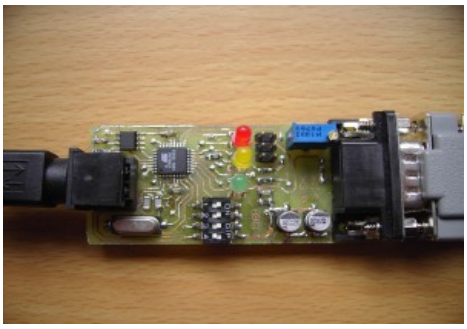
APRS-Tracker

mit AVR Mega8



Eine Weiterentwicklung von

WhereAVR von N4TXI

http://www.knology.net/~gdion/whereavr.html

Ralf Wilke DH3WR

www.dh3wr.de

Inhaltsverzeichnis

Beschreibung des Projekts

Der APRS-Tracker ist im Wesentlichen ein 1200 Baud Packet Radio-Modem, welches durch Software in einem Mikrocontroller realisiert wird. Die Positions-Information wird von einem GPS-Empfänger eingelesen und das hier verwendete NMEA-Protokoll decodiert. Nun werden die Roh-Daten in das APRS-Format codiert und dieses in Frames gemäß dem AX.25-Protokoll eingebunden. Aus den so erzeugten Frames werden die Modulations-Töne 1200 Hz und 2200 Hz, wie sie bei 1200 Baud-Packet Radio verwendet werden, per 4-Bit D/A-Wandlung erzeugt und auf den Moduations-Eingang des Funkgeräts gegeben. Zur zeitlichen Steuerung der Aussendungen wird die von GPS übertragene Uhrzeit benutzt. Ferner erfolgt eine Auswertung, ob sich das Objekt gerade bewegt oder stillsteht. Im letzten Fall wird die Rate der Aussendungen drastisch reduziert, um unnötige Frequenz-Belastung zu vermeiden.

Als weitere Funktion ist eine Empfangsmöglichkeit von APRS-Signalen in der Software implementiert. Der Tracker ist dadurch in der Lage, auf Nachrichten, die zum APRS-Protokoll konform sind, mit einer Empfangsbestätigung zu anworten und den Empfang selbst zu Steuerungszwecken zu nutzen. Die gesamte Decodierung des AX.25-Signals findet auch hier in Software statt.

Zusätzlich ist die Möglichkeit vorgesehen, die Betriebs-Spannung und die Umgebungs-Temperatur der Schaltung zu messen und diese mit den APRS-Positions-Meldungen auszusenden.

Erklärung der Schaltung

Hauptbestandteil der Schaltung ist der Mikrocontroller AT Mega8-16 von ATMEL. Er übernimmt alle Funktionen, die in der Schaltung verwendet werden. Der GPS-Empfänger wird an der 6poligen Mini-Din-Buchse angeschlossen und von dort aus mit Strom versorgt. Beim Autor ist ein GPS-Empfänger vorhanden, der eine Datenleitung mit Invertierung besitzt, daher ist ein Inverter vorgesehen, der aber nur in diesem Fall bestückt werden muss. Sonst ist der 0-Ohm-Widerstand R12 zu schließen und so die Datenleitung auf die serielle Schnittstelle des Mega 8 zu legen. Die gelbe LED D1 wird zur Anzeige verwendet, ob gültige GPS-Daten empfangen wurden.

Das Widerstand-Netzwerk R1 bis R4 erzeugt aus einer digitalen Ansteuerung in Verbindung mit R5 ein analoges Signal, welches mit R5/C4 tiefpassgefiltert wird. Durch diese Schaltung wird das 1200 Baud Packet-Radio-Signal aus Sinus-Schwingungen mit 1,2 kHz und 2,2 kHz erzeugt. C5 sperrt nun den Gleichstrom-Anteil. Das PTT-Signal zum Einschalten des Sender wird über R17 / Q1 erzeugt. Hat das Funkgerät sepearate Leitungen für das Modulations-Signal und PTT, so entfällt R7. Ansonsten sorgt er für einen Gleichstrom-Anteil auf der Leitung, die das Funkgerät zur Umschaltung auswertet. D4 leuchtet synchron mit aktivierter PTT.

Der anfängliche Reset des Mikrocontrollers wird durch R10 / C3 realisiert.

Um die Betriebsspannung zu messen, wird diese mit R19 / R20 um den Faktor ca. 0,25 geteilt, damit sie mit dem max. möglichen Eingangsspannungs-Bereich des A/D-Wandlers im Mikrocontroller übereinstimmt. C13 dient zur Glättung.

Der Temperatur-Sensor ist über das Two-Wire-Interface (auch I²C genannt) an die entsprechenden Pins am Mikrocontroller angeschlossen. Die Kommunikation mit dem Sensor und damit auch die Datenerfassung erfolgt rein digital.

Um Packet-Radio zu empfangen, wird das Lautsprecher-Signal des Funkgeräts über C10

gleichstrom-entkoppelt und mittels R11 / R13 auf die Schaltschwelle des im Mikrocontroller vorhandenen Komparators gelegt. Eine empfangen Modulation lässt die Spannung am Eingang des Komparators um die Schaltschwelle schwingen und die Frequenz der empfangenen Signale kann durch Messung der Zeit von Nulldurchgang (= Komparator-Umschaltung) zu Nulldurchgang ermittelt werden. Mit diesem Trick ist der Mikrocontroller in der Lage, die 1200 Baud Packet-Radio Signal zu decodieren.

Die Leitungen der seriellen Schnittstelle des Mikrocontroller sind auf die 9polige Sub-D-Buchse geführt. Hier kann (bei abgezogenem GPS-Empfänger mit derm Mikrocontroller über einen PC kommuniziert werden. Anwedungen wäre z.B. die Änderung des Konfiguration (Rufzeichen, Kommentar-Text,...) welche aber noch nicht vorgesehen ist. Wird dieses nicht genutzt, so besteht über die Bestückung von R21 / Q3 die Möglichkeit, das anschlossene Funkgerät nur zum Senden einzuschalten. Ein umgebautes Betriebsfunkgerät wie das KF 163 von Bosch bietet dazu einen Steuer-Eingang. Die Stromaufnahme kann so im Stand-By-Betrieb drastisch reduziert werden.

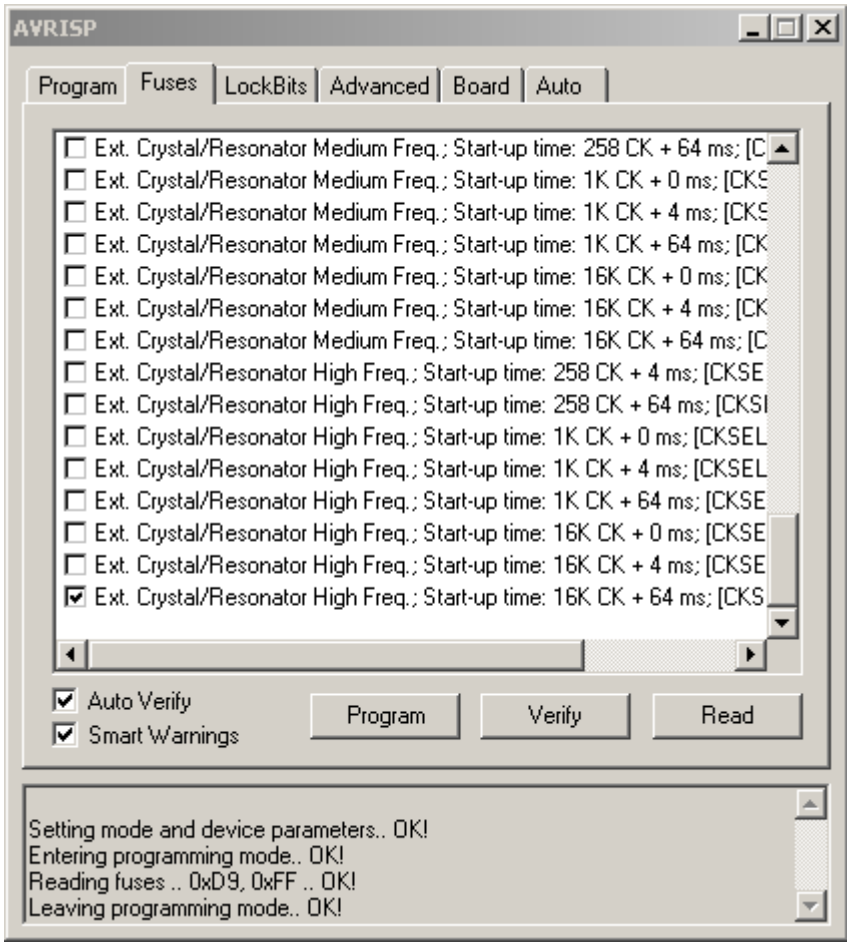
Die Erzeugung der 5 Volt Betriebsspannung wird unspektakulär durch einen 7805-Derivat erzeugt.

Hinweise zum Programmieren des Mikrocontrollers

Das Programm liegt in der ZIP-Datei sowohl als compiliertes Hex-File, also auch als C-Quelltext vor. Zum Quellcode gibt es auch ein AVR-Studio-Projekt, welches bereits alle Dateien und alle Compiler-Einstellungen enthält. AVR-Studio kann man sich kostenlos auf der Homepage von ATMEL herunterladen. Siehe dazu auch der Link auf meiner Homepage. Wer die Software nicht mehr ändern möchte, brennt einfach das Hex-File mit z.B. PonyProg in den AVR. Ansonsten kann die Software zuvor auch nach belieben angepasst werden. Man sei sich aber im Klaren, was man dort macht.

Der fabrikneue AVR hat die Fuse-Bits (Einstellungs-Bits) so gesetzt, dass statt dem Quarz der interne Oszillator läuft. Das ist natürlich nicht gewünscht. Man wähle also z.B. in AVR-Studio den letzten Eintrag aus „External Crystal / High Freq.“ mit der größten Startup-Time. Beim Einschalten haben wir sicher genug Millisekunden Zeit, um die Spannung sich erstmal stabilisieren zu lassen.

Wenn man öfter die Software neu brennen möchte (z.B. beim Debuggen eigener Softwareänderungen), so macht das Aktivieren der Funktion „Preserve EEPROM Memory through the Chip Erase cycle“ sicher Sinn. Sie ist auch bei den Fuse-Bits zu finden. Dadurch wird der EEPROM-Inhalt beim Neu-Beschreiben des Programm-Speichers nicht gelöscht. Dies spart Zeit und EEPROM-Verschleiß.



Hinweise zur Editierung des EEPROM-Hex-Files

Die im ZIP-Verzeichnis enthaltene Datei „APRS_AVR_Tracker_EEPROM_DH3WR.hex“ soll als Vorlage dienen, um das eigene Rufzeichen und eigene Kommentar-Texte einbauen zu können. Zum Editieren empfehle ich das Programm WinHex; der Link ist ebenfalls auf meiner Homepage angegeben. Nach dem Starten wählt man mit „File -> Open...“ die Datei aus. Nun liegt diese im Intel-Hex Format vor. Damit man die Kommentar-Texte im Klartext anzeigen und ändern kann, wähle man „Edit -> Convert...“ , und anschließend „Intel Hex -> Binary“ aus. Die folgende Frage nach der Datei-Größe beantwortet man mit „No“. Nun sieht man den zufünftigen EEPROM-Inhalt vor sich: Links die Adresse, oben den Offset in der jeweiligen Spalte, und rechts die ASCII-Darstellung.

Folgende Tabelle soll den Aufbau der Datei etwas verdeutlichen:

Start-Adresse	***End-Adresse***	***Inhalt***	***Bemerkung***
00	06	82 A0 B4 82 AC A4 60	Ziel-Call „APZAVR-0“ nach AX.25-Codierung
07	0D	88 90 66 AE A4 40 00	Eigenes Call „DH3WR “ mit Leerstelle am Ende (40), da nur 5 Zeichen in DH3WR sind. Die 00

Start-Adresse	***End-Adresse***	***Inhalt***	***Bemerkung***	
			markiert das Ende, da die SSID des eigene Call in der Software selbst hinzugefügt wird. Diese ist von den DIP-Schaltern abhängig und kann weiter unten eingestellt werden. Die Codierung des Calls muss gemäß unten gezeigter AX.25-Tabelle stattfinden.	
0E	14	AE 92 88 8A 62 40 62	1. Pfad-Call „WIDE1 -1“	
15	1B	AE 92 88 8A 64 40 63	2. Pfad-Call „WIDE2 -1“ Laut AX.25-Tabelle ist die Ziffer 1 in der SSID mit 62 zu codieren. Da dies aber das letzte Rufzeichen im Pfad ist, die auf den Wert 1 zu addieren, daher 63.	
1C	1E	03 F0 00	Ende des AX.25 UI Frames, sowie 00-Terminierung	
1F	27	44 48 33 57 52 20 20 20 20	Eigenes Call in ASCII-Darstellung. Wird benötigt, um auf an den Tracker gerichtete Nachrichte reagieren zu können. Es können auch 6 stellige Calls und ggf. SSIDs angegeben werden. Der Bindestrich ist dabei mit zu codieren. Zum Beispiel: DL1ACB-13. Freie Stellen sind mit „Leerzeichen“ (0x20) auf zufüllen.	
28	67	<Status-Text>	Hier kann der Status-Text angegeben werden. Er ist mit 00 zu beenden.	
68	68	32	Faktor für Spannungsmessung. (Sollte nicht viel von 50 = 0x32 abweichen)	
69	69	47	Konfigurations-Byte 1. Bit gesetzt = Funktion aktiv	
			Bit 7	Zeit-Stempel aussenden
			Bit 6	Nachrichten-Fähigkeit anzeigen
			Bit 5	Noch frei
			Bit 4	Noch frei
			Bit 3	TRX aus nach 30 Minuten Stillstand
			Bit 2	Temperatur aussenden
			Bit 1	Betriebsspannung aussenden
			Bit 0	Beim Start auf gültiges GPS warten
6A	6A	60	SSID für Icon 1 im AX.25-Format laut Tabelle	
6B	6B	2F	Symbol-Tabellen-Auswahl für Icon 1 ASCII-Wert von „/“: Primäre Tabelle ASCII-Wert von „\“: Alternative Tabelle	

Start-Adresse	*End-Adresse*	*Inhalt*	*Bemerkung*
6C	6C	3E	Konkrete Auswahl aus obiger Tabelle Icon 1 Hier im Beispiel ASCII-Wert von „>“: Auto
6D	6D	60	SSID für Icon 2 im AX.25-Format laut Tabelle
6E	6E	2F	Symbol-Tabellen-Auswahl für Icon 2 ASCII-Wert von „/“: Primäre Tabelle ASCII-Wert von „\“: Alternative Tabelle
6F	6F	3C	Konkrete Auswahl aus obiger Tabelle Icon 2 Hier im Beispiel ASCII-Wert von „<“: Motorrad
70	D3	<Kommentar-Text 1>	Kommentartext 1. In der rechten Spalte der WinHex-Anzeige kann der Text direkt geändert werden. **Wichtig:** Das letzte Zeichen nach dem Text muss 00 sein.
D4	137	<Kommentar-Text 2>	Kommentartext 2. In der rechten Spalte der WinHex-Anzeige kann der Text direkt geändert werden. **Wichtig:** Das letzte Zeichen nach dem Text muss 00 sein.
138	19B	<Kommentar-Text 3>	Kommentartext 3. In der rechten Spalte der WinHex-Anzeige kann der Text direkt geändert werden. **Wichtig:** Das letzte Zeichen nach dem Text muss 00 sein.
19C	1DF	<Kommentar-Text 4>	Kommentartext 4. In der rechten Spalte der WinHex-Anzeige kann der Text direkt geändert werden. **Wichtig:** Das letzte Zeichen nach dem Text muss 00 sein.

Wichtig: Nach dem letzten Zeichen des jeweiligen Kommentartextes muss 00 stehen. Dadurch weiß die Software, dass der Text hier zu Ende ist. Fehlt das 00, so wird weitergelesen und ungültige AX.25 Pakete entstehen. Im schlimmsten Fall hängt sich die Software auf und der Sender bleibt auf Dauer-Sendung.

Die Werte für die Rufzeichen im AX.25-Format sind in der nachfolgenden Tabelle aufgelistet:

Zeichen	*Wert*	*Zeichen*	*Wert*	*Zeichen*	*Wert*	*Zeichen*	*Wert*
A	0x82	L	0x98	W	0xAE	7	0x6E
B	0x84	M	0x9A	X	0xB0	8	0x70
C	0x86	N	0x9C	Y	0xB2	9	0x72
D	0x88	O	0x9E	Z	0xV4	10	0x74
E	0x8A	P	0xA0	0	0x60	11	0x76
F	0x8C	Q	0xA2	1	0x62	12	0x78
G	0x8E	R	0xA4	2	0x64	13	0x7A

Zeichen	*Wert*	*Zeichen*	*Wert*	*Zeichen*	*Wert*	*Zeichen*	*Wert*
H	0x90	S	0xA6	3	0x66	14	0x7C
I	0x92	T	0xA8	4	0x68	15	0x7E
J	0x94	U	0xAA	5	0x6A	Leerstelle	0x40
K	0x96	V	0xAC	6	0x6C		

Wichtig: Das letzte Rufzeichen im Digipeater-Pfad (hier WIDE5 -5) muss an der 6. Stelle eine Leerstelle (0x40) haben und auf die SSID (5 = 0x6A) muss 1 addiert werden: 0x6B.

Alle Rufzeichen müssen 6 Stellen lang sein (ggf. plus SSID). Ist die 6. Stelle leer, muss hier die Leerstelle (0x40) eingetragen werden.

Die Tabelle für die Icons findet man in der APRS-Protokol-Dokumentation ab Seite 104 „APPENDIX 2: THE APRS SYMBOL TABLES“. Dieses Dokument kann von meiner Homepage heruntergeladen werden.

Wichtig: Beim Auswählen der SSIDs möge man bedenken, dass auch mit der SSID bereits ein Symbol festgelegt wird. Laut APRS-Protokoll überwiegt zwar die Symbol-Definition, die im Daten-Feld gesendet wird, dennoch sollte man, falls nicht zwingende Gründe für die bewusste Verwendung von „falschen „SSIDs (z.B. Durchnummerierung, falls mehrere Tracker unter einem Call betrieben werden), die passende SSID zum Symbol wählen.

Nachdem man die Datei seinen Wünschen angepasst hat, wählt man wieder „Edit -> Convert...“ , und anschließend „Binary -> Intel Hex“ aus. Damit ist die Datei wieder in das Intel-Hex-Format zurückgewandelt und kann nun als „APRS_AVR_Tracker_EEPROM_MYCALL.hex“ gespeichet werden. Für „MYCALL“ kann natürlich das eigene Rufzeichen eingesetzt werden. Diese Hex-Datei schreibt man nun mit PonyProg oder dem AVR-Studio in das EEPROM des AVR.

Hinweise zu Einstellungen des AVR-Studios

Es sind einige Einstellungen in den Projekt-Eigenschaften zu tätigen, wenn nicht die in der Zip-Datei enthaltene Projekt-Datei „APRS_AVR_Tracker.aps“ verwendet werden soll oder kann. Unter „Project -> Configuration Options“ sind eine Reihe von Einstellmöglichkeiten vorhanden. Hier ist als Device „atmega8“, darunter die Quarz-Frequenz von 14745600 Hz und schließlich bei Optimization „-Os“ auszuwählen. Diese Einstellungen sind sehr wichtig und müssen unbedingt vorgenommen werden.

Hinweise zum mechanischen Aufbau

Zuerst sollte der AVR und der TMP275 aufgelötet werden. Mit etwas SMD-Flussmittel lässt sich das auch mit normal großen Lötspitzen bewerkstelligen. Anschließend das restliche SMD-“Vogelfutter“ auflöten. Abschließend die großen Bauteil wie Stiftleiste, Sub-D-Stecker und Mini-DIN-Buchse. Die Platine kann in das empfohlene Gehäuse so eingebaut werden, dass der auf der unterweite befindliche 7805 mit seinem Gehäuse am Gehäuseboden aufliegt. Evtl. ist hier etwas Wärmeleitpaste zwischenzugeben. Die Schraub-Aufnahmen des Sub-D-Steckers können temporär entfernt werden und für die Stifte ein entsprechender Ausbruch an einer der Stirnseiten des Teko-Gehäuses vorgenommen werden. Danach werden die Stifte von innen durchgesteckt und die Metall-

Aufnahme mit ihren Verschraubungen von außen auf das Gehäuse geschraubt werden. So hat die Platine ihren mechanischen Halt. Die Halte-Klammern des Sub-D-Steckers, die in die Platine hineinragen, sollten gut verlötet werden, damit der Stecker mechanisch gut mit der Platine verbunden ist.

Grober Aufbau der Software

Die Software ist in mehreren Modulen aufgebaut, die nun einzeln beschrieben werden.

Main

Das Hauptprogramm. Von hier werden alle Funktionen gesteuert und aufgerufen. Zunächst einmal findet die Initialisierung aller Komponenten statt. Dann wird der Status-Text ausgesendet und danach gewartet, bis die GPS-Maus gültige Positions-Daten liefert. Anschließen geht das Programm in eine Endlosschleife über. Wenn nötig, wird die Position oder der Status-Text gesendet, anschließend werden empfangene Pakete daraufhin überprüft, ob sie eine zu bestätigene Nachricht enthalten. Wenn ja, wird dies getan und ein ggf. definiertes Kommando ausgeführt.

StdDefines.h

Hier sind alle Definitionen zusammengefasst. Die Einträge sein selbsterklärend.

ACD

In diesem Modul wird der AD-Wandler initialisiert. Außerdem befindet sich hier die Interrupt-Routine, welche bei Fertigstellung einer neuen Wandlung das Ergebnis ausließt und zwischenspeichert.

Serial

Die Kommunikation über die ser. Schnittstelle des Mega8 ist hier implementiert. Dazu gehört ins besondere der Empfang von Zeichen der GPS-Maus. Diese werden in einen Puffer geschrieben, und anschließend ausgewertet.

Messaging

Hier geschieht die Daten-Aufbereitung. Die empfangenen NMEA-Datensätze der GPS-Maus werden zerlegt und in die APRS-Pakete eingebaut. Dabei wird ebenfalls die Batterie-Spannung und die Temperatur erfasst und in eine vom Menschen gut lesbare Form gebracht. Nachdem diese Vorbereitungen abgeschlossen sind, kann das APRS-Paket ausgesendet werden, falls dies nötig ist.

I2C_by_hand

Zur Kommunikation mit dem Temperatur-Sensor wird NICHT die intere I2C-Hardware im Mega8 verwendet, sondern eine durch Software definierte Schnittstelle. Diese ist hier zu finden.

TMP275

Wie der Name vermuten lässt ist hier die Kommunikation mit dem dem TMP275 implementiert.

Dazu wird das Modul I2C_by_hand verwendet. Der Sesor wird initialisiert und kann anschließend abgefragt werden.

AX25

Die Erstellung eines AX.25-Pakets und die Aussendung durch Töne geschieht hier.

APRS-Fähigkeiten

Die Software hält sich streng an die in APRS Protocol Reference – Protocol Version 1.0, wie sie auch auf meiner Homepage zu finden ist. Folgende Eigenschaften sind bereits implementiert:

- Positions-Meldung mit Zeitstempel, Kurs, Geschwindigkeit, Höhe und Signalisierung zur APRS-Messaging-Fähigkeit
- Bestätigung von empfangenen Nachrichten.
- Beantwortung von gerichteten Anfragen
  - APRSS: Status der Station
  - APRSP: Position der Station

Anschlussbelegung Funkgerät

Das Funkgerät wird am 9 polige SUB-D-Stecker angeschlossen. Hier sind alle benötigten Signale gebndelt. Die Stromversorgung von Tracker und GPS-Maus geschieht ebenfalls über diesen Anschluss. Die folgende Tabelle gibt Aufschluss über Details.

Pin	*Funktion*	*Bemerkung*
1	+12V	Betriebsspannung +10 Volt DC bis max. +16 Volt DC
2	GND	Masse Betriebsspannung
3	SPK IN	Eingang empfangene PR-Signale
4	PTT	Sende/Empfangsumschaltung. Wird von Transistor auf Masse gezogen.
5	MIC OUT	Ausgang modulierte PR-Signale
6	GND	Masse Betriebsspannung
7	Data-RX	Ausgang des GPS-Empfängers, auch Eingang der ser. Schnittstelle des Mega8. Hierzu muss die GPS-Maus abgezogen und falls bestückt Q2 ausgelötet werden. TTL-Pegel.
8	Data-TX	Ausgang der ser. Schnittstelle des Mega8, TTL-Pegel
9	TRX ON/OFF	Funkgerät einschalten. Wird von Transistor auf Masse gezogen.

Anschlussbelegung GPS-Empfänger

Funktion der DIP-Schalter

Schalter A

Hier kann die zum Senden zu verwendende Konfiguration ausgewählt werden. In der EEPROM-Datei können sowohl das Icon, als auch die SSID eingestellt werden. Ist der Schalter geschlossen, wird Konfiguration 2 verwendet, sonst Konfiguration 1.

Schalter B

Mit diesem Schalter kann eingestellt werden, ob die Software bei Geschwindigkeiten unter 5 km/h (quasi Stillstand) die Senderate reduzieren soll, um das Aussenden von redundanten Daten zu hindern. Die Reduktion erfolgt in der Art, dass bei einer Geschwindigkeit unter 5 km/h noch 4 Mal im alten 30 sec-Takt gesendet und danach in einen 7 Minuten-Takt umgeschaltet wird. Steigt die Geschwindigkeit wieder auf 5 km/h oder mehr, so wird zurückgeschaltet und das Spiel beginnt von neuem. Ist der Schalter geschlossen, so wird immer im 30 sec-Takt gesendet.

Schalter C / D

Mit diesen Schaltern kann ausgewählt werden, welcher Kommentartext gesendet wird. Die Kodierung des Schalter erfolgt binär.

Schaltplan

TBD

Layout

TBD

Bestückungsplan

TBD

Auflistung des Bauteile und deren Bestell-Bezeichung bei Reichelt Elektronik

Best.-Nr.	Bezeichnung	Menge	Bemerkung
14,7456-HC49-SMD	Quarz SMD	1	
SMD-0805 1,00K	Widerstand	6	
SMD-0805 3,90K	Widerstand	1	
SMD-0805 8,20K	Widerstand	1	
SMD-0805 220	Widerstand	1	
SMD-0805 2,20K	Widerstand	3	
SMD-0805 10,0K	Widerstand	1	
SMD-0805 3,30K	Widerstand	1	
SMD-0805 33,0K	Widerstand	1	
SMD-0805 100K	Widerstand	1	
EB-DIO M06	PS2-Buchse	1	
X7R-G0805 100N	Kondensator 100nF	8	
NPO-G0805 15P	Kondensator 15pF	2	
NT 04-SMD	DIP-Switch	1	
LED 3mm rt	LED rot	1	
LED 3mm gn	LED grün	1	
LED 3mm ge	LED gelb	1	
BC 847C SMD	Transistor	3	
ATMEGA 8-16 TQ	AT MEGA 8 16 MHz	1	
SMD ELKO 22/16	Elko 22 µF	1	
SMD ELKO 10/35	Elko 10 µF	1	
SL 2x10G 2,54	Stiftleiste	1	
D-SUB ST 09US	Sub-D 9 pol kurze Ausf.	1	
SMD-0805 4,70K	Widerstand	2	Nur nötig für GPS-Inverter
SMD-0805 47,0K	Widerstand	1	Nur nötig für GPS-Inverter
SMD 1N 4148	Diode	1	Nur nötig für GPS-Inverter
TEKO A1	Gehäuse 38x72x28mm	1	Passt zur Platine
64Y-10K	Trimmer 10k	1	

Außerdem benötigt: TMP275 im MSOP-8 Gehäuse und ua7805 im TO-263 Gehäuse.

Verweis auf den Ursprung des Projekts

Der Ursprung des Projekts ist das WhereAVR-Projekt von N4TXI. Von ihm stammen die Implementierung der AX.25 Kanalcodierung und die AX.25 Empfangsroutinen, sowie die Grundcodierung der APRS-Frames. An vielen Stellen wurde die Schalung geändert oder erweitert, so dass eine klare Trennung nicht mehr möglich ist.

Danksagung

Dank sagen möchte ich Christian Jansen DF6EF für die Mitarbeit und Bereitstellung seines Messplatzes, Frank Ulbrich DO2FU für die Software für den Temperatur-Sensor und Hans-Peter DL9EBF für die zahlreichen Hinweise beim Testen der Software.