Antennenanalyser mit Arduino

Beschreibung

1. Allgemein

Der Antennenanalyser besteht aus einer Ohmschen SWR Messbrücke('Resistive SWR Bridge'), die von einem DDS- Generator gespeist wird. Die Einstellung der Frequenz und die Messung der Brückenspannung erfolgt über einen Arduino - Mikrocontroller. Ein PC dient zur Anzeige und Bedienung.

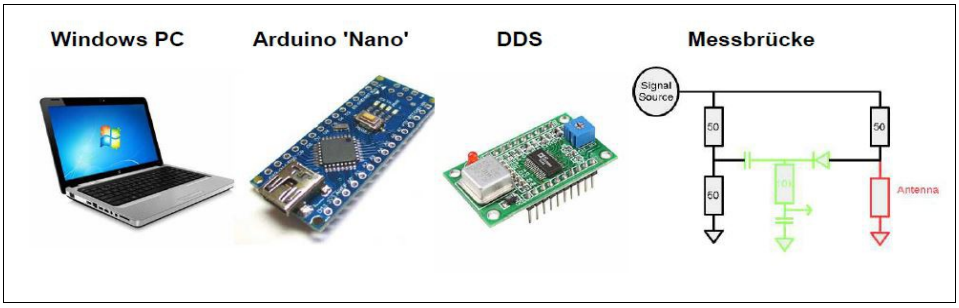


Abb. 1: Komponenten [Ref.1]

2. Anwendungen

- SWR- Messung 1 bis 30 Mhz
- Signalgenerator bis 30Mhz
- Resonanzmessung
- Telegraphie - Bakensender

3. Schaltungsbeschreibung

Die Frequenz des DDS AD9850 wird über 4 Steuerleitungen vom Mikrocontroller eingestellt. Die Frequenz kann auf 1 Hz genau programmiert werden. Das Ausgangssignal ist nahezu sinusförmig. Der DDS liefert einen Pegel von ca. 400mVss. Dieses Signal wird durch den DDS- Verstärker mit den Transistoren Q1 & Q2 verstärkt und auf ca. 1Vss an 50 Ohm angehoben.
Die Messung der Brückenspannung erfolgt über zwei Messkanäle. Jeder Messkanal besteht aus einer Spitzenwertgleichrichtung mit nachfolgendem logarithmischen Verstärker, welcher mit der gleichen Dioden bestückt ist wie der Gleichrichter; so wird die nichtlineare Kennlinie des Spitzenwertgleichrichters kompensiert. Es werden Germaniumdioden (AA1xx, OAxx, 1N34, 1N60P) verwendet, da Siliziumdioden bei dem geringen Pegel noch keine ordentliche Spannung liefern. Im Prototyp werden 1N60P verwendet. Der zweite OP verstärkt das Signal auf ca. 500mV am Eingang des Analog-Digital-Wandlers des Mikrocontrollers. Um bei den kleinen Spannungen im mV - Bereich die Offsetfehler klein zu halten, sind die OPs möglichst symmetrisch beschaltet. Als OP genügt ein LM324, der bei + 5Volt und 0-Volt- Versorgung Ausgangsspannungen von 0 Volt bis ca. 3 Volt Ausgangsspannung liefert.

4. Aufbau

Alle Bauelemente befinden sich auf einer Platine 100mm x 75mm, die in die Führungsschienen eines Profilgehäuses passt (Fischer Frame Gehäuse FR 80 42 100 ME [Ref.6]). Das DDS- und Arduino - Nano- Modul sind in IC- Sockel mit doppeltem Federkontakt gesteckt. Für die Bestückung von Hand hat die Platine extra große Pads mit Durchkontaktierungen. Die Platine wird durch die BNC-Buchse an der Front festgehalten.

5. Bestückung

- Arduino- Board vorbereiten: Das Arduino- Modul hat je nach Lieferung evtl. noch keine Stiftleisten; Dazu werden ein passender IC- Sockel, die mitgelieferten 15-poligen Stifte und die Nano-Board-Platine zusammengesteckt und die Stifte oben an 4 Ecken verlötet. Die Stifte werden senkrecht ausgerichtet und endgültig verlötet. Die 6-pol. Stiftleiste auf dem Nano-Board (ISP-Stecker) braucht nicht bestückt werden.
- IC-Sockel für Nano-Board vorbereiten: Aus einem 40-poligen IC-Sockel wird ein 2 x 15-poliger geschnitten; bei älteren Platinen gibt es an der 'Nord-Ost-Seite' für 3 Pins des Nano-Boards keine Pads, daher müssen die zugehörigen Stifte des Sockels flachgelegt werden. Zunächst IC-Sockel an 4 äußeren Pins fixieren und dann endgültig anlöten.

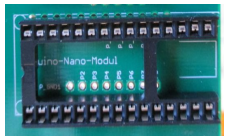


Abb. 2: Arduino-Nano-Sockel

- IC-Sockel für das DDS-Board vorbereiten: Ein 20-poliger IC-Sockel wird in der Mitte zu 2 x 10 Pins längs geteilt. Nun das DDS-Board auf die Kontaktreihen stecken und auf der Platine platzieren. Den IC-Sockel an 4 äußeren Pins fixieren und dann endgültig anlöten.
- Entfernen des DDS - Moduls und des Nano-Boards.
- Bestückung der restlichen Bauelemente, wobei mit den niedrigsten begonnen wird. Falls die Bauelemente nicht sortiert sind, erst messen und sofort an passender Stelle einlöten. Bei den Dioden Polarität beachten!

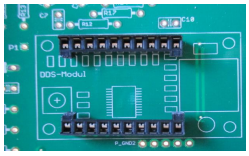


Abb. 3: DDS-Sockel

- Einlöten der BNC - Buchse; hierbei darauf achten, dass diese rechtwinklig auf der Platine sitzt.
- Zuletzt den OP in den 14-poligen IC_Sockel stecken.
- DDS - Modul und Nano-Board wieder stecken.

6. Verbindung Aufnehmen

Was wird benötigt?

- USB-Kabel mit USB Mini-Stecker.
- Arduino Programm *'dds_sweeper_xx.ino'* [Ref.3]
- Programm 'VNA.EXE' mit zugehörigen DLLs [Ref.2]
- Microsoft Runtime Library DotNet 4.x
- Arduino IDE z.B. Arduino 1.6.x [Ref.5]

Die Module „DDS" und „Arduino Nano" sind gesteckt. Nun wird der USB-Port mit dem Rechner verbunden. Jetzt sollte am Nano-Board die grüne und am DDS- Board die rote Power-LED leuchten. Falls der PC das Board nicht erkennt, so ist die Installation eines Treibers erforderlich [Ref. 3].
Nun wird die Arduino - IDE aufgerufen. Als erstes wird das Board 'Nano' ausgewählt und der COM -Port eingestellt.

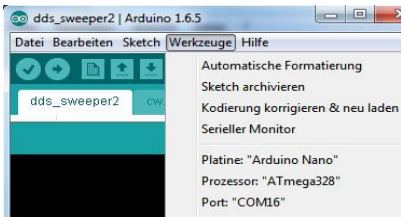


Abb. 4 COM-Port

Zum Überprüfen, welcher COM- Port vom Arduino belegt wird, kann man im Gerätemanager nachsehen („*devmgmt.msc*"); dort nach „*USB -Serial CH340*" suchen.

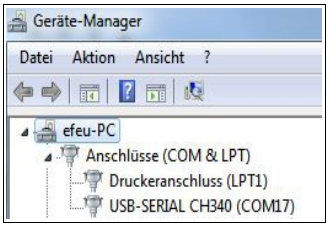


Abb. 5: Gerätemanager

7. Erster Test

Arduino mit Programm *'dds_sweeper2.ino'* [Ref.3] laden. Mit dem Knopf 'Hochladen' (Pfeil nach rechts) oder 'Strg+U' wird ein Programm in den Arduino - Mikrocontroller geladen. Nach dem Hochladen rufen wir den seriellen Monitor des Arduino auf über 'Werkzeuge' -> 'serieller Monitor'. Die Baudrate muss auf 57600 stehen! Nun geben wir in der Eingabezeile des Monitors ein

`'7032000c'` (d.h. 7,032 Mhz als Dauersignal)

Das Programm sendet nun ein Dauersignal auf 7,032 Mhz. Auf dieser Frequenz 7,032 Mhz sollte jetzt ein Träger hörbar sein. Evtl. die Frequenz nach eigenen Bedürfnissen einstellen.

8. Offset prüfen

Zum Prüfen des Offsets geben wir die Frequenz Null ein

`'0a0b10n'` (d.h. Von 0 Mhz bis 0 Mhz in 10 Schritten)

Mit

`'s'`

wird der Durchlauf (Sweep) gestartet. Die Anzeigewerte für Vorlauf und Rücklauf in der 4. und 5. Spalte sollten nahe Null sein (< 40). Ohne Eingangssignal wird nur der Offset gemessen;

9. Funktionstest 'dds-sweeper'

Wir nehmen eine KW-Empfänger und stellen ihn auf eine Frequenz ein (Beispiel: 7,032 Mhz). Als Betriebsart wählen wir CW oder SSB. Jetzt tippen wir am Terminal („Arduino – Serial - Monitor") ein:

`'7032000c'` *(d.h. 7,032 Mhz als Dauersignal)*
`'?'` *(d.h. Statusanzeige)*

Die Eingabe erfolgt in „Hertz" . Das Kommando 'c' stellt diese Frequenz ein und das Kommando '?' zeigt die aktuellen Einstellungen auf dem Bildschirm an. Im KW-Empfänger sollte jetzt ein Signalton hörbar sein; vielleicht kann man einen kurzen Draht als Antenne in die Koaxbuchse als „Sendeantenne" stecken. Wir können auch den Bereich, der durchlaufen wird, ausprobieren: (Die Eingaben werden immer mit der Enter - Taste abgeschlossen). Wir tippen ein:

`'7000000a'` *Startfrequenz = 7.000 MHz*
`'7001000b'` *Endfrequenz = 7.001 MHz*
`'?'` *Status Anzeige*

Jetzt 's' eingeben: Das Terminal zeigt die durchlaufenen Frequenzen an.

Im KW-Empfänger müsste jetzt ein ansteigender oder abfallender Ton hörbar sein – je nach eingestelltem Seitenband.

Noch ein Experiment: Wir ändern die Schrittanzahl ('Num Steps') auf 10, indem wir eingeben:

`'10n'` *(d.h. 10 Schritte)*

Wenn wir jetzt wieder 's' eingeben, sollte man im KW-Empfänger deutlich die Frequenzschritte hören können. Wir haben bis jetzt nachwiesen, dass der DDS Signale erzeugt.

Jetzt schließen wir wieder den 50-Ohm-Widerstand an. Eine Ausgabe von Daten erfolgt mit dem Kommando 's' am Terminal („Arduino – Serial - Monitor") :
Die 3. Spalte gibt den SWR - Wert mal 1000 wieder; es sollten also bei einem 50-Ohm Abschlusswiderstand Werte etwas größer als 1000 entsprechend einem SWR von 1.0 angezeigt werden.

```
7000000.00, 0, 1017.33935, 444, 4
7000100.00, 0, 1012.76232, 444, 3
7000200.00, 0, 1012.79138, 443, 3
7000300.00, 0, 1012.76232, 444, 3
7000400.00, 0, 1017.37890, 443, 4
7000499.50, 0, 1012.76232, 444, 3
7000600.00, 0, 1012.76232, 444, 3
7000700.00, 0, 1012.76232, 444, 3
7000800.00, 0, 1017.33935, 444, 4
7000900.00, 0, 1012.79138, 443, 3
7001000.00, 0, 1008.24334, 442, 2
End
```

Abb. 6: Sweep

Die Tests jetzt mit den anderen Widerständen wiederholen:

100 Ohm	SWR= 2,0	25 Ohm	SWR= 2,0
75 Ohm	SWR= 1,5	33,3 Ohm	SWR= 1,5
150 Ohm	SWR= 3,0		

Die Werte sollten ungefähr eingehalten werden. An dieser Stelle muss betont werden, dass dieser 'Antennenanalyser' kein Präzisionsinstrument ist.

10. Bedienoberfläche VNA.exe

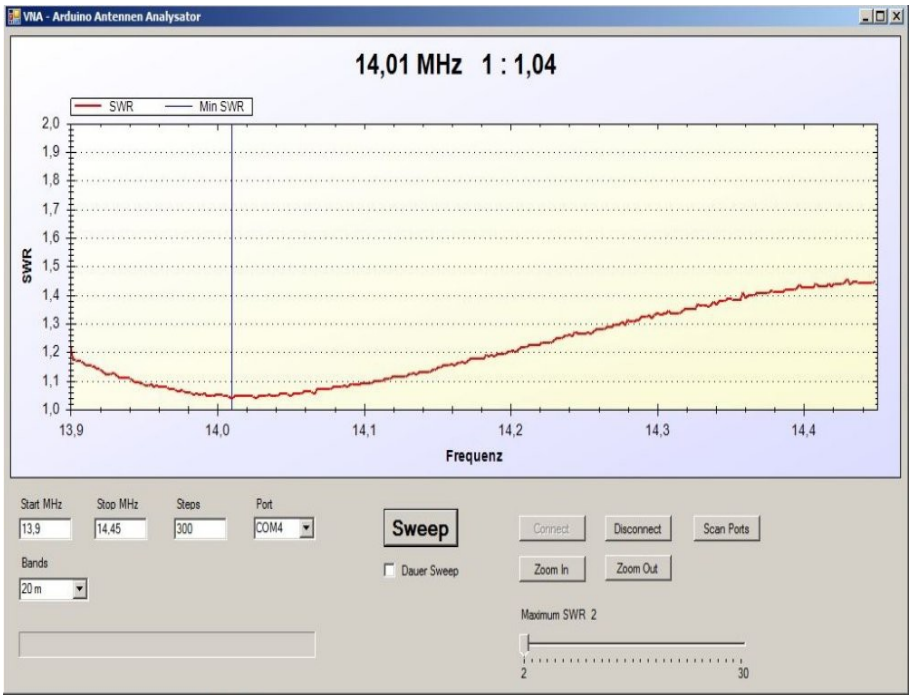


Abb. 7: VNA.exe [Ref.2]

Die Software ist unter DotNet programmiert und benötigt eine aktuelle Version des DotNet Frameworks (Version 4.0 oder höher). Dieses kann kostenlos von Microsoft herunter geladen werden. Ein Software Installation ist nicht erforderlich. Es reicht, das Verzeichnis mit den Programmdateien an eine beliebige Stelle zu kopieren und die Datei VNA.exe zu starten. Nach dem Programmstart wählt man unter Port die Com - Schnittstelle des USB – Adapters aus. Mit einem Klick auf Connect wird eine serielle Verbindung zum Arduino Modul hergestellt. Unter Start MHz und Stop MHz ist der Frequenzbereich 1 .. 30 MHz voreingestellt. Diesen kann man beliebig ändern oder mit der Auswahlbox Bands ein Amateurfunkband wählen. Durch einen Klick auf Sweep werden die Daten an das Arduino Modul gesendet, dieses antwortet mit den entsprechenden SWR - Werten. Aus diesen wird dann eine Verlaufskurve gezeichnet. Unter Steps ist als Wert 101 eingestellt. Dies bedeutet, dass bei einem Frequenz - Sweep 101 Frequenzen gemessen werden. Man kann diesen Wert bis auf 1000 erhöhen, dann bekommt man eine höhere Auflösung mit bis zu 1000 Messpunkten – die Messung dauert dann aber etwas länger. Die Auflösung erhöht sich auch, wenn man den Frequenzbereich verkleinert, z.B. statt 1 .. 30 MHz >> 13 .. 15 MHz. Dies erreicht man auch

durch Drücken der Tasten Zoom In und Zoom Out und dann Sweep. Die Grafik ist auf einen SWR Bereich von 1 .. 10 voreingestellt. Mit dem Schieberegler Maximum SWR kann man den Wert zwischen SWR 2 .. 30 zoomen. Zusätzlich kann man die Grafik auch zoomen, indem man mit der linken Maustaste einen Rahmen um einen Teilbereich zieht. Wenn man mit der rechten Maustaste auf die Grafik klickt, kann man die aktuelle Grafik in eine Datei speichern. Dies ist für eine spätere Auswertung sehr praktisch. Darüber hinaus bietet das Auswahlmenü auch noch einige zusätzliche Optionen.(Quelle: Programm und Text von Norbert Redeker DG7EAO [Ref.2])

11. Messbeispiele

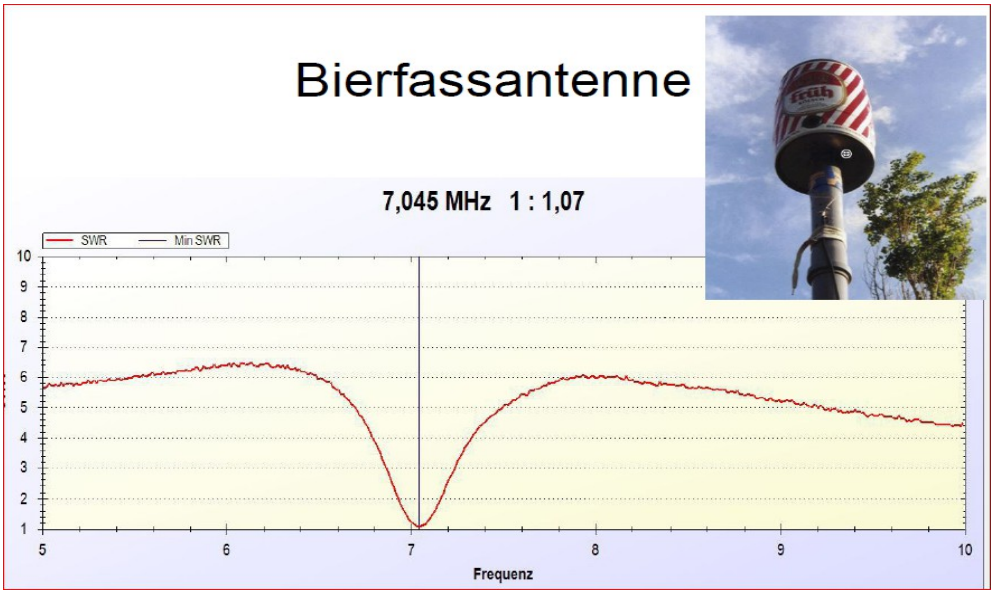


Abb. 8 „Bierfassantenne" (Eigenbau)

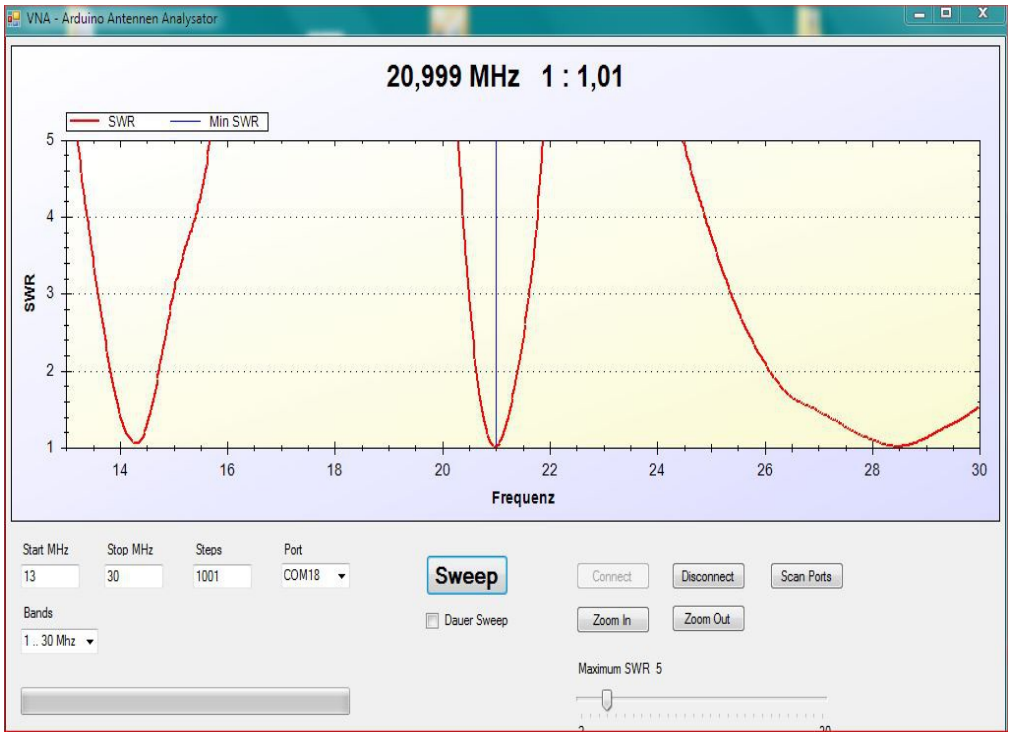


Abb. 9 GPA-3 (Fritzel)

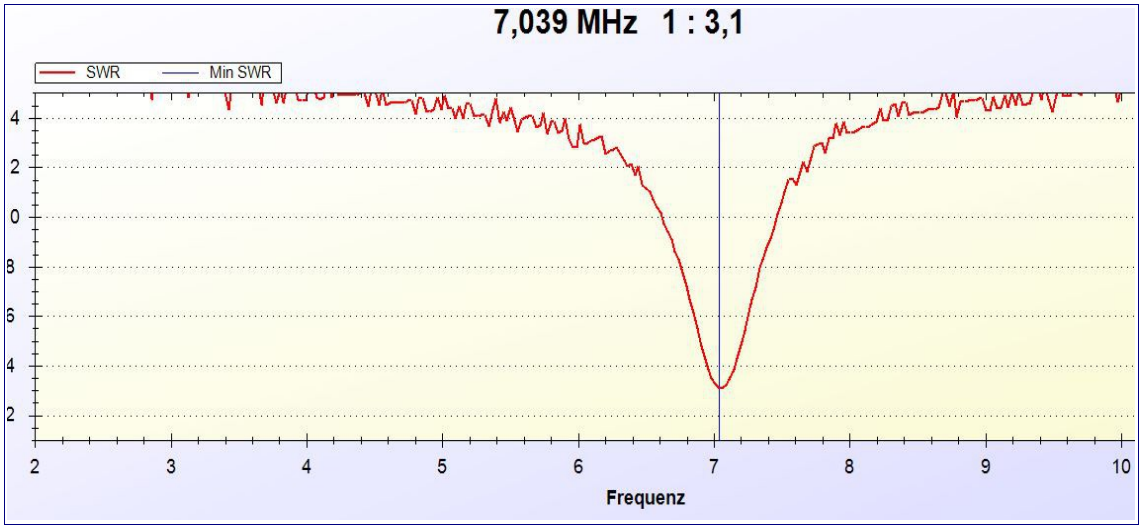


Abb. 10 Sperrkreis aus W3DZZ ; mit Linkleitung angekoppelt

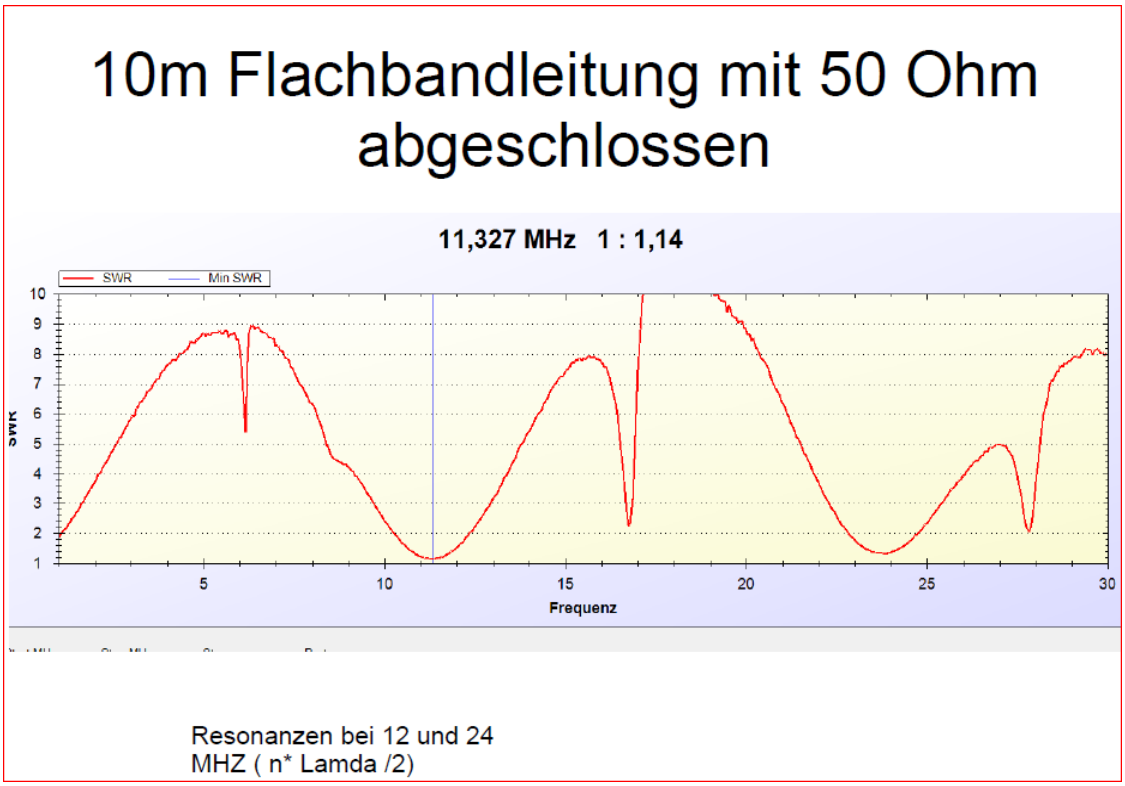


Abb. 11 Flachbandleitung , Verkürzungsfaktor =0,85

12. Gimmicks

Als Spielerei wurde noch ein Bakensender implementiert. Ein im EEPROM gespeicherter Text wird auf der Startfrequenz wiederholt als Morsezeichen gesendet. Der Text wird mit Ausrufezeichen beginnend und mir Enter endend eingegeben. Der Text ist nun im EEPROM gespeichert. Mit dem Kommando '#' wird dieser Text gesendet. Die Sendefrequenz wird z.B. mit 7032000c eingegeben (7,032 Mhz). Zum Beenden der Bakensendung muss wieder '#' eingegeben werden.

```
Gimmick:
'!'     store following text in EEPROM for beacon tx, end with <enter>
'#'     send beacon tx text again and again ( break with '#' )
        Beacon frequency is start frequency ( command 'a' or 'c' )
actual beacon text: " de dk2jk"
```

Abb. 12: Gimmicks

13. Befehlsliste

```
---- commands---
letter  description             example         result
'a'     set start frequency     6000000a        6.000 Mhz
'b'     set end frequency       8000000b        8.000 Mhz
'c'     set constant frequency  7032000c        7.032 Mhz
'n'     set numer of steps      100n            100 steps
's'     perform sweep from frequency a to b with n steps
'h'     help or '?'
commands may be lower or upper case      e.g. 'a' or 'A'
```

Abb. 13: Befehle

14. Quellen

[Ref.1] K6BEZ:
http://www.hamstack.com/hs_projects/antenna_analyzer_docs.pdf

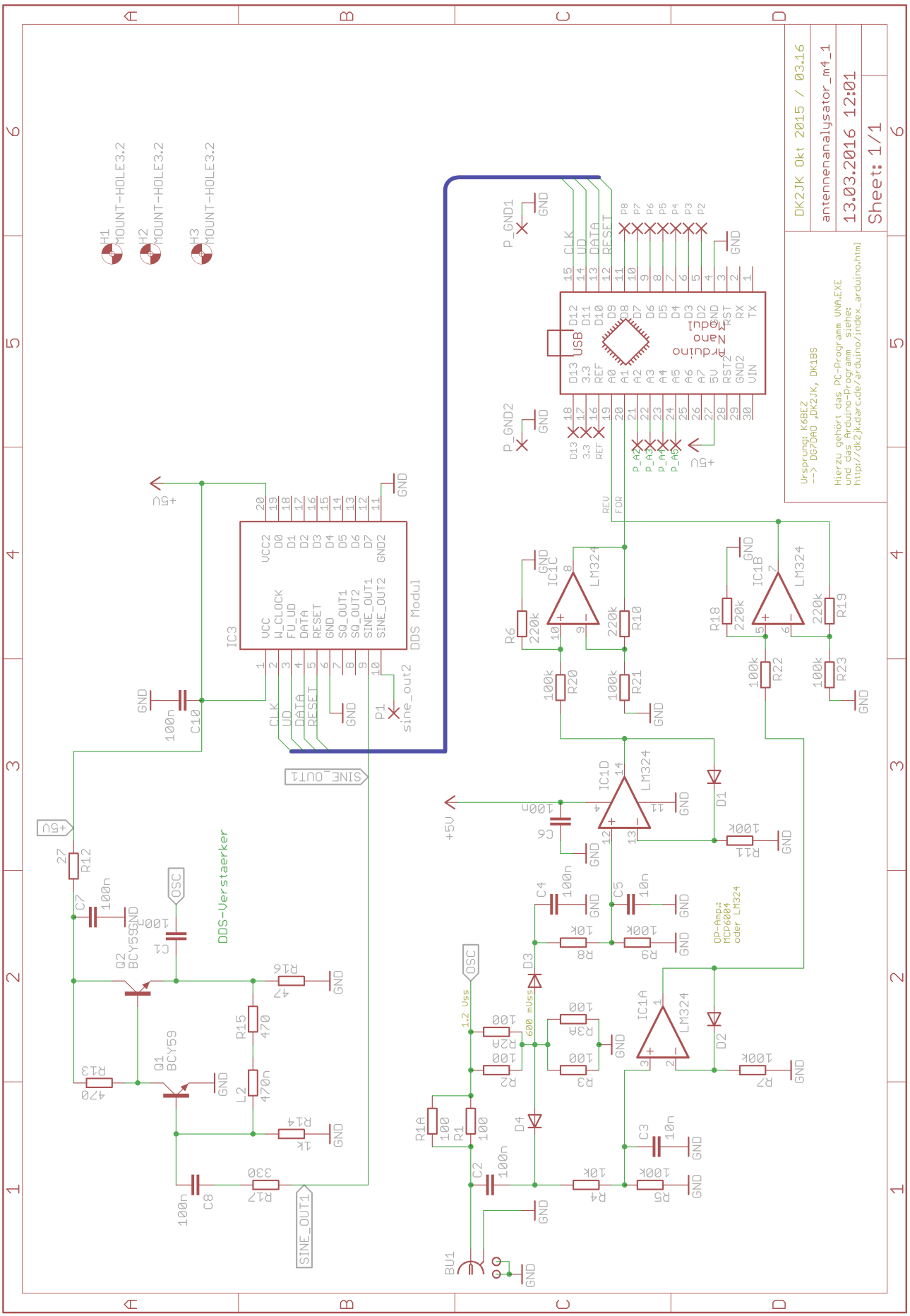
[Ref.2] DG7EAO :
http://lima05web.wordpress.com/2014/03/22/arduino-antennen-analysator-dg7eao/

[Ref.3] DK2JK:
http://dk2jk.darc.de/arduino/index_arduino.html (→ Prototyp 'm4')
http://dk2jk.darc.de/arduino/antennenanalyser/CH341SER.ZIP (Treiber für CH340)

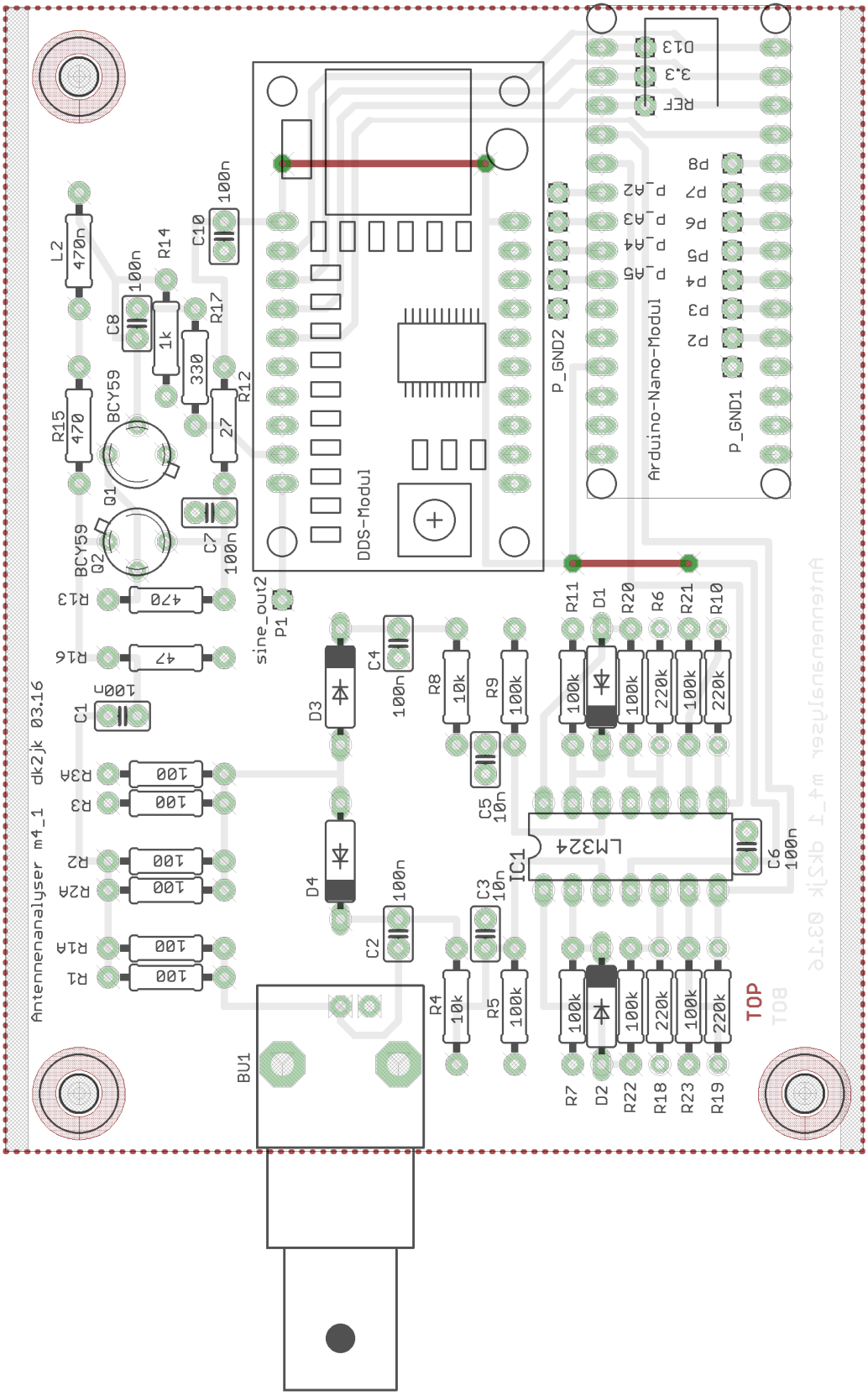
[Ref.4] DDS:
http://www.analog.com/static/imported-files/data_sheets/AD9850.pdf

[Ref.5] Arduino:
http://www.arduino.cc/

[Ref.6] Gehäuse:
http://www.fischerelektronik.de/
http://www.schubert-gehaeuse.de/



Anhang 1: Schaltplan



Anhang 2: Bestückung

Position	Anzahl	Bauteil	Wert	Beschreibung
1	1	BU1	BNC	UG 1094W1
2	7	C1	100n	X7R-2,5 xx
		C2	100n	X7R-2,5 xx
		C4	100n	X7R-2,5 xx
		C6	100n	X7R-2,5 xx
		C7	100n	X7R-2,5 xx
		C8	100n	X7R-2,5 xx
		C10	100n	X7R-2,5 xx
3	2	C3	10n	X7R-2,5 xx
		C5	10n	X7R-2,5 xx
4	4	D1	Ge- Diode	1N34, 1N60P , AA119, , OAxx
		D2	Ge- Diode	1N34, 1N60P , AA119, , OAxx
		D3	Ge- Diode	1N34, 1N60P , AA119, , OAxx
		D4	Ge- Diode	1N34, 1N60P , AA119, , OAxx
5	1	IC1	LM324	OP AMP evtl. MCP 6004
6	1	L2	470nH	SMCC 0,47uH evtl. Luftspule 7mm=L, 4mm=D, 16=N
7	2	Q1	BCY59	NPN TRANSISTOR Universal BC337, BCY59 o.ä.
		Q2	BCY59	NPN TRANSISTOR Universal BC337, BCY59 o.ä.
8	6	R1	100	Widerstand Metallschicht 0207 1%
		R1A	100	Widerstand Metallschicht 0207 1%
		R2	100	Widerstand Metallschicht 0207 1%
		R2A	100	Widerstand Metallschicht 0207 1%
		R3	100	Widerstand Metallschicht 0207 1%
		R3A	100	Widerstand Metallschicht 0207 1%
9	8	R5	100k	Widerstand Metallschicht 0207 1%
		R7	100k	Widerstand Metallschicht 0207 1%
		R9	100k	Widerstand Metallschicht 0207 1%
		R11	100k	Widerstand Metallschicht 0207 1%
		R20	100k	Widerstand Metallschicht 0207 1%
		R21	100k	Widerstand Metallschicht 0207 1%
		R22	100k	Widerstand Metallschicht 0207 1%
		R23	100k	Widerstand Metallschicht 0207 1%
10	2	R4	10k	Widerstand Metallschicht 0207 1%
		R8	10k	Widerstand Metallschicht 0207 1%
11	1	R14	1k	Widerstand Metallschicht 0207 1%
12	4	R6	220k	Widerstand Metallschicht 0207 1%
		R10	220k	Widerstand Metallschicht 0207 1%
		R18	220k	Widerstand Metallschicht 0207 1%
		R19	220k	Widerstand Metallschicht 0207 1%
13	1	R12	27	Widerstand Metallschicht 0207 1%
14	1	R17	330	Widerstand Metallschicht 0207 1%
15	1	R16	47	Widerstand Metallschicht 0207 1%
16	2	R13	470	Widerstand Metallschicht 0207 1%
		R15	470	Widerstand Metallschicht 0207 1%
17	1	Gehäuse	80x100	FISCHER Frame-Gehäuse FR 80 42 100 ME
18	1	Platine	100 x 75	W09973AS1 / 3PCB
19	1	IC-Sockel	40-polig	IC-Sockel, 40-polig, doppelter Federkontakt , auf 2x 15 gekürzt
29	1	IC-Sockel	20-polig	IC-Sockel, 20-polig, doppelter Federkontakt ,in 2x 10 geteilt
30	1	IC-Sockel	14-polig	IC-Sockel, 14-polig, doppelter Federkontakt ,
31	1	ARDIUNO	ARDUINO	Arduino Nano Modul
32	1	DDS	DDS	AD9850 DDS Module / China

Anhang 3: Stückliste