

APRS mit dem TinyTrak II

Dipl.-Ing. FRANK RUTTER – DL7UFR

Nach kochbuchartiger Vorstellung anderer Varianten ist in dieser Ausgabe ein Bausatz an der Reihe, der eine preisgünstige Alternative zu APRS-fähigen Transceivern darstellt. TinyTrak bildet ein Interface zwischen GPS-Empfänger und 2-m-Funkgerät, das zur selbsttätigen Aussendung von APRS-Paketen dient.

Mit TinyTrak hat Byon Garrabrant, N6BG, eine Schaltung entwickelt, mit der es möglich ist, Daten eines GPS-Empfängers [1] auszuwerten, in das MIC-E Format zu konvertieren und zu versenden [9]. Anders als beim TNC2 [8] kann man zusätzlich zu den GPS-Daten weitere Informationen versenden. Alle Funktionen übernimmt ein einziger PIC16F84.

Eine weiterentwickelte Version ist nun als TinyTrak II erhältlich. Während die Schaltung beibehalten wurde, erfolgte ein Austausch des PIC16F84 gegen einen PIC16F628 ausgetauscht. Dies war notwendig, da sich die neue Software auf dem PIC16F84 nicht mehr unterbringen ließ.

■ Aufbau des Gerätes

Der Bausatz, den uns dankenswerterweise WiMo zur Verfügung stellte, besteht aus einer Leiterplatte, allen benötigten Bauelementen, einer Diskette und einer englischen Kurzbeschreibung. Auf der Diskette befindet sich neben der aktuellen Bauanleitung die Software zur Programmierung der nutzerspezifischen Daten. Ein Gehäuse und Buchsen zum Anschluß der Stromversorgung und des Funkgerätes gehören nicht zum Lieferumfang, dafür ist das Ganze recht preisgünstig.

Die Leiterplatte verfügt über verzinnte Leiterbahnen sowie einen Bestückungsaufdruck. So gelingt der Aufbau auch ohne Englischkenntnisse im Handumdrehen. Ich habe die Leiterplatte in einem kleinen Weißblechgehäuse $74 \times 55 \times 30 \text{ mm}^3$ untergebracht. Um die vorhandenen Anschlußkabel, die für den Betrieb eines TNC2 an meinen Funkgeräten gefertigt wurden, weiter nutzen zu können, habe ich in das Weißblechgehäuse eine 5polige DIN-Buchse eingebaut. Die Belegung entspricht exakt der Belegung eines TNC2. Für den Anschluß der Stromversorgung eignet sich das Hohlsteckersystem.

Nach dem Zusammenbau kommen der erste Funktionstest und die Konfiguration des TinyTrak mittels der Software auf der Diskette. Nach dem Einschalten der Stromversorgung leuchtet die LED Power, die anderen blinken kurzzeitig. Der TinyTrak wird anschließend mittels eines 9poligen Sub-D-Kabels mit einem Computer verbunden.

Zum Abgleich startet man das Programm *TinyTrakIIConfig* von der Diskette. Bild 1 zeigt das Fenster mit zwei Karteikarten, die alle Einstellmöglichkeiten enthalten. Zum Test der Verbindung zum TinyTrak betätigt man am einfachsten die Schaltfläche *Read Version*. Bei ordnungsgemäßem Anschluß öffnet sich ein kleines Fenster, das die Version der TinyTrak-Firmware ausweist.

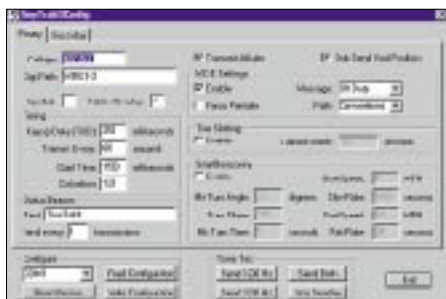


Bild 1: Karteikarte *Primary* im Konfigurationsprogramm „TinyTrakIIConfig“

Als erstes wird der NF-Pegel des Sendesignals bei über die DIN-Buchse angeschlossenen Funkgerät eingestellt. Über einen Mithörfempfänger beobachtet man die Lautstärke des Signals. Im Bereich *Tone Test* des Konfigurationsprogramms gibt es vier Schaltflächen, drei zum Ansteuern des zu sendenden Signals, eine zum Stoppen der Aussendung. Nach Betätigung einer der Sendesignalschaltflächen geht das Funkgerät auf Sendung, und die zweite rote LED „PTT“ leuchtet.

Mit R6 wird das Sendesignal so lange erhöht, bis ein leichter Rückgang der Lautstärke im Mithörfempfänger feststellbar ist. In einigen Fällen ist es notwendig, den 220-k Ω -Widerstand R5 gegen 100 k Ω , im Fall meiner Testkonfiguration mit einem VX-1R sogar auf 10 k Ω , zu verkleinern, um das oben beschriebene Verhalten beim Abstimmen beobachten zu können.

Im Anschluß erfolgt die Einstellung der Trägererkennung mit R9. TinyTrak ist nicht in der Lage, zwischen Daten, Sprachsignal oder Rauschen zu unterscheiden. Die Schaltung nutzt ausschließlich die Rauschsperrung des Funkgerätes. Deshalb beschränkt sich der Abgleich lediglich darauf, daß bei geöffneter Rauschsperrung die gelbe LED „DCD“ leuchtet, und beim Abfallen der Rauschsperrung umgehend verlischt.

Den Abschluß bildet die Prüfung der Frequenz des NF-Signals. Dazu schließt man einen Frequenzmesser an die DIN-Buchse. Mittels der Schaltflächen *Send 1200 Hz* und *Send 220 Hz* und der Veränderung des Wertes des Parameters *Calibration* erfolgt der Abgleich.

Mit dem vorgegebenen Wert in *Calibration* von 128 konnte ich bei meinem Exemplar 1198 bzw. 2197 Hz messen. Die Erhöhung bzw. Verringerung des Wertes in *Calibration* um 1 führte zu einer Absenkung bzw. Erhöhung beider Frequenzen um etwa 10 Hz. Der Vorgabewert konnte beibehalten werden. Damit ist die Schaltung vollständig abgeglichen.

■ Parametereinstellung

Im TinyTrak können zwei verschiedene Konfigurationen gespeichert werden. Diesen sind im Konfigurationsprogramm die beiden Karteikarten *Primary* und *Secondary* zugeordnet. Die Aktivierung der jeweiligen Konfiguration erfolgt über den Schalter S1. Liegt Pin 12 des PIC an 5 V, ist die *Primary*-Konfiguration aktiviert, bei 0 V *Secondary*. Die Einstellmöglichkeiten sind für beide Karteikarten gleich.

Unter *Call Sign* kann ein Amateurfunkrufzeichen eingetragen werden. Seine Länge ohne SSID ist auf 6 Zeichen begrenzt. Als Zusatz zum Rufzeichen ist ein SSID von 1 bis 15 erlaubt. Wird ein taktisches Rufzeichen wie z.B. *SHUTLE* benutzt, ist das Amateurfunkrufzeichen dann unbedingt als Stations-ID einzutragen.

Im Punkt *Digi Path* wird eine Festlegung darüber getroffen, wie APRS-Pakete über Digipeater weiter übertragen werden sollen. Als Standard kann hier *RELAY*, *WIDE* eingetragen werden [3], [5].

Mit den Parametern *Symbol* und *Table/Overlay* wird das Symbol festgelegt, das als Stationssymbol an Position der Station in Programmen wie WinAPRS dargestellt werden soll. Für meine Tests habe ich als Symbol ein kleines Flugzeug ausgewählt. Dazu ist unter *Symbol* ein ' ' und unter *Table/Overlay* ein / einzutragen. Eine Liste der darstellbaren Symbole findet man in [3]. Das *Keyup Delay (TXD)* ist Packet-Radio-Nutzern als *TX Delay* bekannt. Hier sollte man 200 ms eintragen.

Unter *Transmit Every* wird festgelegt, aller wieviel Sekunden das Bakensignal ausgesendet wird. Werte von 1 bis 5535 sind zulässig. Nach [3] wird eine Aussendung der Positionsdaten in Abhängigkeit von der Anzahl der benutzten Digipeater in folgenden Intervallen empfohlen:

- direkte Aussendung (ohne Nutzung von Digipeatern): 10 Minuten,
- über einen Digipeater: 10 Minuten,
- über zwei Digipeater: 20 Minuten,
- über drei und mehr Digipeater: 30 min.

Für Feststationen sollten 30 min, also 1800 s eingestellt werden. Alternativ zu dem festen Sendezeitintervall ist in TinyTrak II eine intelligente Steuerung des Sendeintervalls in Abhängigkeit von der Bewegungsgeschwindigkeit implementiert – mehr dazu weiter unten.

Quiet Time ist die Zeit, die vergehen muß, bis der Sender auf Sendung geht, nachdem die Rauschsperrung geschlossen hat.

Der Parameter *Calibration* wurde bereits beim Abgleich des Gerätes eingestellt.

Unter *Status Beacon Text* kann eine Information zur Station eingetragen werden. Dafür stehen maximal 89 Zeichen zur Verfügung, wenn der Status-Text der *Primary*- und *Secondary*-Konfiguration gleich sind. *Send Every* legt die Anzahl von Aussendungen fest, nach denen zusätzlich der Statustext ausgesandt wird.

Legt man Wert auf die Mitteilung der Höhendaten, gehört unter *Transmit Altitude* ein Häkchen hin.

TinyTrak basiert auf der Auswertung von Daten eines GPS-Empfängers. Die manuelle Eingabe von Positionsdaten ist nicht vorgesehen. Deshalb kann man unter *Only Send Valid Position* wählen, ob nur gültige Daten ausgesandt werden sollen.

MIC-E steht für ein Protokoll der Komprimierung von APRS-Daten. Detaillierte Informationen findet man in [3].

Mit einem Häkchen in *MIC-E Settings Enable* wird die MIC-E-Kodierung eingeschaltet. Wenn *MIC-E Force Printable* aktiviert ist, wird das Datenformat leicht modifiziert, damit alle druckbaren Zeichen darstellbar sind. Allerdings werden dadurch die zu übermittelnden Positionsdaten ungenauer.

Unter *MIC-E Messages* kann man zwischen acht verschiedenen vordefinierten Anmerkungen zur aktuellen Position wählen. Dies sind *Off Duty*, *En Route*, *In Service*, *Returning*, *Committed*, *Special **, *Priority ** und *Emergency ***. Während das

Aussenden einer mit * gekennzeichneten Anmerkung bei allen Empfangsstationen zu einer hervorgehobenen Darstellung führt, erfolgt bei Aussendung einer mit ** gekennzeichneten Anmerkung eine akustische Alarmierung – hier ist also Vorsicht geboten.

In *MIC-E Path* findet man 16 vordefinierte Einstellungen. Hat man eine Festlegung unter *Digi Path* getroffen, sollte hier *Conventional* gewählt werden.

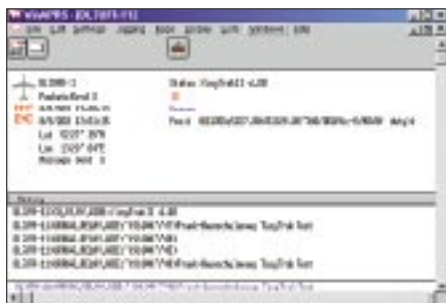


Bild 2: Das Anzeigefenster von „WinAPRS“ zeigt APRS-Pakete, die mit dem Testgerät ausgesandt wurden.

Time Slotting und *Smartbeaconing* beschreiben Verfahren zur Steuerung des Sendezeitintervalls. Beim *Time Slotting* wird der Sendezeitpunkt mit der GPS-Zeit synchronisiert. Wurde unter *Transmit Every* ein Wert von 600 bzw. 60 gewählt, erfolgt die Synchronisation mit der GPS-Zeit aller 10 min bzw. jede Minute. Unter *Transmit Offset* wird der Sendezeitpunkt nach einer Synchronisation festgelegt.

SmartBeaconing beschreibt eine intelligente Steuerung des Sendezeitintervalls in Abhängigkeit von der Bewegungsgeschwindigkeit und der Bewegungsrichtung. *SmartBeaconing* und *Time Slotting* schließen sich gegeneinander aus. Der Algorithmus sieht vereinfacht so wie im untenstehenden Kasten aus. Eine genaue Beschreibung von *SmartBeaconing* findet sich unter [10].

Sind alle Parameter eingestellt, kann man mittels der Schaltfläche *Write Configura-*

tion die Daten in den EEPROM des PIC schreiben. Das erfolgreiche Schreiben wird mit einem Fenster *Write Successful* quittiert.

Nun steht der Aussendung von APRS-Daten mittels TinyTrak II nichts mehr im Weg. Zum Anschluß des GPS-Empfängers mittels eines Original-Garmin-Datenkabels an den TinyTrak benötigt man einen Nullmodem-adapter mit 9poligen SUB-D-Steckern. Ein solcher Adapter ist schnell selbst zusammengelötet. Die beiden Pin 5 sind miteinander zu verbinden (Ground) sowie die Pin 2 und Pin 3 über Kreuz (TXD und RXD). Hat man alles zusammengesteckt und schaltet den GPS-Empfänger nach dem TinyTrak ein, blinkt die grüne LED. Dies ist ein Zeichen dafür, daß der TinyTrak eine Verbindung zum GPS-Empfänger hat, die Daten aber noch nicht gültig sind.

Erst wenn der GPS nach einer Zeit mehrere Satelliten empfangen hat, ist er in der Lage, eine gültige Standortposition zu ermitteln und auszugeben. Dann geht die blinkende grüne GPS-Status-LED in Standlicht über. Hat man unter „Only Send Valid Position“ ein Häkchen gemacht, beginnt nun der TinyTrak mit der Aussendung von APRS-Daten.

■ Fazit

TinyTrak II ist eine preiswerte Alternative zum Kauf eines APRS-fähigen Funkgerätes. TinyTrak II zeichnet sich durch einen geringen Stromverbrauch aus (3 mA plus 3 mA je leuchtende LED). Ein Betreiben der Schaltung aus einer Batterie ist damit möglich. Der Bausatz läßt sich trotz Fehlens einer deutschen Beschreibung ohne Probleme aufbauen. Das Sendezeitintervall paßt sich automatisch der Bewegungsgeschwindigkeit und -richtung an. Damit hinterläßt man sehr genau seine Spuren, ohne unnötig oft Daten auszusenden.

Literatur

- [1] National Marine Electronics Ass.: Homepage. <http://www4.coastalnet.com/nmea/default.html>
- [2] Bennett, P.: NMEA-0183 and GPS Information. <http://vancouver-webpages.com/peter>
- [3] The APRS Working Group: Automatic Position Reporting System – APRS Protocol Reference – Protocol Version 1.0. www.tapr.org
- [4] Garmin Corporation: GPS 48 Handbuch. Firmenschrift, deutsche Übersetzung, 1998
- [5] Rutter, F., DL7UFR: APRS aus der Hand – die Starthilfe für den Einsatz des TH-D7E, FUNKAMATEUR 50 (2001) H. 7, S. 737–739
- [6] Hirschelmann, K., DJ7OO: Die GPS-Homepage. www.kh-gps.de
- [7] Horzepa, S., WA1LOU: APRS: Tracks, Maps and Mobiles. 2. Auflage, ARRL, Newington 2000
- [8] Rutter, F., DL7UFR: APRS mit TNC2 – geht das überhaupt? FUNKAMATEUR 50 (2001) H. 8, S. 917–918
- [9] Garrabrant, B., N6BG: TinyTrak-Homepage. www.byonics.com/tinytrak
- [10] Bragg, Steve, KA9MVA: HamHUD-Homepage. www.hamhud.net

Vereinfachter Programmablauf des SmartBeaconing

```

IF (speed < low_speed)
{
    beacon_rate = slow_rate;
}
ELSE
{
    // Adjust beacon rate according to speed
    IF (speed > high_speed)
        beacon_rate = fast_beacon_rate;
    ELSE
        beacon_rate = fast_beacon_rate * high_speed / speed;
    // Corner pegging
    turn_threshold = turn_min + turn_slope / mph;
    IF (heading_change_since_beacon > turn_threshold) AND (secs_since_beacon > turn_time)
        secs_since_beacon = beacon_rate;
}
if (secs_since_beacon > beacon_rate)
    // ... send beacon

```