure noch zu Hause aus Packet-Radio-Zeiten herumstehen, sie verbrauchen ebenfalls wenig Strom und sind auf dem Gebrauchtmarkt im Preissegment von 20...50 € erhältlich.

Software vorbereiten

Um den Raspberry Pi u.a. für AX.25 und weitere Schritte fit zu machen, laden wir erst mal ein paar Softwarepakete:

```
sudo apt-get install libax25 libax25-dev
ax25-apps ax25-tools minicom
```

Nun folgt das Herzstück: Matti Aarnio, OH2MQK, hat die Software APRX entwickelt. Mit ihr lässt sich leicht die hier beschriebene Anwendung – ein APRS-Empfangsgateway mit Datenweitergabe zum Internet – aufbauen. Neuere Softwareversionen erlauben auch das Digipeating, welches aber nicht Gegen-



Bestandteile eines APRS-iGates auf Basis eines USB-zu-Seriell-Adapters. Der Empfänger auf 144,800 MHz wird an den TNC angeschlossen, dieser decodiert die AX.25-Daten und gibt sie über seine serielle Schnittstelle an den Raspberry Pi weiter

stand des Beitrags sein soll. Zum Herunterladen wechseln wir z.B. ins /home-Verzeichnis und laden den Quellcode mit wget herunter:

```
cd /home
sudo wget http://ham.zmailer.org/
oh2mqk/aprx/aprx-2.05.svn485.tar.gz
```

Nun entpacken wir die tar-Datei und wechseln danach in jenes Verzeichnis mit

```
sudo tar xvf aprx-2.05.svn485.tar.gz
cd aprx-2.05.svn485
```

und können mit dem Compilieren beginnen. Dies geht im Vergleich zu der Anwendung in [3] deutlich schneller vonstatten. Source-Code-Konfiguration, Compilierung und Installation lassen sich mit folgender Zeile bewerkstelligen:

```
sudo ./configure && make && make
install
```

Mit dem TNC sprechen

Es wird Zeit, sich mit der Kommunikation zum TNC zu beschäftigen. Zunächst ein Blick auf dessen Betriebssystem. Hier sollte es sich entweder um „TheFirmware" oder um „6pack" handeln. Modernere TNCs (wie der TNC2Multi) erlauben das einfache Umschalten, bei anderen Geräten ist nur das EEPROM zu tauschen.

6pack ist insofern zu bevorzugen, als dass es von der Parametrierung seitens des Linux-Systems leichter angesprochen werden kann. Die TheFirmware startet nach dem Einschalten des TNCs im Hostmode, benötigt wird jedoch der KISS-Mode, der über einen Befehl geschaltet werden muss. Das klappt durch einen Init-String (s.u.), in Versuchen hatte sich der Aufbau TNC–Raspberry Pi aber dabei schonmal „verhaspelt": Der Raspberry hatte bereits den APRX-Dienst laufen, der TNC verweilte jedoch noch im Hostmode. Im Folgenden wird daher vorwiegend die Vorgehensweise bei TheFirmware beschrieben.

Hallo TNC!

Auf der Grundlage, dass der eingangs erwähnte USB-Seriell-Wandler vom System unterstützt wird, sollte dieser als /dev/ttyUSB0 dem Linux-System bekannt sein. Über selbige „Adresse" wird nun auch der TNC angesprochen. Tipps für einen Wandlertyp kann ich nicht geben, meine Versuche beruhten auf einem Typ aus einer GPS-Maus.

Damit man nicht blind konfiguriert und sich wundert, ob der TNC im Hostmode überhaupt antwortet, sollte die serielle Verbindung zunächst mit einem Terminal-Programm getestet werden. Dazu haben wir vorhin bereits Minicom installiert. Durch Aufruf von

```
sudo minicom -s
```

startet das Terminalprogramm mit dem Eingabefenster für die Schnittstelleneinstellungen. In meinem Fall: /dev/ttyUSB0, 19200 Baud, 8N1. Sofern das TNC mit TheFirmware-EEPROM ausgestattet ist, sollte man nach seinem Einschalten mit dem Versionstext auf dem Bildschirm begrüßt werden. Nun kann man ein wenig herumspielen und schauen, ob man schon etwas auf 144,800 MHz empfängt. Im Hostmode gibt es einige Steuerbefehle, die de facto ein Packet-Radio-Terminal abbilden. Drückt man die Folge „<ESC>S 0", so befindet man sich im „Monitorfenster". Es bleibt aber noch leer, da APRS-Sendungen unprotokolliert erfolgen. Zum Anzeigen selbiger gibt man „<ESC>m u" ein, wobei alle Unproto-Sendungen nun in Minicom angezeigt werden sollten.

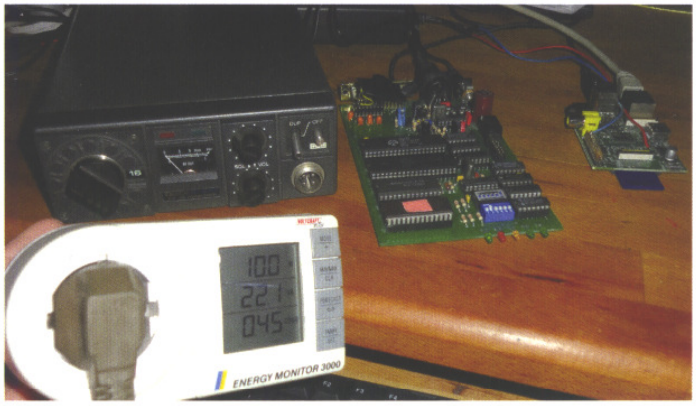
Nun müssen wir in den KISS-Mode wechseln, wobei der TNC die Daten 1:1 an die serielle Schnittstelle weitergibt. Das Umschalten können wir händisch einmal mit „<ESC>@K" ausführen. Später wird ein Skript das Schalten in den KISS-Mode – hoffentlich – automatisch übernehmen.

APRX konfigurieren

Im Gegensatz zu anderen Betriebssystemen lässt sich AX.25 in Linux tief verwurzeln, daher passen wir zunächst die Systemdatei /etc/ax25/axports an, die dem Linux-Kernel einen Packet-Port definiert **(Kasten 1)**. „Packet" ist ein willkürlich definierter Name, gefolgt vom iGate-Rufzeichen (hier: DH5FFL–10), der seriellen Schnittstellengeschwindigkeit, zweier Paketparameter und einer Beschreibung. Als nächstes muss das APRX-Paket konfiguriert werden, die Steuerdatei lautet /etc/aprx.conf. Diese ist von Hause aus ausführlich mit vielen Kommentaren versehen, sodass Sie die Chance haben, Ihr System nach eigenen Wünschen anzupassen. So kann man es z.B. auch als (sendenden) Digipeater ausbauen.

Kasten 2 listet die Minimal-Konfiguration, wie sie im Testsystem zum Einsatz kommt. An oberster Stelle wird aber-

Wird dieses iGate im lokalen Netzwerk betrieben, so sind im evtl. vorhandenen Router keine Portweiterleitungen nötig. Portsperren dürfen selbstredend nicht bestehen.



Versuchsaufbau des Empfangs-iGates: IC-240 (l.), der TNC und rechts der Raspberry Pi benötigen inklusive zweier Schaltnetzteile (5 V/12 V) 10 W Input aus dem Stromnetz

Kasten 1: axports

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
Packet DH5FFL-10 19200 128 2 144.800MHz APRS 1200bps
```

Kasten 2: aprx.conf

```
mycall  DH5FFL-10

<aprsis>
   server euro.aprs2.net 14580
</aprsis>

<logging>
   pidfile /var/run/aprx.pid
   rflog /var/log/aprx/aprx-rf.log
   aprxlog /var/log/aprx/aprx.log
   erlangfile /var/run/aprx.state
</logging>

<interface>
  ax25-device $mycall
  tx-ok false  # transmitter enable defaults to false
</interface>

<beacon>
   beaconmode aprsis
   cycle-size  5m
   beacon symbol „R&" lat „5125.50N" lon „00920.50E" comment
   „Rx-iGate auf Raspberry Pi"
</beacon>
```

mals das iGate-Call definiert. Unter <aprsis> wird der Internet-Server angegeben, an den die empfangenen Pakete versendet werden, hier: euro.aprs2.net auf Port 14580. Unter <logging> werden einige Log-Dateien definiert. Die Sektion <interface> verweist APRX auf das Linux-eigene AX.25-Interface mit dem in der ersten Zeile definierten Rufzeichen. „tx-ok false“ sagt dem System (im Gegensatz zum Wert „true“), dass es sich nur um ein Empfangs-Gateway handelt. Der <beacon>-Bereich definiert den Baken-Modus; „aprsis“ sagt dem System, dass Baken auf der Zeitbasis „cycle-size“ (= 5 Minuten) nur ins Internet gesendet werden sollen. Die letzte Zeile gibt das APRS-Symbol vor, und man trägt hier die Geo-Koordinaten der eigenen Empfangsstelle ein. Damit ist APRX funktionsfähig.

Etwas Komfort, bitte!

Damit kämen wir bereits zum Skript in **Kasten 3**. Dieses legt man optimalerweise im Suchpfad an, sodass man es von überall im Dateisystem aus starten kann, nämlich unter /usr/local/bin. Um es händisch zu erstellen, wechseln Sie z.B. in jenes Verzeichnis und legen eine leere Datei – z.B. mit dem Namen „start-aprx“ – an mit

```
cd /usr/local/bin
touch start-aprx
```

Den Rest nehme ich als alter DOS- und Windows-User gern im Midnight-Commander (Aufruf: „sudo mc“) vor. Empfehlenswert ist dazu der interne Texteditor des selbigen. Zum einmaligen Aktivieren drückt man F9, um ins Menü zu gelangen, geht auf „Options“, „Configuration“ und aktiviert „Use internal edit“. Nun wechsele man ins Vereichnis unserer leeren Start-Datei und sage hier mit F4 „Bearbeiten“. Tippen Sie ruhig die paar Zeilen in Kasten 3 ab – so bekommt man sogleich ein Gefühl für die Linux-relevanten Befehle! Damit das Skript startfähig wird, drücken wir im Midnight-Commander F9 fürs Menü, gehen auf „File“ und dann auf „Advanced chown“. Start-aprx erhält nun bei den Rechten den x-Parameter durch Drücken auf selbige Taste.

Das Skript gibt sich zunächst sehr kommunikativ und spricht zu Ihnen mit dem Echo-Befehl. Über stty wird die serielle Schnittstelle in ihrer Geschwindigkeit parametriert (19200 Baud zum TNC). Der Sleep-Befehl lässt das System kurz eine Sekunde warten, was dafür sorgen soll, dass der TNC den wichtigen nächsten Befehl mitbekommt: Die String-Sequenz \r\ 033@K\r versetzt ihn in den KISS-Mode. Mit Kissattach wird die serielle Schnittstelle dem Linux-System bekannt gemacht, hier taucht wieder der eingangs definierte Name in der /etc/ax25/axports auf, gefolgt von einer IP-Adresse. Für den Fall mit 6pack-EEPROM im TNC ist in Kasten 3 auch die alternative Zeile (auskommentiert) angegeben.

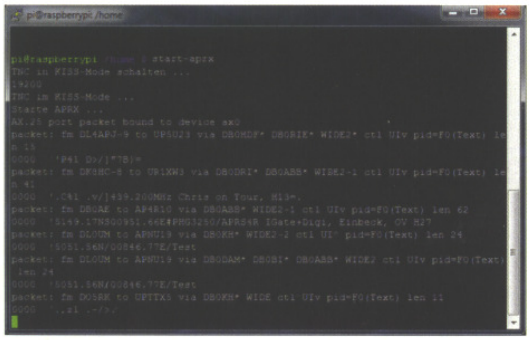
Die IP-Adresse ist in diesem Anwendungsfall nicht grundsätzlich wichtig. Sofern aber vorhanden, tragen Sie hier Ihre AX.25-IP-Adresse ein. Vor geraumer Zeit konnte man sich mal welche bei IP-koordinierenden OMs zuweisen lassen … Kissparms definiert folgende Parameter: Txdelay (–t) = 700 ms, Slottime (–s) = 200 ms, Perist (–r) = 32, Txtail (–l) = 100 ms, Simplex (–f). APRX startet dann unsere Anwendung. „axlisten -p packet“ zeigt uns schließlich alle empfangenen Pakete zur Übersicht an. Letzteres kann man durch einen Druck von <STRG>+C abbrechen, APRX läuft im Hintergrund weiter. Gibt man den axlisten-Befehl später noch einmal, so erscheinen wieder alle Empfangspakete auf dem Bildschirm.

Spätestens jetzt sollte man sich mit seinem APRS-Tracker ins Auto setzen oder aufs Fahrrad schwingen und mal eine

Kasten 3: start-aprx

```
#!/bin/bash
#
echo „TNC in KISS-Mode schalten ...“
stty -F /dev/ttyUSB0 speed 19200
sleep 1
echo -ne „\r\033@K\r“ > /dev/ttyUSB0
echo „TNC im KISS-Mode ...“
sleep 2
echo „Starte APRX ...“
kissattach /dev/ttyUSB0 packet 44.130.27.78
#für 6pack anstelle TF-Firmware lautet der Kissattach-Befehl:
#kissattach -6 /dev/ttyUSB0 packet 44.130.27.78
kissparms -p packet -t 700 -s 200 -r 32 -l 100 -f n
aprx
axlisten -p packet
```

Lohn der Mühen: Screenshot der Aprs.fi-Webseite. Hier hat das iGate eine Station direkt empfangen [Karte: (C) OpenStreetMap-Mitwirkende, Daten unter Open-Database-Lizenz verfügbar, CC BY-SA]



Bildschirmausgaben des Start-Skriptes, das nach erfolgreichem Durchlauf APRS-Pakete auf 144,800 MHz anzeigen sollte

Literatur und Bezugsquellen

- [1] Webseite zur Visualisierung des APRS-Verkehrs: www.aprs.fi
- [2] Markus Heller, DL8RDS: „AirGate – ein sparsames Rx-IGate für APRS“, CQ DL 9/12, S. 636ff.
- [3] Stefan Hüpper, DH5FFL: „Ein Echolink-Rechner mit <3 W Leistungsaufnahme“, CQ DL 10/12, S. 695ff.
- [4] Pinbelegung des GPIO-Ports: http://elinux.org/RPi_Low-level_peripherals
- [5] Gehäuse für den Raspberry Pi, z.B. bei Reichelt Elektronik, Best.-Nr. „TEK-BERRY“, www.reichelt.de
- [6] Ein weiteres Projekt, ein D-Star-DV Access Point: www.youtube.com/watch?v=_llJ_yvXy4s

Kasten 4: Die APRS-Box – alles drin!

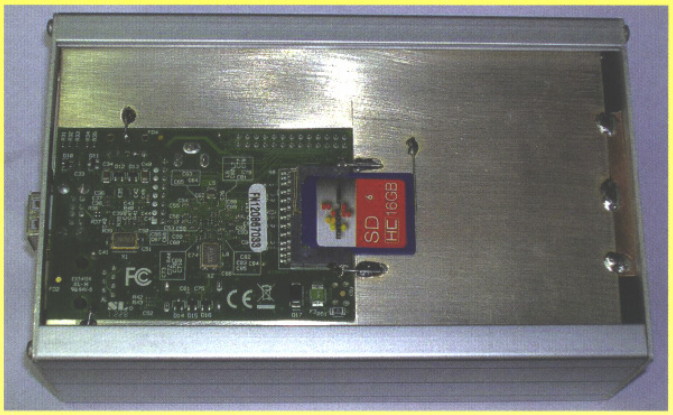
Der im Beitrag beschriebene USB-zu-Seriell-Wandler könnte gar entfallen und durch einen Pegelwandler ersetzt werden: Der Raspberry Pi bietet auf seiner GPIO-Pinleiste [4] bereits eine serielle Schnittstelle an. Problem: Sie arbeitet mit 3,3-V-TTL-Pegel und verträgt keine 5-V-TTL-Signale! Weiterhin wird auf vielen Webseiten berichtet, dass die GPIO-Pins in jedem Fall vor Kurzschlüssen und „ohmschen Schocks" zu schützen sind, da das Schaltungskonzept sehr sparsam ausgelegt ist. Der Bastler arbeite mit Vorsicht.

Zur direkten Adaption hat Dietmar Austermühl, DL1ZAX, eine SMD-Platine entworfen, die man direkt auf die GPIO-Pinleiste steckt. Sie führt über vier Leitungen (RxD, TxD, GND, +5 V) zu einer weiteren, welche anstelle des MAX-232 auf die entsprechende IC-Fassung auf der TNC-Platine gesteckt wird. Auf diese Weise erfolgt die serielle Datenverbindung sowie die Stromversorgung des Raspberry Pi's direkt aus dem TNC. Sein eingangsseitiger 7805-Spannungsregler wurde infolge der erhöhten Stromaufnahme durch einen Schaltwandler ersetzt.

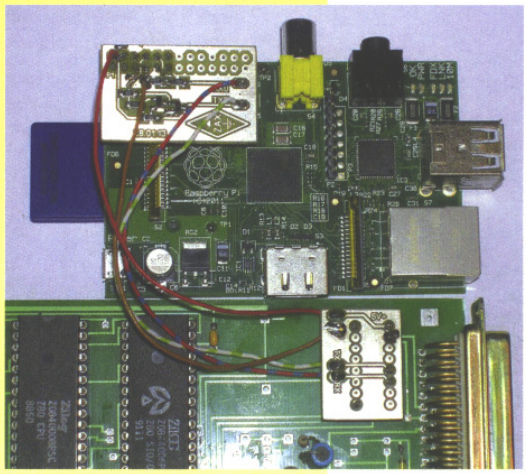
Um diese Errungenschaft zu nutzen, sind softwareseitig drei Dinge nötig:

- Anstelle /dev/ttyUSB0 muss es nun überall /dev/ttyAMA0 lauten
- In der Datei /etc/inittab muss die Zeile „T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100" durch ein „#" vorneweg auskommentiert werden. Diese Änderung macht die bordeigene UART frei, die zuvor für einen System-Login via serielle Konsole vorgesehen war.
- In der Datei /boot/cmdline.txt gibt es folgende Zeile: „dwc_otg.lpm_enable =0 *console=ttyAMA0,115200 kgdboc=ttyAMA0,115200* console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait" – entfernen Sie die hier kursiv gesetzten Verweise auf ttyAMA0.

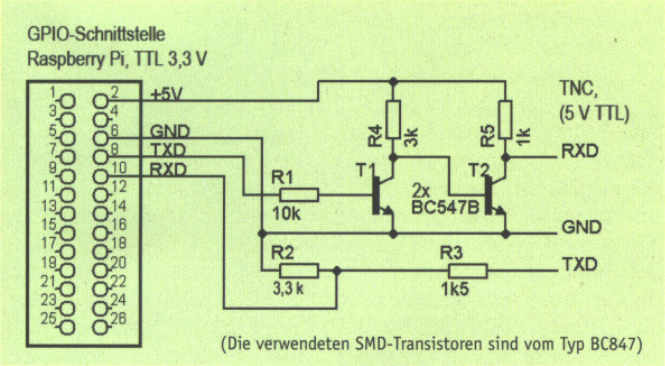
Nach einem Neustart sind die Änderungen aktiv.



Befestigungs-Vorschlag des Raspberry Pi (alte Platinenversion ohne Bohrlöcher) im TNC-Gehäuse



Verbindung Raspberry Pi zum TNC



5-V-zu-3,3-V-Pegelwandler für die GPIO-UART

In Versuchen funktionierte die Verbindung zum TNC nur mit 9600 Baud; beachten Sie dies ggf. in der /etc/ax25/axports, im Aprx-Startskript (Zeile mit stty) sowie an den TNC-Dip-Schaltern.

In der Fortentwicklung des Projekts entstand eine kompakte APRS-Box. Trx und Strom dran, Netzwerkkabel einstecken – fertig!

Interessenten für Platinen wenden sich an die Atmel User Group Kassel (http://loh-auster.de/ATMega).

Dietmar Austermühl, DL1ZAX
Stefan Hüpper, DH5FFL

kleine Runde drehen, um das iGate mit eigenen Positionsbaken zu versorgen.

Erfahrungen und Tipps

Mittlerweile wird die nächste Platinenversion des Raspberry Pi's ausgeliefert, sie verfügt im Gegensatz zur ersten – endlich – über zwei Befestigungslöcher zur Montage. Zur Unterbringung gibt es weiterhin beispielsweise bei Reichelt ein preiswertes Gehäuse [5]. Im Übrigen werden alle neuen Raspberrys (zum gleichen Preis) nun mit 512 MB ausgeliefert, da die ursprünglichen 256 MB RAM den Entwicklern offenbar noch etwas dürftig erschienen.

Infolge des ersten Beitrags [3] bleibt abermals festzustellen, dass der kleine Linuxrechner offenbar in vielen Sparten des Amateurfunks zum Einsatz kommen kann. Beispielsweise zeigt OM Roy, M1EIV, in einem Youtube-Video [6] einen portablen D-Star DV-Accesspoint. Suchen im Internet lohnen sich, um auf neue Projekte aufmerksam zu werden.

Ich stelle mir die Frage: Muss ich mir jetzt den dritten Raspberry Pi bestellen?

CQ DL