Der digitale Drehko

Mittelwelle:

L = 30 uH

0.5 MHz → 3.377 nF
1.6 MHz → 330 pF

C = 0.33nF … 3.3 nF
also 330pF fest + 3 nF variabel

Delta-F : 1.6 – 0.5 = 1.1 MHz

für eine Auflösung von 10 kHz: 1.1 MHz / 10 kHz = 110 steps

mit 7 Bit = 128 steps; also 7 D-Outputpins am Nano für die Generierung des variablen C-Wertes einsetzen.

Es ist ggf. zweckmäßig, für den „Drehko“ einen Nano extra zu verwenden, weil sonst ein Engpass für die Ports an einem einzelnen Arduino entstehen könnte.

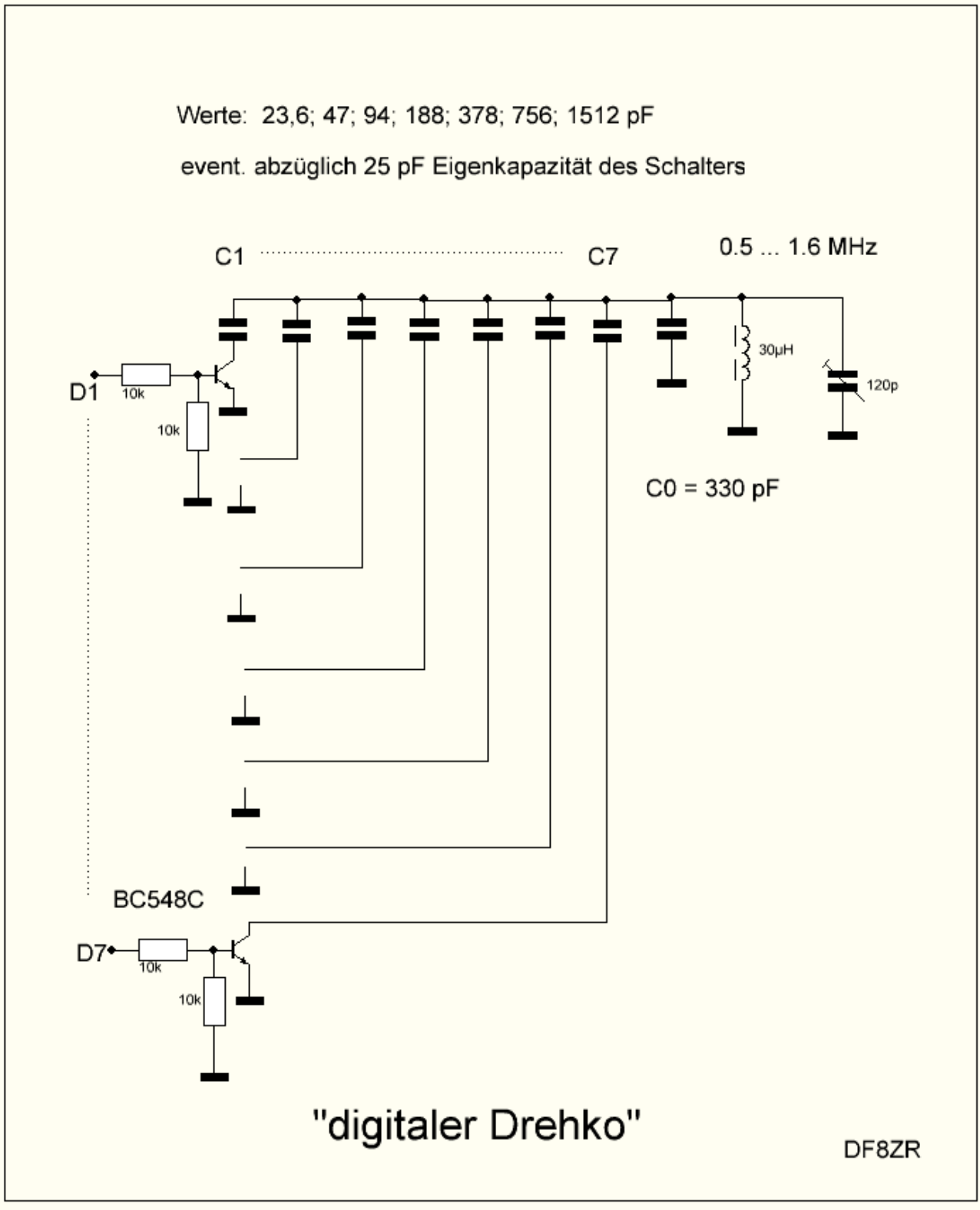
1 100 kHz / 127 = 8.66 kHz/Step

3000 pF / 127 = 23.6 pF / Step

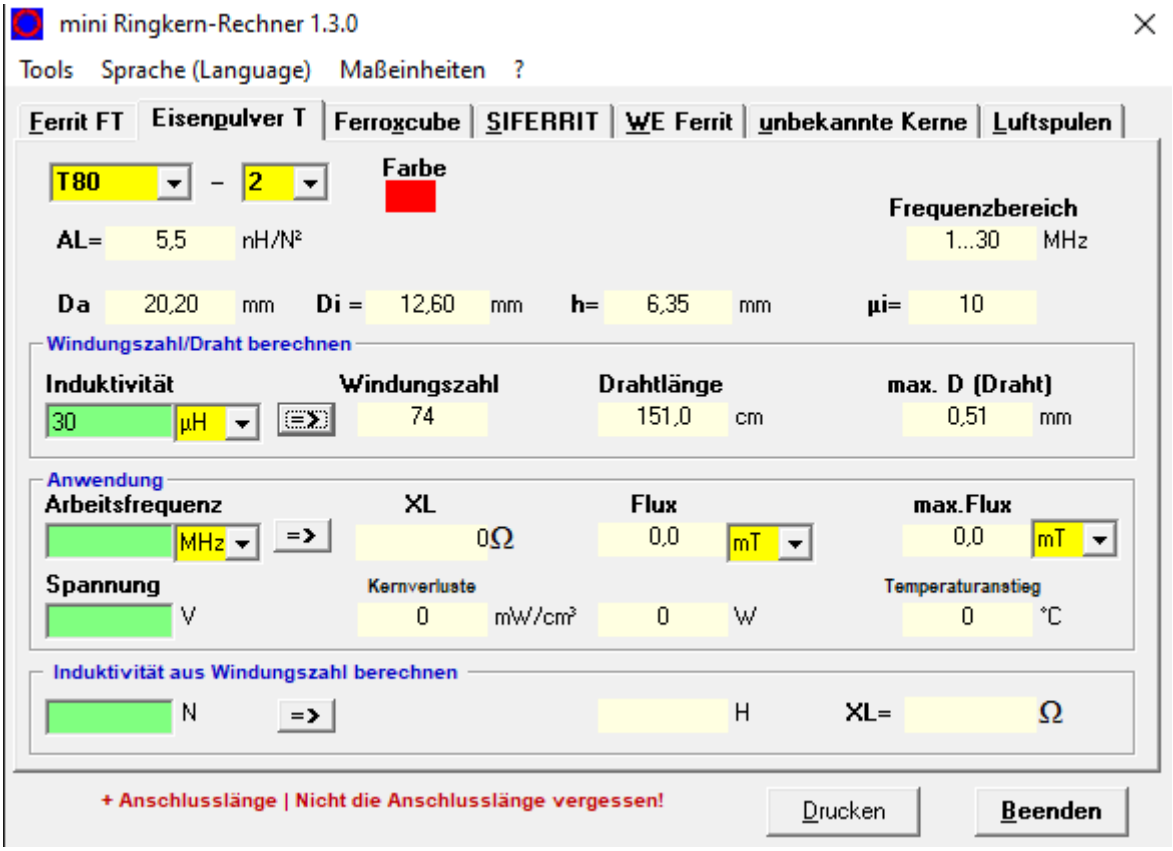
Werte: 0, 23,6; 47; 94; 188; 378; 756; 1512 pF

davon müssen event. Jeweils 25 pF Eigenkapazität des Schalters subtrahiert werden

Der Anfangswert ist 330 pF, die variable Kapazität also 3 nF
2^7 steps = 127



So ein Gebilde muss messtechnisch untersucht werden. Also zunächst mal die Resonanzkurven aufnehmen.



74 Wdg mit dem Kern T80-2 von Amidon für 30uH.

Software

Die Software wurde so geschrieben, dass man mit einem seriellen Schieberegister CD164 die 7 Bits ausgibt und schiebt. Wie gut das mit der o.g. Schaltung funktioniert, zeigt das folgende Beispiel. Es wurde eine Kapazität von 2584 pF vorgegeben:

```
2584
c7 = 1512
c6 = 756
c4 = 188
c3 = 94
c1 = 24
rest = 10
```

Die fehlende Restkapazität kann bis 24 p(Stufenwert) betragen. Das ist aber nicht relevant, weil die Resonanzkurve nicht sehr selektiv ist. Durch einen Rückkopplungsregler wird im Betrieb die Feinabstimmung vorgenommen.

Hier der Sketch für den Testbetrieb:

```
/*
"Digitaler Drehko"
Mit geschalteten Kondensatoren wird ein LC-Schwingkreis abgestimmt. Weil man
dazu > 7 OutputPINs braucht, wird hierfür einSchieberegister CD 164 eingesetzt.
Die Kondensatoren werden mit MOSFETs geschaltet. Die Kondensatoren sind relativ groß bemessen,
damit sich die Streukapazitäten gering auswirken. Hier wird ein Anstimmbereich von
500kHz ... 1.6 MHz vorgesehen. Die Resonanz bleibt dabei zunächst noch breit. Durch eine Rückkopplung
kann sie selektiv werden. Außerdem steht eine kapazitive Feinabstimmung mit C-Dioden zur Verfügung.
*/

/*
Es müssen Variable vereinbart werden: */

int capa = 2584;   //Übernahme der Kapazität; hier ein Testwert
                   //für Kapazität

int capalt;        //Zwischenspeichern

int cap;           //aktueller Wert

int rest;          //nur für den Test

//die Werte für das MW-Radio
int c7 = 1512;
int c6 = 756;
int c5 = 378;
int c4 = 188;
int c3 = 94;
```

```
int c2 = 47;
int c1 = 24;


void setup() {

 // Öffne die serielle Schnittstelle bei 9600 Bit/s:
  Serial.begin(9600);

//Ausgänge definieren als Output:

pinMode(7, OUTPUT); //D4  //als Date

pinMode(8, OUTPUT); //D5  //als CLK

pinMode(9, OUTPUT); //D6  /als Clear


//alle Ausgänge auf Null setzen
digitalWrite(7, LOW);
digitalWrite(8, LOW);

digitalWrite(9, LOW);  //Clear
delay (1);

digitalWrite(9, HIGH);  //Grundzustand herstellen, Clear = HIGH

}

void loop() {

cap = capa;  //Kapazität falls unverändert

//Auflösung für das MW-Radio 0.5MHz ... 1.6 MHz ist 1.1
MHz/127 = 8,66 kHz
```

```
//in solchen steps wird die Ansteuerung geschaltet

//Grundzustand des Schieberegisters herstellen; auf NULL setzen
digitalWrite(9, LOW);  //Clear
delay (1);
digitalWrite(9, HIGH);  //Grundzustand herstellen

//falls cap ungleich capalt, dann nach schalten()
//hier noch Code schreiben!!!

schalten();   // jetzt zur Ausgabe


}


void schalten() {

//hier werden die digitalen Ausgänge aktiviert, d.h. auf "1" gesetzt
//zuvor waren alle auf LOW gesetzt

Serial.println(cap, DEC);  // Drucke im Testbetrieb


if (cap > c7) {digitalWrite(7, HIGH);
cap = cap - c7;

Serial.print("c7 = ");
Serial.println(c7, DEC);  // Drucken im Test

}


impuls();
```

```
if (cap > c6)
{
  digitalWrite(7, HIGH);
cap = cap - c6;

Serial.print("c6 = ");
Serial.println(c6, DEC);  // Drucken
}

impuls();

if (cap > c5) {digitalWrite(7, HIGH);
cap = cap - c5;
Serial.print("c5 = ");
Serial.println(c5, DEC);  // Drucken
}

impuls();

if (cap > c4) {
  digitalWrite(7, HIGH);

cap = cap - c4;

Serial.print("c4 = ");
Serial.println(c4, DEC);  // Drucken
}

impuls();

if (cap > c3){
  digitalWrite(7, HIGH);

cap = cap - c3;
```

```
Serial.print("c3 = ");
Serial.println(c3, DEC);  // Drucken
}

impuls();

if (cap > c2){
  digitalWrite(7, HIGH);

cap = cap - c2;

Serial.print("c2 = ");
Serial.println(c2, DEC);  // Drucken
}

impuls();

if (cap > c1) {
  digitalWrite(7, HIGH);   //ACHTUNG?
  cap = cap - c1;

Serial.print("c1 = ");
Serial.println(c1, DEC);  // Drucken

}
rest = cap;
Serial.print("rest = ");
Serial.println(rest, DEC);


delay(100);   //Einschwingzeit für den Schwingkreis
exit(0);//Test
}
```

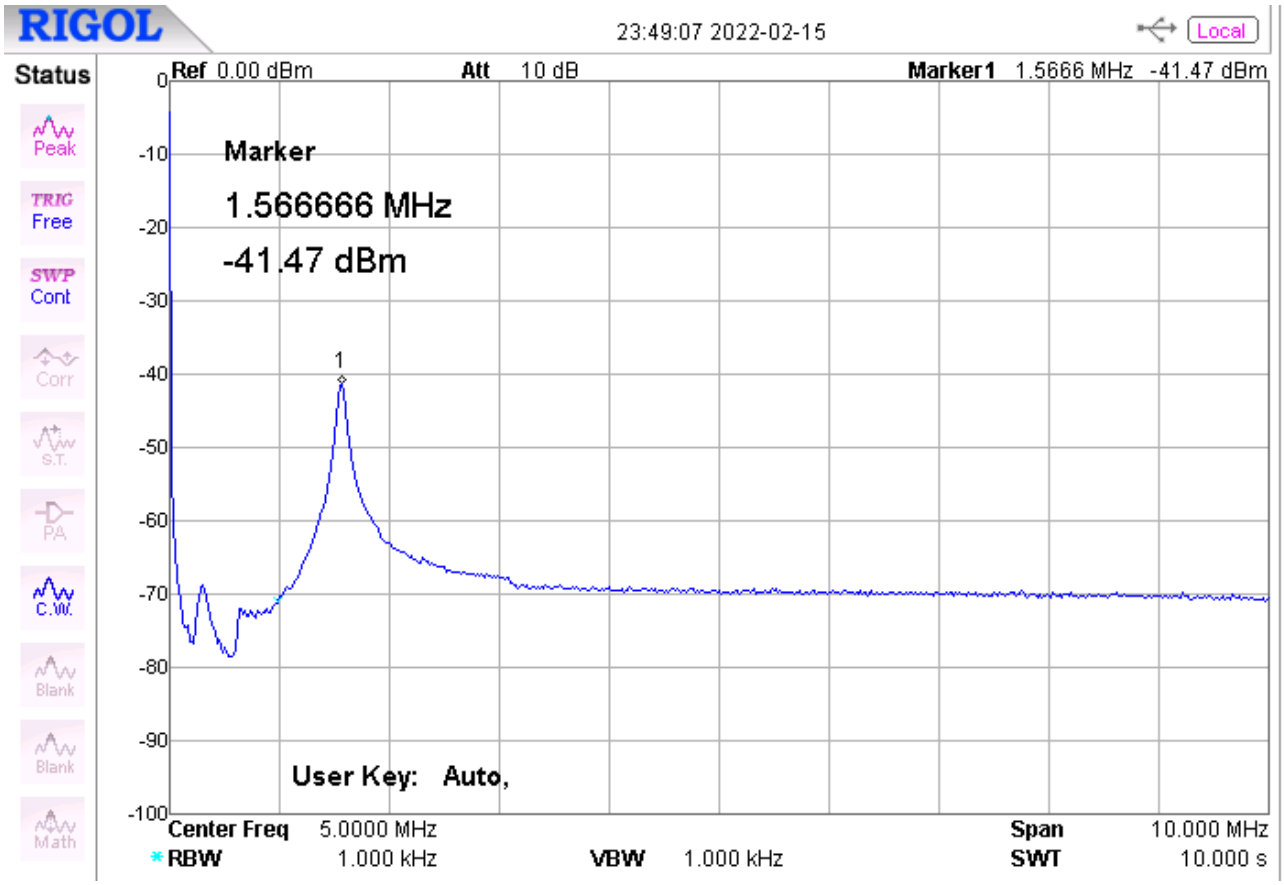
```
void impuls(){

//Ausgabe eines Taktimpulses mit Übernahme der aktuellen Date an D4 = PIN 7

digitalWrite(8, HIGH);   //CLK = übernimmt Date von PIN 7
delay(1);
digitalWrite(8, LOW);

}
```

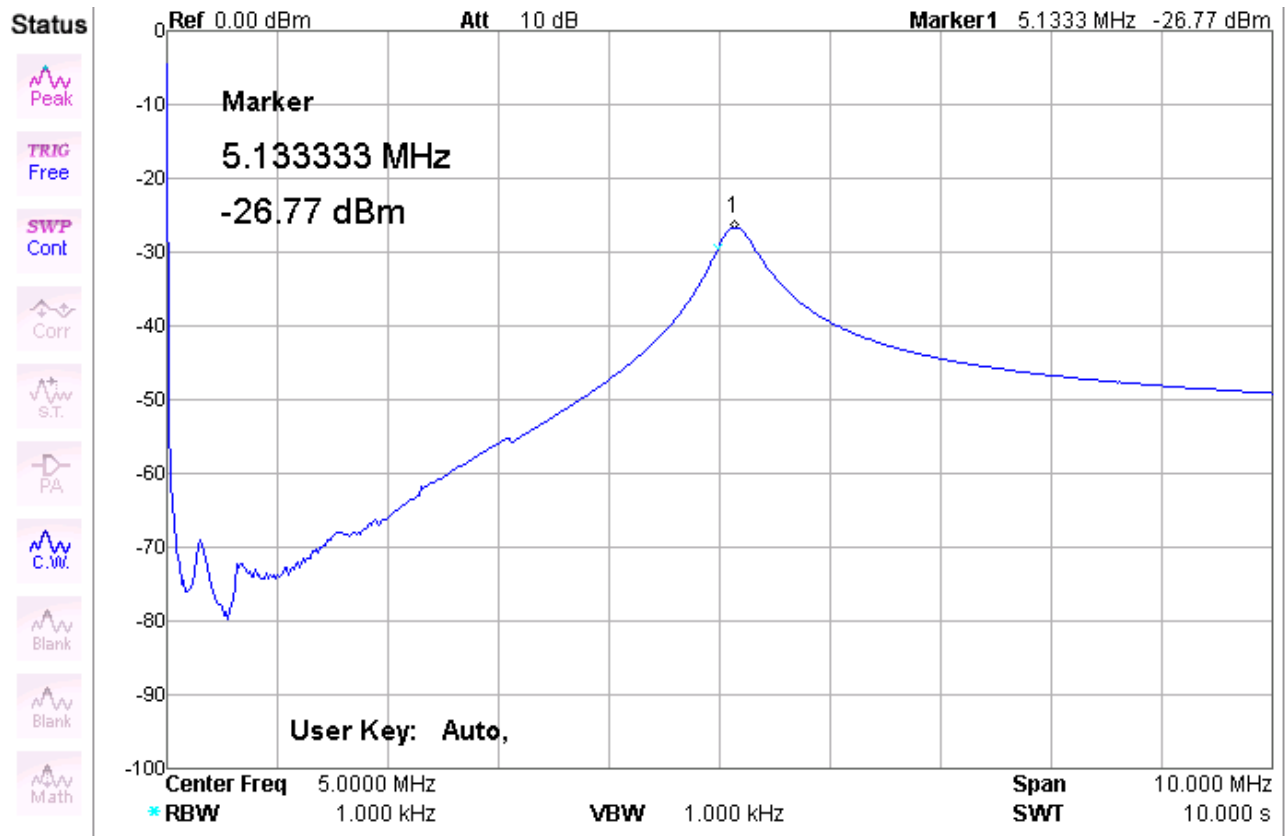
Die drei Ports(D4,D5,D6) müssen am Nano natürlich dafür unbelegt sein.

Kapazität
Sie wird durch Berechnung aus der Empfangsfrequenz ermittelt und in die globale Variable „capa“ geschrieben. In der „Loop“ wird dann laufend überprüft, ob sie durch den Bediener über den Stepper verändert wird. Nur in diesem Fall erfolgt eine neue Speicherung im Schieberegister.

Die Induktivitäten der Schwingkreise sind bekannt. Hier 30 uH. Nach der Formel F = 1/(2Pi x Wurzel(LC) wird C bestimmt, da ja die Frequenz F bekannt ist.

Erster Versuchsaufbau
Zunächst wird mal ein Schwingkreis aufgebaut und mit dem SA(Spektrum-Analysator) untersucht. Die Spule(Ringkern) hat ca. 35 uH. Es wurden parallel zwei Kondensatoren(ca. 300P + 3n) über einen NPN(BC548C) geschaltet. Der Transistor hatte einen Widerstand von der basis nach Masse von 1k. Über einen zweiten 1k wurden dann +4,8V angelegt. Nachstehend die Resonanzkurven. Zunächst mit leitendem Transistor, danach mit dem gesperrten.


RIGOL
23:49:07 2022-02-15
Local
Status
Ref 0.00 dBm
Att 10 dB
Marker1 1.5666 MHz -41.47 dBm
Marker
1.566666 MHz
-41.47 dBm
User Key: Auto,
Center Freq 5.0000 MHz
Span 10.000 MHz
RBW 1.000 kHz
VBW 1.000 kHz
SWT 10.000 s


Status
Ref 0.00 dBm
Att 10 dB
Marker1 5.1333 MHz -26.77 dBm
Marker
5.133333 MHz
-26.77 dBm
User Key: Auto,
Center Freq 5.0000 MHz
Span 10.000 MHz
RBW 1.000 kHz
VBW 1.000 kHz
SWT 10.000 s

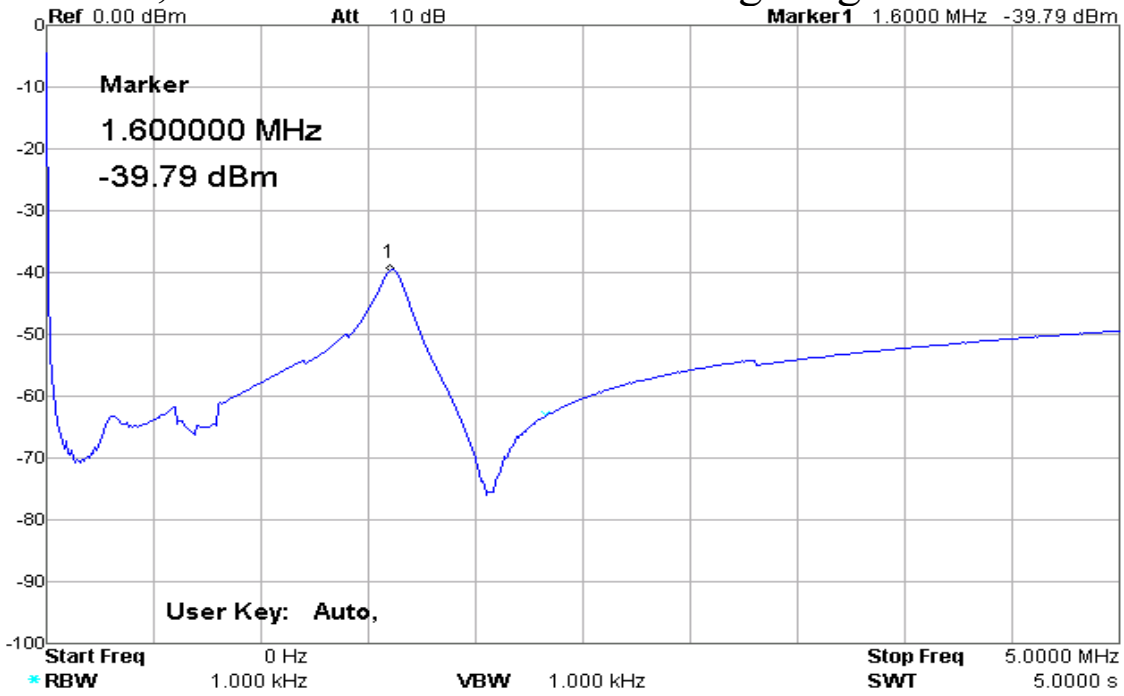
Die -3dB-Bandbreite ist bei 1,6 MHz ca. 40 kHz. Das ist ein guter Wert, zumal der Transistor im Spiel war. Bei gesperrtem Transistor war eine Streukapazität von ca. 25p wirksam. Nach dem Zuschalten eines zweiten BC548C war die Resonanz bei 4,8000MHz. Die Kapazität war jetzt 31,4p. Demnach muss man mit einer Eigenkapazität von 31,4 – 25 = 6.4p/Transistor rechnen. Bei 7 Transistoren also 63,4p Grundkapazität am Schwingkreis. Die dazu wirksame Resonanz wäre bei 3,38 MHz. Also bleibt noch Spielraum für andere Streukapazitäten, wenn man von einer oberen Empfangsfrequenz von 1,6 MHz ausgeht.

Erkenntnis

Der digitale Drehko kann gebaut werden! Es sind sogar die BC548C als Schalter gut geeignet.

Antennenankopplung

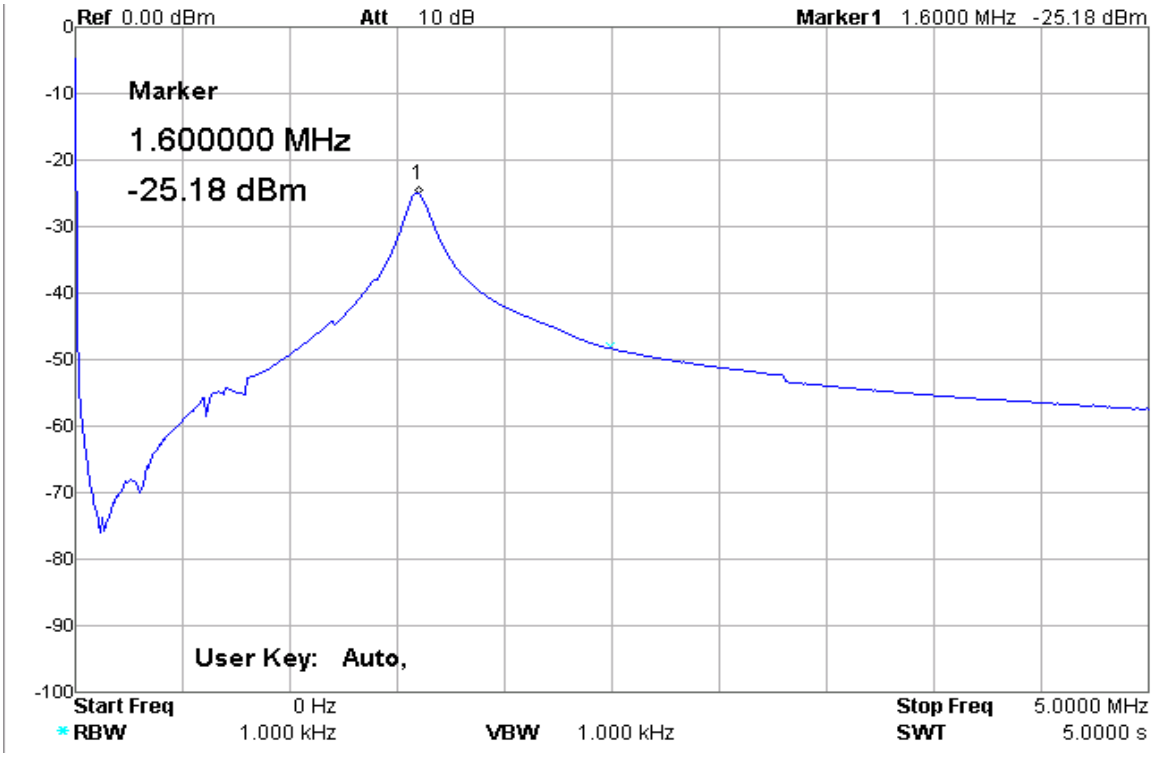
Wieviel Wdg brauche ich für eine Impedanz von 50 Ohm? Dazu wurde der Ringkern, der sekundär 75 Wdg hatte, mit 20 Wdg primär bewickelt. In der Wobbelung zeigte sich, dass man 20 Wdg braucht, um die Verluste bei 500 kHz gering zu halten.



Das Bild zeigt den Verlauf an der 50-Ohm-Wicklung, also primär

als Antennenanschluss. -39,8 dBm ist also die Eingangsleistung.

Hier nun am Schwingkreis:



Es zeigt sich eine Resonanzüberhöhung von 14,6 → 15 dB. Die Bandbreite -3dB war 100 kHz.

20 Wdg sind geeignet, eine Antenne mit Koaxzuführung optimal anzuschließen. Der T80-2 wurde mit Teflonfolie bewickelt, bevor der Draht aufgebracht wurde. Der Lackdraht sollte nicht den Pulverkern direkt berühren. Die Güte des Schwingkreises ist mit Q = 160 gut genug. Im zweiten Vorkreis werden die Verluste durch die Rückkopplung ja aufgehoben und dadurch die Selektivität sehr wirksam gesteigert.

Schwingkreisabstimmung MW-Radio

Wenn man die Einstellung des „Ddrehkos“ digital macht dann gilt folgende Reihe:

MHz	pF
0.5	3377
0.6	2345
0.7	1723
0.8	1390
0.9	1042
1.0	844
1.1	698
1.2	586
1.3	500
1.4	431
1.5	375
1.6	330

3377 / 127 = 26,6 p als 2^0 Basiswert

also 27p als Anfangswert, dann:

27, 54, 108, 216, 432, 864, 1728 p Digitalwerte

Der Drehko lässt sich in Stufen von 27p abstimmen. Man beginnt mit der Streukapazität und addiert dann den DDrehko.

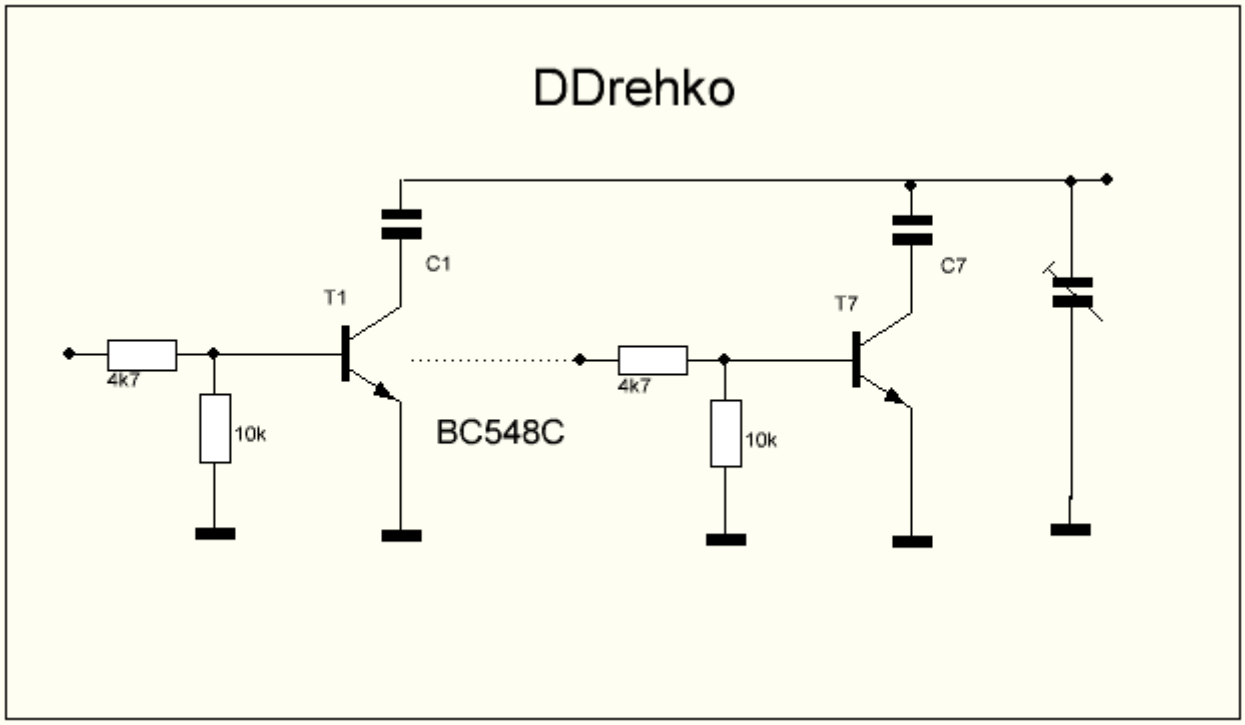
Wie passt das zur linear ansteigenden Frequenz?

Hier würde dann eine Tabelle helfen, die noch feiner auflösen sollte. Aber dafür muss man diese zunächst als Array ermitteln. Geht man von Ca = 100p aus, dann rechnet man:

100 + 230 = 330p → höchste f

230p aus dem DDrehko : 204 + 27 = 2^1 + 2^4 → 1.6 MHz

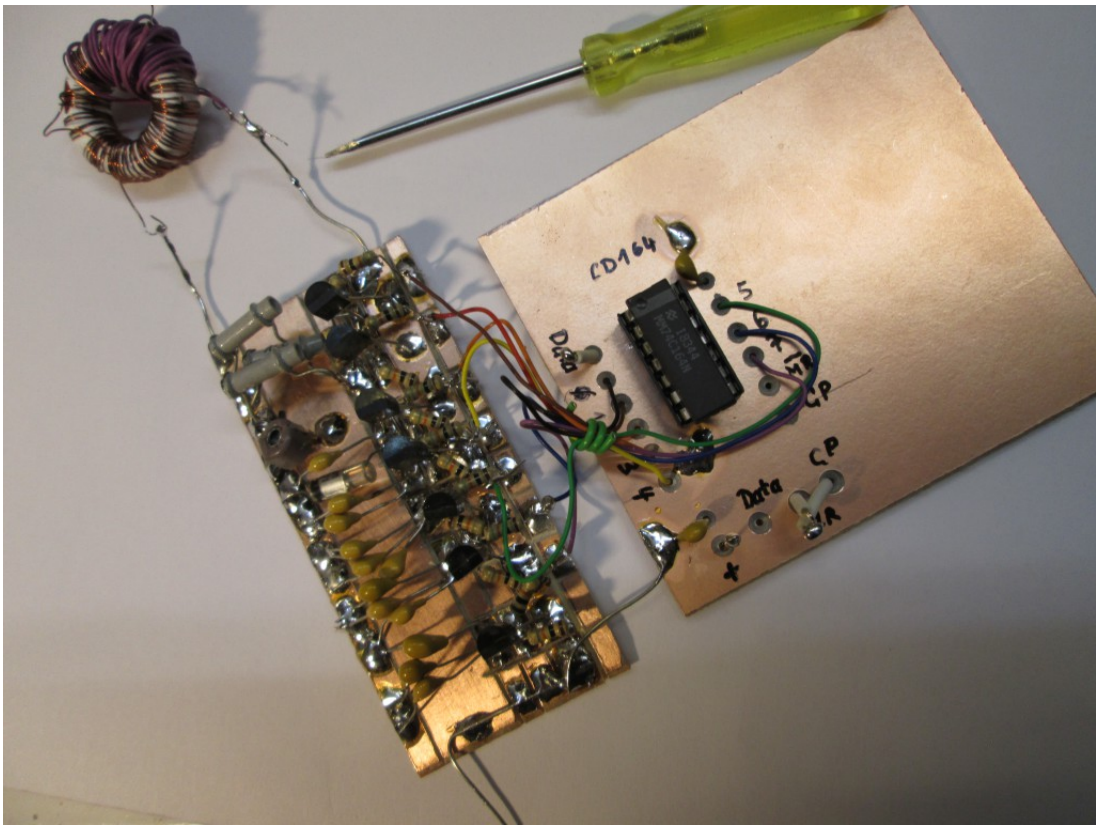
Bei der Bestimmung des einzustellenden C-Wertes ist also zuvor 100p zu subtrahieren. Doch wie korreliert der C-Wert mit der Frequenz? Die Lösung beschreibt die Thomsonsche Formel. Und die habe ich bereits in die Software eingearbeitet.

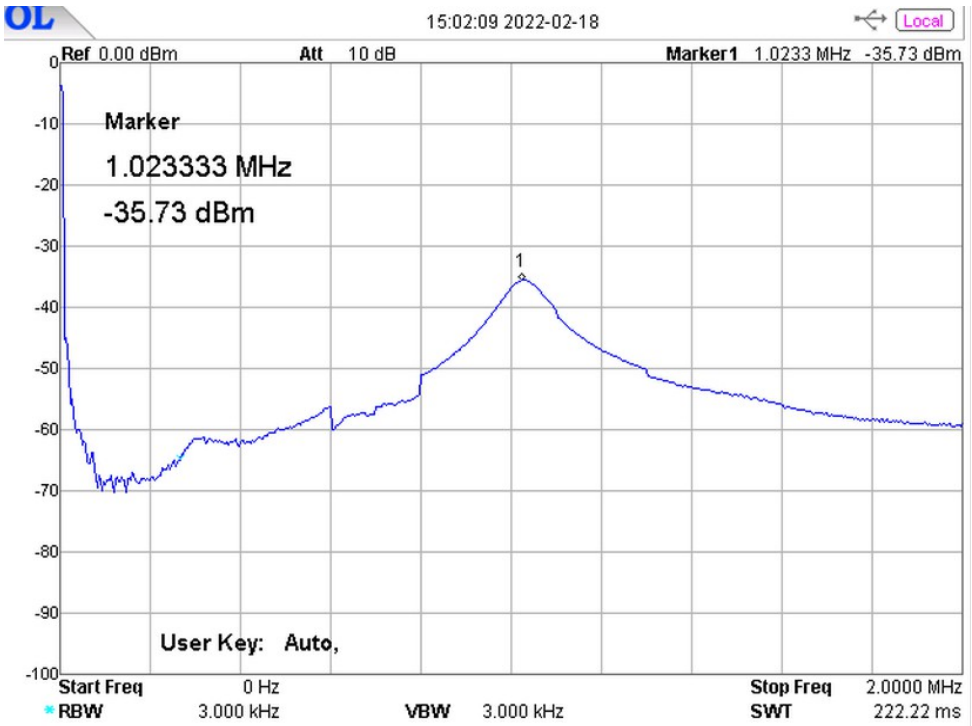


Aus der bekannten Empfangsfrequenz wird der C-Wert errechnet. Davon werden 100p Streukapazität subtrahiert. Erst dann wird mit dem Unterprogramm „kapazitaet();“ der DDrehko eingestellt. Es wird sich aber stets eine geringe Fehlabstimmung zeigen, weil die digitale Reihe nicht ganz korrekt ist. Da aber der Vorkreis eine Bandbreite von >80 kHz hat, wirkt sich das auf die generelle Empfindlichkeit nicht nachteilig aus. Mit der Feinabstimmung über die C-Dioden wird von Hand nachgestimmt, wenn das erforderlich sein sollte. Durch die Rückkopplung im zweiten Vorkreis erzielt man eine hohe Selektivität und besonders wirksame Steigerung der Empfangsleistung.

Das folgende Foto zeigt den ersten Versuchsaufbau des Vorkreises. Der DDrehko ist mit BC548C bestückt. Wegen des „hochohmigen“ Ausgangs der Registerspeicher habe ich einen 10k statt 4k7 vor die basis geschlatet. Ich werde prüfen, ob der CD164

genügend Power rausgibt, um die BC548C durchzuschalten.

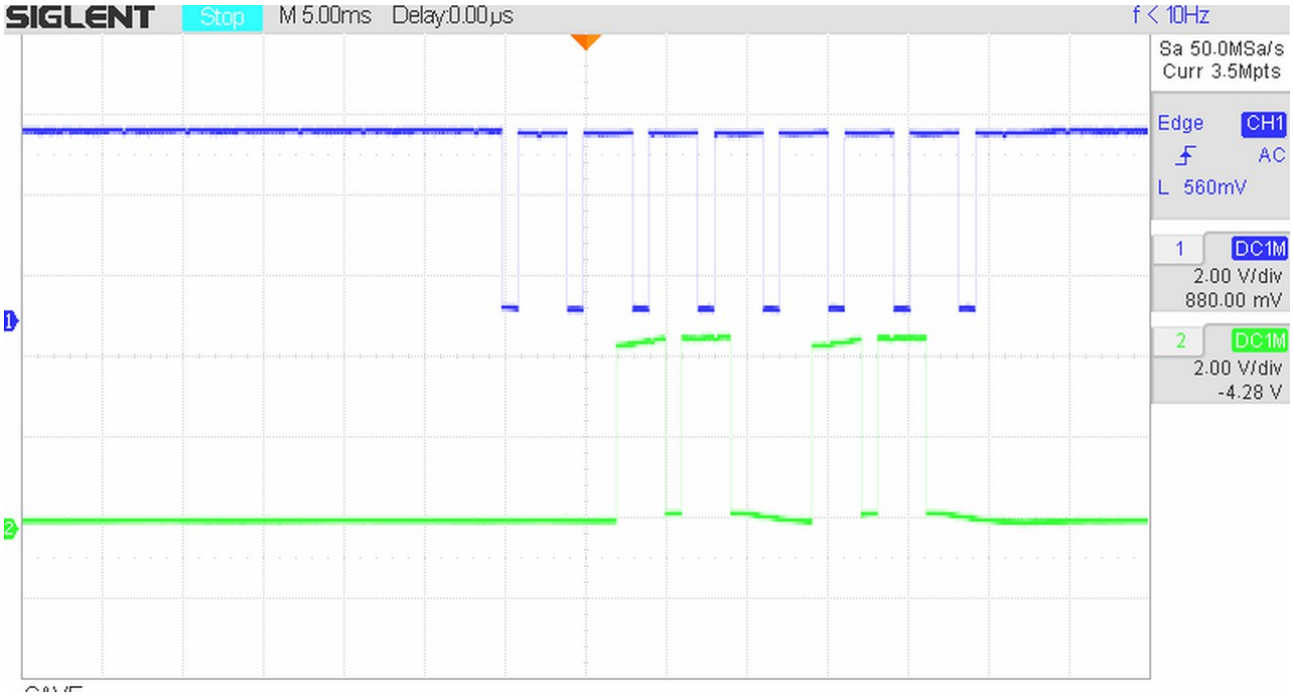




Erste Ergebnisse

Das Bild zeigt die Resonanz für die Einstellung 1MHz. Die Abweichung von 23 kHz ist nicht relevant, da ja eine

Feinabstimmung vorgesehen ist. Die – 3dB-Bandbreite ist ja ca. 40 kHz. Und wenn es sich zeigen sollte, dass eine Auflösung von 8 statt 7 Bits erforderlich sein sollte, dann kann ich ja einen Schalter hinzufügen. Hier das Impulsdiagramm beim Abstimmschritt um 100 kHz auf 1 MHz:



Oben der Schiebetakt, unten die Daten am 74HC164.

Die Induktivität des Schwingkreises und die Streukapazität müssen noch justiert werden. Doch prinzipiell können Abstimmungsfehler passieren, da die Auflösung von 27 pF der binäre Schaltfehler ist.

Jetzt folgt ein Aufbau des zweiten digitalen Kondensators, um ein zweikreisiges Frontend zu bauen. Hier erwarte ich Probleme mit der Schirmung. Die sollte Eigenschwingungen verhindern.

Abstimmungsgenauigkeit
Die Tabelle zeigt die auf dem Display angezeigten und mit dem Spektrumanalysator(SA) beobachteten Resonanzfrequenzen.

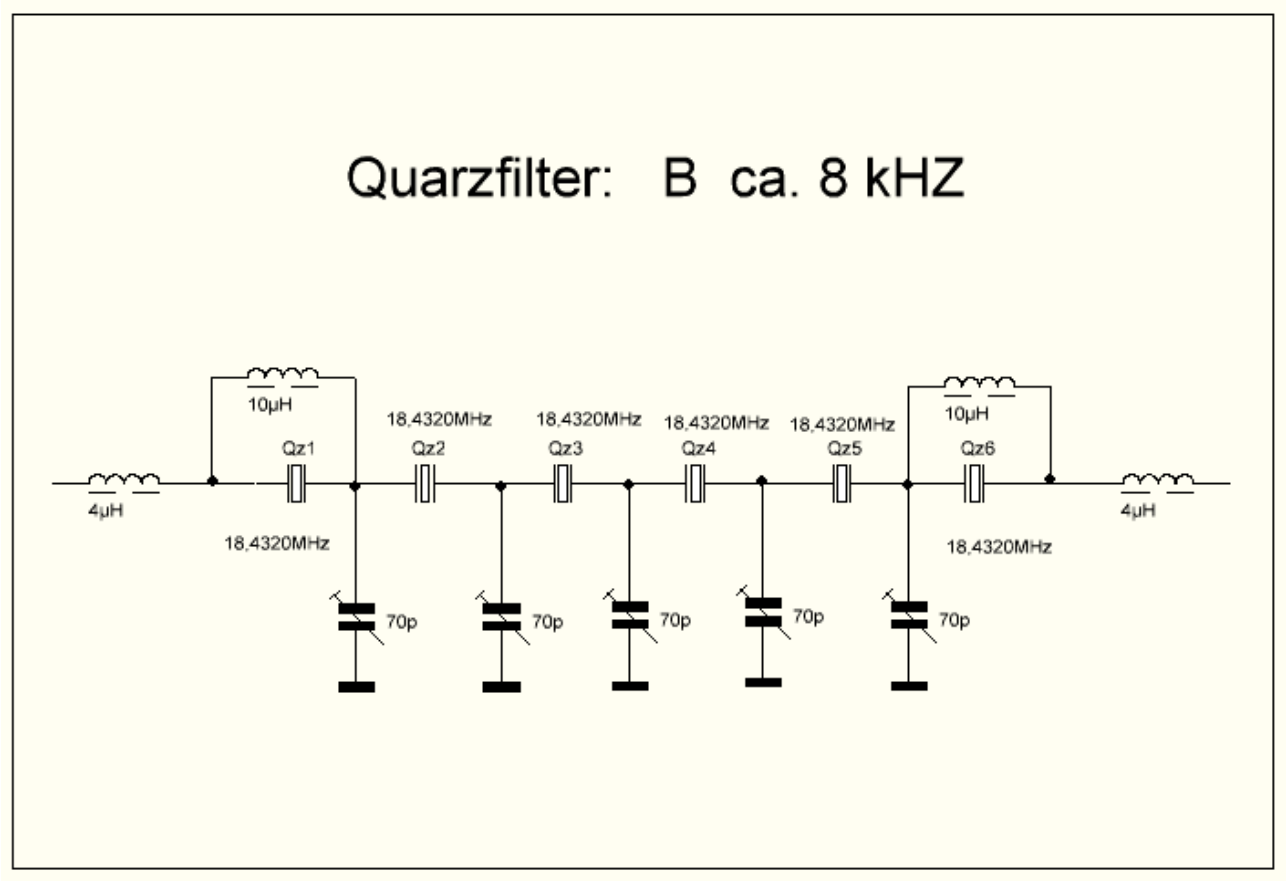
Anzeige kHz	**gemessen kHz**	
500	510	
600	603	
700	710	
800	*800*	
900	*800*	Digitalisierungsfehler?
1000	1000	
1100	1077	
1180	1173	
1199	1220	
1200	*1216*	„ „
1300	*1216*	
1400	1393	
1500	1490	
1600	1600	

Der Abgleich der Streukapazität erfolgte bei 1600 kHz.
Es wird sich zeigen, wie sich die Abweichungen bei den mittleren Frequenzen im praktischen Betrieb auswirken.

Erkenntnis
Die BS170/2N7000 MOSFETs sind wegen ihrer hohen Eigenkapazität nicht geeignet! Der BC548C hat 6 p und der 2N7000 35p. Der74HC164 kann ohne Probleme zweimal 10k bedienen. Daher ist nur ein Schieberegister erforderlich. Daran werden die beiden digitalen Drehkos mit den parallelen Steuerleitungen D1...D7 angeschlossen. Der Anschluss OUT 0 bleibt frei.

Das Quarzfilter
Nach einigen Versuchen mit Quarzen von 4 … 6MHz brachte nur eine höhere Frequenz den Erfolg. Mit 6 Quarzen aus der PC-Elektronik gelang es mir ein genügend breites Filter zu basteln.

Ich nahm 18,432 MHz-Quarze. Schließlich leistet der Si5351 ja diesen Frequenzbereich ohne Probleme. Und mit wenig Aufwand kam dabei ein brauchbares Filter heraus. Es hat ca. 8 kHz Bandbreite und kaum einen Durchlassverlust:



Der Entwurf der Hardware

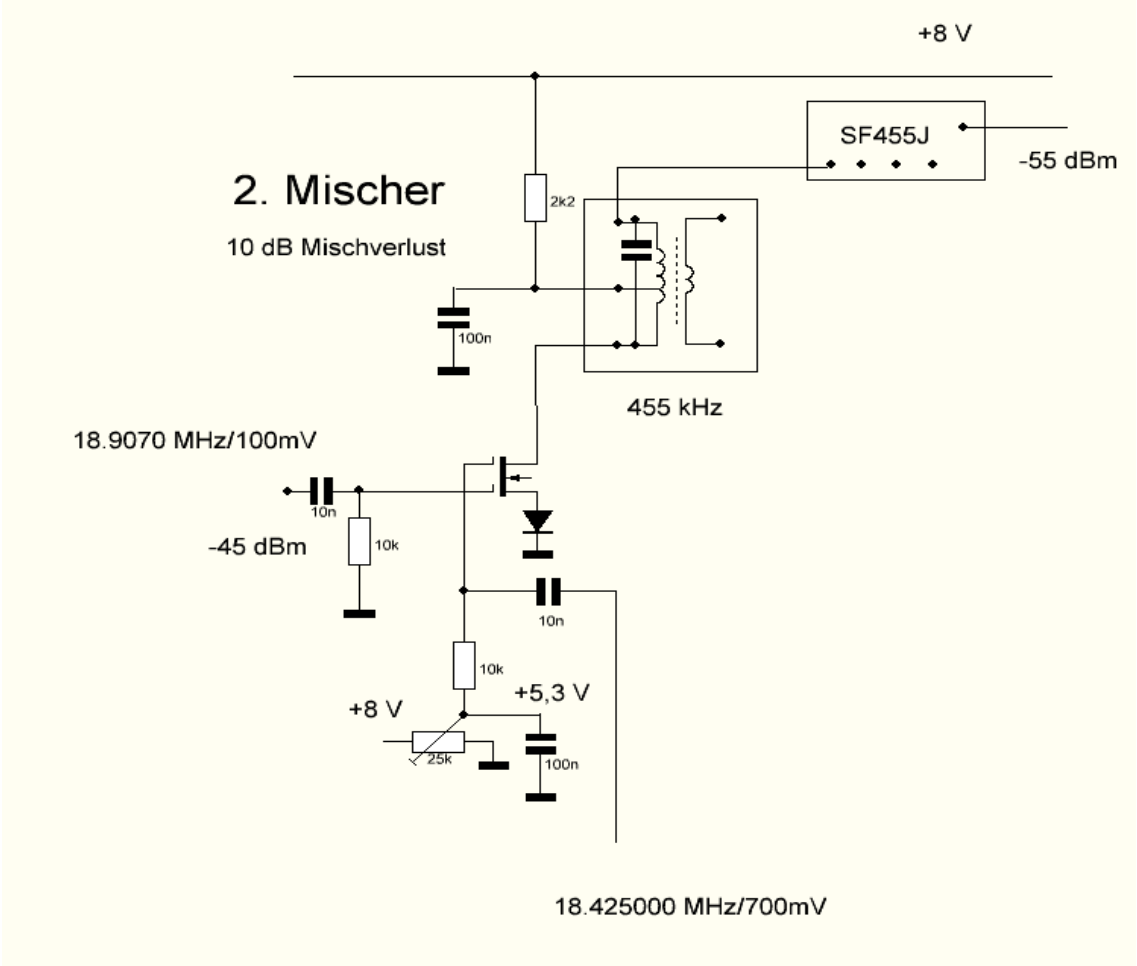
Was soll alles möglich sein? Die Schlüsselfrage für die Gestaltung des Empfängers und seiner Frontplatte. Also zweimal der digitale Kondensator. Zwei Vorkreise, die gut gegeneinander geschirmt werden müssen. Zweimal eine Feinabstimmung. Die sollte von einem gemeinsamen Poti geregelt werden. Dann noch für die Bequemlichkeit bei der Sendersuche drei Tasten für die Steps 100kHz, 5 kHZ und 21 kHz. Die anderen kann man über den Druckschalter des Impulsgebers erreichen. Ferner ist ein Rückkopplungsregler erforderlich. Und ein Pegelregler für die Antennenenergie sollte auch sein. Bleibt noch der NF-Pegelregler und die Kopfhörerbuchse. Also die Frontplatte wird ähnlich wie beim letzten RX gestaltet sein. Vor dem Aufbau der einzelnen

Stufen des RX sollten deshalb die notwendigen Bohrungen vorgenommen werden. Ein gewisser Plan muss eingehalten werden. Also los geht's.

Erste Messungen
Nach dem Quarzfilter ist der Pegel + 10dB höher als an der Antenne. Die RK kann ich noch nicht zum Schwingen bringen, sie regelt aber. Eine Verstärkung wird dadurch noch nicht erreicht. Hier muss ich vermutlich noch mehr Windungen im Sourcekreis aufbringen. Die LO-Frequenz hat noch gut 40kHz Abweichung relativ zur tatsächlichen Empfangsfrequenz(Messsender).

Es gibt noch viel zu tun!

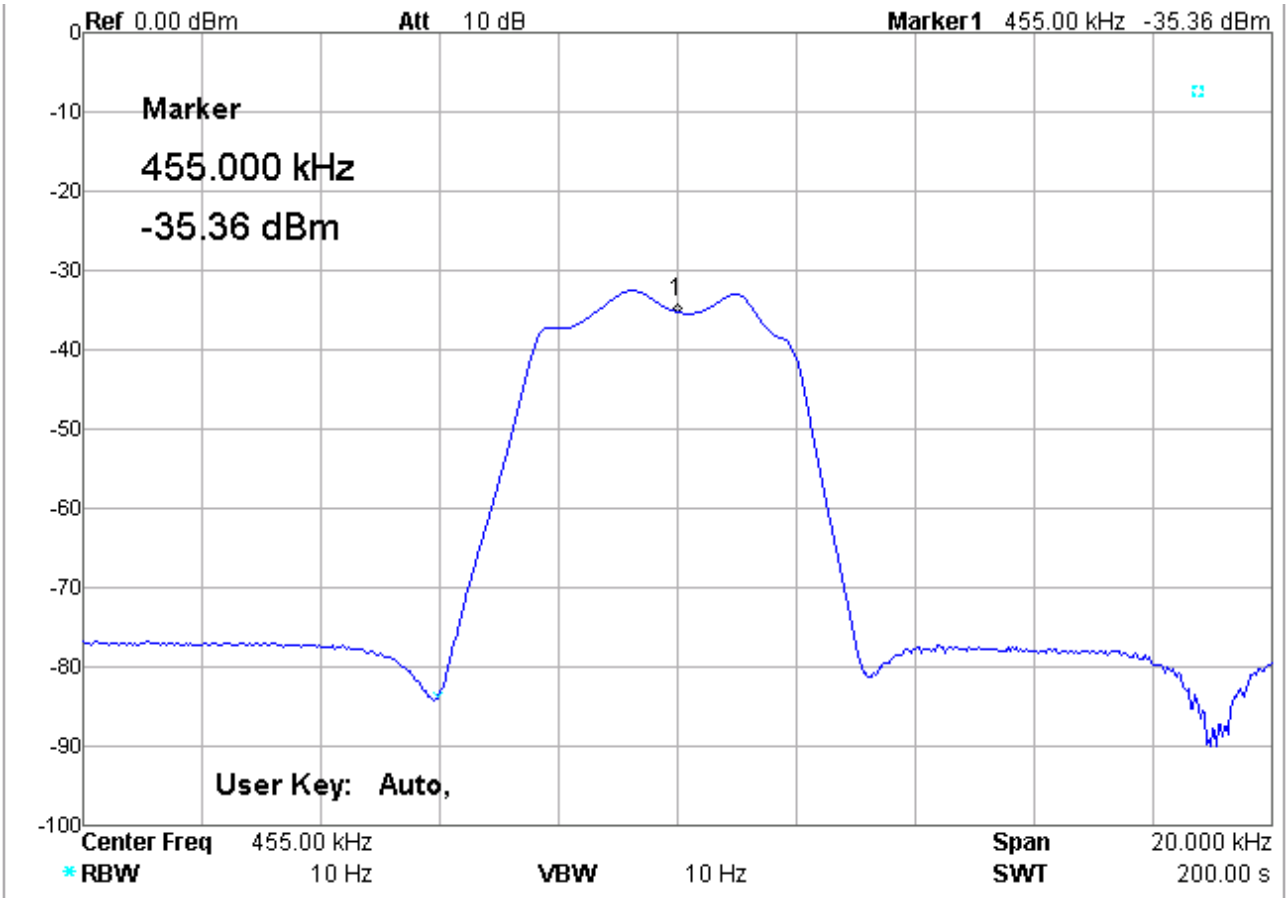
2. Mischer



Man muss mit ca. 10 dB Mischerverlust rechnen. Damit hebt sich die Vorverstärkung wieder auf. Die gesamte Verstärkung muss jetzt vom 2. Mischer bereitgestellt werden.

Zweites ZF-Filter

Es ist ein SF455J von Murata. Eigentlich mit 6 kHz Bandbreite angegeben. Doch hier stellt es sich schmaler dar.



Deshalb werde ich einen Schalter vorsehen, der das Filter überbrückt. Dann hat man die Wahl z wischen zwei Bandbreiten.

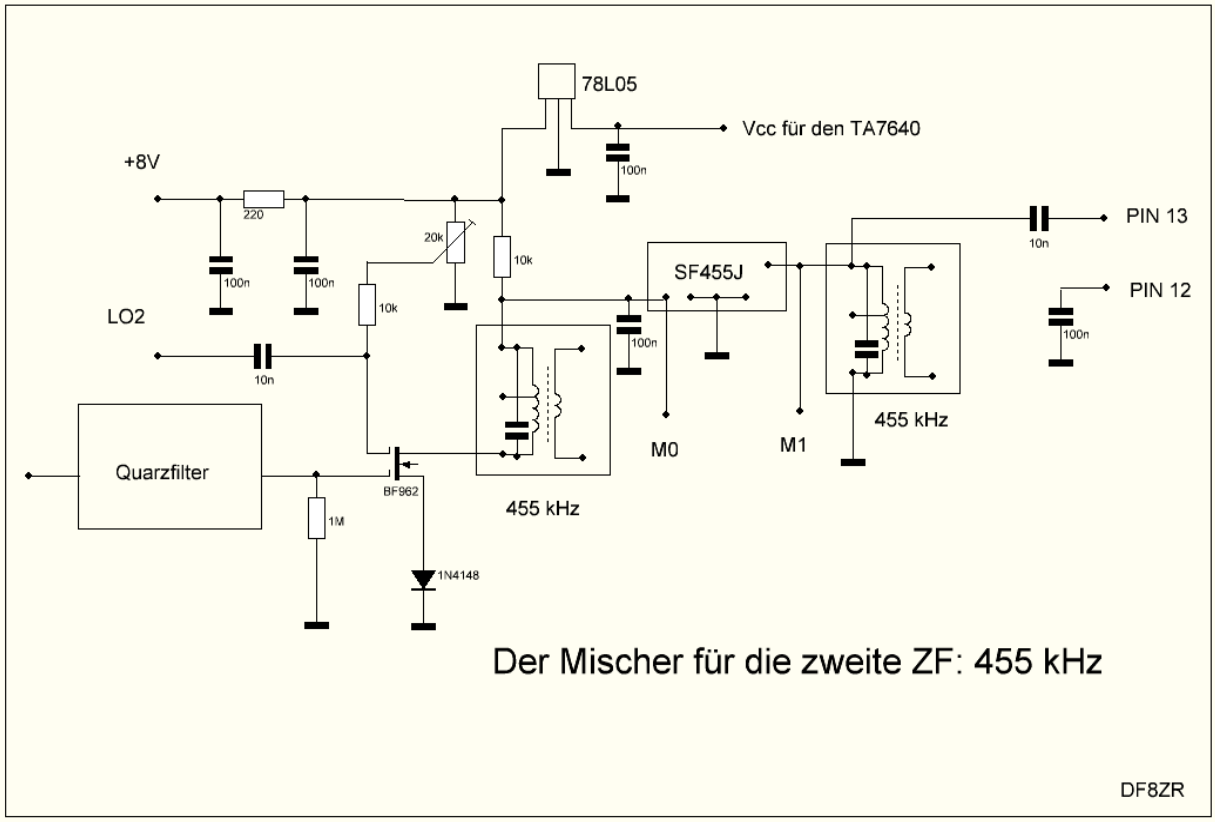
Erster Erfolg

Zwei Wochen nach der Idee kamen die ersten Töne hinten heraus. Toll, doch leider war es um 10.30 LT. Die Tagesdämpfung war zu hoch, um die Sender aus Europa zu empfangen. Die Grenzempfindlichkeit war aber auch noch bei 20mV an 50 Ohm. Zu wenig für den Fernempfang, aber ein Rauschen war bei

Anschluss der Hochantenne deutlich laut zu hören. Das ließ hoffen, hi.

Der Schaltkreis TA7640

Ich hatte davon mal einige ICs vor längerer Zeit gekauft. Nun kam der erste Einsatz. Und natürlich funktionierte er nicht gleich. Zunächst machte das Keramikfilter SF455J Probleme. Es muss hochohmig gespeist werden. Und ein weiterer zusätzlicher Aufwand am IC, nämlich ein Schwingkreis für die Gleichrichtung an PIN5, ist erforderlich. Der Schaltkreis hat nämlich einen internen Schalter für die Ausgabe der NF. Und der braucht leider etwas ZF-Pegel, der am externen Schwingkreis entsteht. Dabei beobachtete ich, dass das Keramikfilter doch schmaler als erwartet ist. Mein Quarzfilter hat ja ca. 9 kHz Bandbreite, hier aber ist nur knapp 5 kHz in der Wobbelkurve zu sehen. Doch zunächst werde ich im weiteren Verlauf mal die Tonqualität prüfen. Ist der Ton zu dunkel, dann werde ich den Schalter für die Überbrückung des Keramikfilters einbauen. Am kommenden Abend wird sich zeigen, was der Empfänger kann.



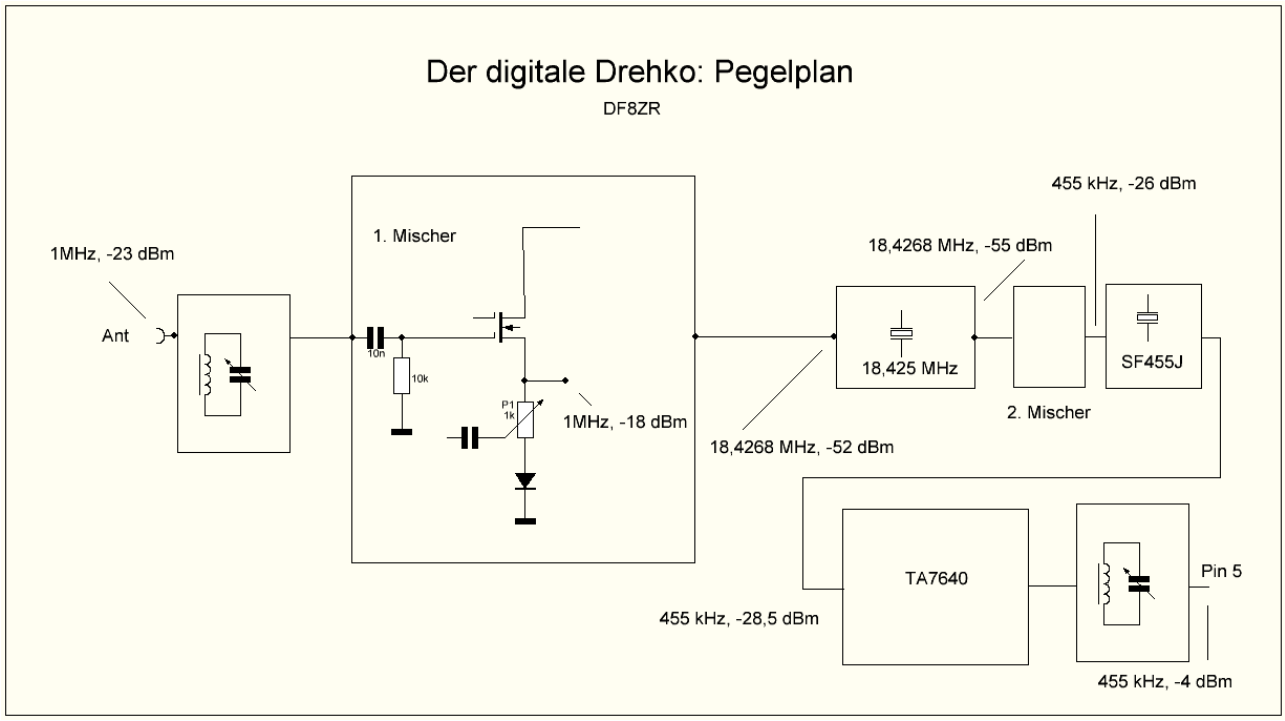
Das zweite Spulenfilter nach dem Keramikfilter kann man weglassen. Der Eingang PIN13 am IC ist hochohmig, wie ich durch ein Experiment feststellte. Der Schwingkreis, der am PIN 5 des ICs ist, muss aber sein. Denn sonst findet keine Gleichrichtung statt, bzw. wird der RX zu unempfindlich und schaltet die NF intern nicht weiter. Wer diesen Umstand vermeiden will, kann ja ein anderes AM-IC einsetzen, z.B. den CD2003.

Na endlich!

Das Radio spielt. Da waren noch einige Problemchen zu lösen. Die Rückkopplung funktioniert nur bei sehr schwachen Sendern. Natürlich, denn die AGC des ICs bügelt alles aus. Und die korrekte Abstimmung der LO-Frequenzen ist nicht möglich. Es entstehen Abweichungen von bis zu 1k. Deshalb habe ich noch den Taster 100Hz zusätzlich eingebaut. Damit kann man feiner abstimmen. Es fehlt noch die geplante Feinabstimmung der Resonanz im zweiten Vorkreis mit Hilfe von C-Dioden. Hierzu nehme ich stets die 1N4007. Hier vielleicht im „Doppelpack“. Aber auch diese Wirkung ist eigentlich marginal. Man könnte darauf verzichten, denn es bringt nicht viel Gewinn. Vielleicht nur bei den schwachen Trägern. Aber es soll ja ein Experimentier-Radio sein!

Hier ein Pegelplan:

Es waren Messungen mit dem SA. Auch hier sind die Frequenzangaben etwas abweichend. Sie hängen ab von der Auflösung am SA(Bandbreite) und der Berechnung in der Software für den Si5351. Die mit einem Frequenzzähler ermittelten Abweichungen der LO-Frequenzen waren bis 1,5 kHz. Daher ist es sinnvoll, eine Feinabstimmung des LO1 mit 100 Hz zu machen. Dann trifft man auch die mittleren Durchlassfrequenzen der ZF-Filter.



Man sieht eine Dämpfung von > 30 dB von der Antenne bis zum ersten ZF-Filter. Diese Mischer mit einem Dualgate-MOSFET haben offenbar hohe Verluste. Das könnte man verbessern.

Dennoch bin ich mit der Empfangsleistung zufrieden. Am Vormittag des 2. Febr 2022 hörte ich gegen 10 Uhr LT immer noch lautstark Lyca Radio auf 1458 kHz. Hier mal ein Link:

https://de.wikipedia.org/wiki/Liste_digitaler_Lang-,_Mittel-_und_Kurzwellenrundfunksender

Allerding betreibe ich eine Hochantenne: 15m lang, 10m hoch.

Fazit
Bis hier soll mal die Beschreibung genügen, um Hilfe für den Nachbau zu sein. Die weiteren Einzelheiten sind nicht wichtig. Ich werde noch an der ersten Mischstufe arbeiten. Insbesondere die Rückkopplung hat noch nicht die erwartete Wirkung.

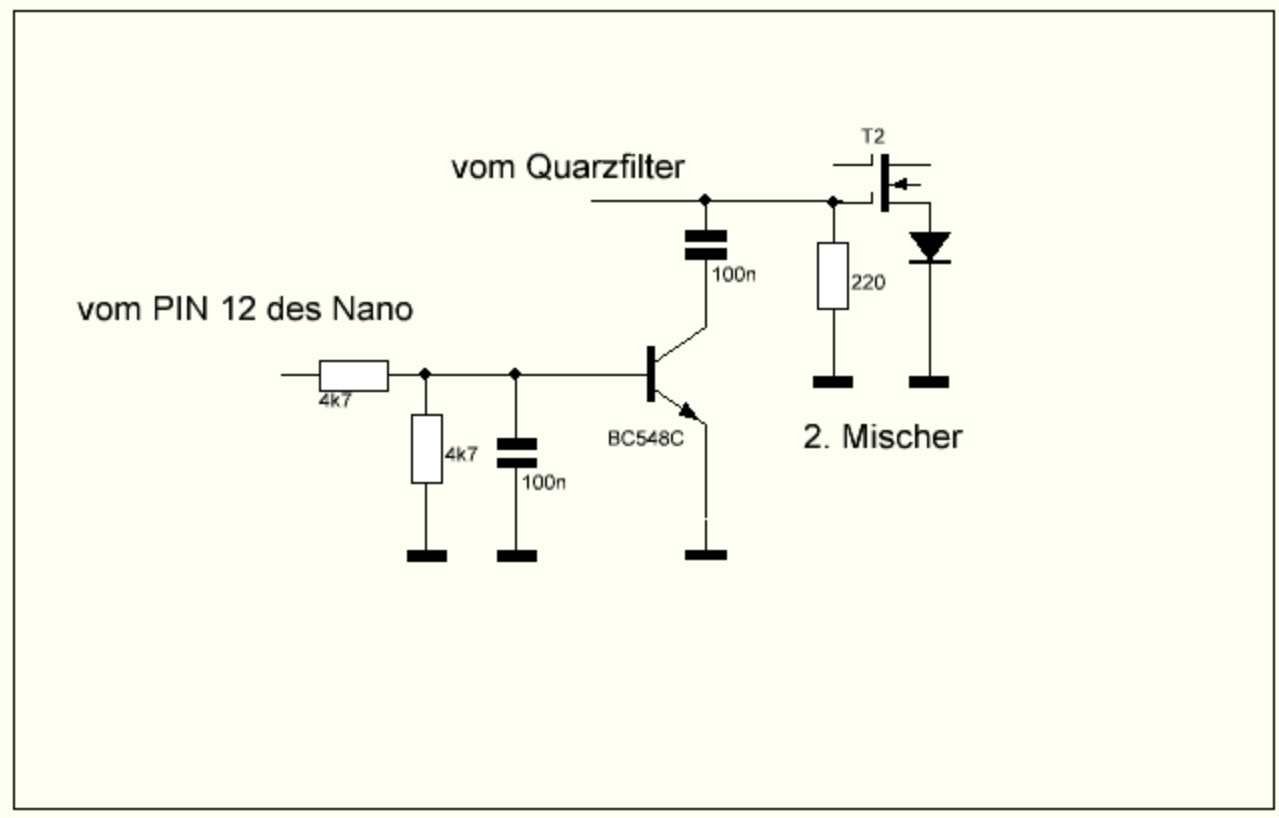
Der digitale Drehko funktioniert. Die Resonanzkurven sind akzeptabel. Man hört natürlich noch die eine oder andere

Störfrequenz aus dem Arduino und dem Schieberegister. Aber das könnte man durch einen optimierten Aufbau verhindern. Mich stört das nicht. Die Schaltgeräusche beim Abstimmen könnte man mit einem für Millisekunden geschalteten Dämpfungsglied im NF-Pfad unterdrücken. Ich werde die Eigenschaften des Empfängers in einem YouTube-Film demonstrieren. Das Eigenrauschen ist noch etwas zu hoch. Die Empfindlichkeit ist aber jetzt schon ausgezeichnet. Den modulierten Messender höre ich bei 2mV immer noch laut und fast rauschfrei. Das sollte für den Empfang der Mittelwellensender aus Europa genügen.

YouTube: https://youtu.be/UbYkZrbBR7o

DF8ZR; im Febr. 2022

Nachtrag

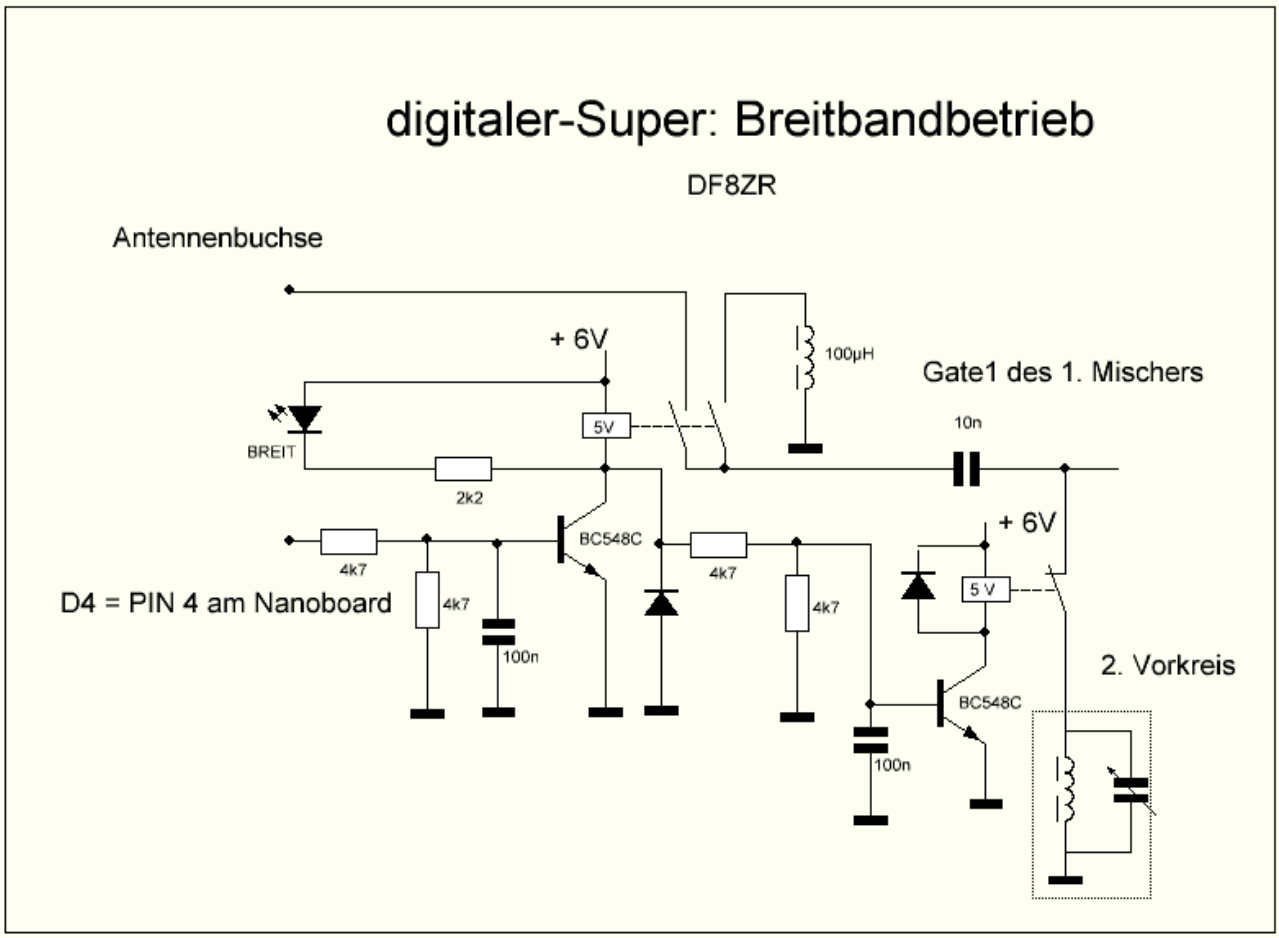


Habe die Schaltgeräusche wirksam unterdrückt. Dazu wurde die

Software geändert. Immer dann, wenn eine Änderung der Frequenz ausgelöst wird, gibt der Nano einen Schaltimpuls am PIN 12 heraus. Damit wird nach dem Quarzfilter für ca. 60ms der HF-Pegel nach Masse geschaltet. Man hört kein Geräusch mehr. Der Empfänger ist ruhig, doch plötzlich fällt ein Sender ein, wenn man nicht zu schnell an der Abstimmung dreht. Mit Schritten in kHz aber „überdreht“ man nichts. Bin selbst überrascht, wie gut das funktioniert und erwäge nochmal einen Film zu machen, der die Nachbesserungen wiedergibt.

Breitbandbetrieb

Es hat sich gezeigt, dass der Empfänger bis über 10 MHz noch Rundfunk empfängt. Dabei sind die Vorkreise natürlich nicht optimal abgestimmt. Aber die hohe Empfindlichkeit macht es möglich, Kurzwellensender auch mit Behelfsantennen zu hören. Daher habe ich ein Umschalt-Relais hinzugefügt. Wenn die Abstimmung über 1,6 MHz dreht, schaltet es sich ein. Es werden die beiden Vorkreise überbrückt. Einen BfO für SSB-Betrieb wollte ich nicht ergänzen. Allerdings hört man mit längeren Drahtantennen auch die Funkamateure auf dem 80m-Band.



BFO-Test: Hier werden die Tasten 100k und 5k für die erhöht die Taste 100k in 100 Hz-Schritten und die Taste 5k erniedrigt sie. Sie wird angezeigt. Den Wert notieren und nach der Rückstellung in den Normalbetrieb bei IF2 eintragen.

Hier der digitale Drehko mit Schaltunterdrückung: MUTE
****
**********/

```
//Libraries
#include <Wire.h>               //IDE Standard
#include <Rotary.h>             //Ben Buxton
https://github.com/brianlow/Rotary
#include <si5351.h>             //Etherkit
https://github.com/etherkit/Si5351Arduino
#include <Adafruit_GFX.h>       //Adafruit GFX
https://github.com/adafruit/Adafruit-GFX-Library
```

```
#include <Adafruit_SSD1306.h>     //Adafruit SSD1306 https://github.com/adafruit/Adafruit_SSD1306

//User preferences
//--------------------------------------------------------------------------------------------------
#define IF 18423.413                //Enter your IF frequency, ex: 455 = 455kHz, 10700 = 10.7MHz, 0 = to direct convert receiver or RF generator, + will add and - will subtract IF offfset.
                                    // hier ist der korrigierte Wert ; die Mittenfrequenz des Quarzfilters ist 18.425133 MHz, der LO erzeut aber etwa mehr:+0,00172 MHz

#define FREQ_INIT  500000 //3600000 für BFO-Test   //Enter your initial frequency at startup, ex: 7000000 = 7MHz, 10000000 = 10MHz, 840000 = 840kHz.
#define XT_CAL_F   33000     //Si5351 calibration factor, adjust to get exatcly 10MHz. Increasing this value will decreases the frequency and vice versa.
#define tunestep   A0        //Change the pin used by encoder push button if you want.

#define IF 18425       // 2. ZF in kHz angeben
#define IF2 17970         // 2. ZF in kHz angeben

#define FREQ_INIT2 17687.5 //??????


//#define IF3 4920. 7

//#define IF3



Rotary r = Rotary(2, 3);
```

```
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);
Si5351 si5351;


unsigned long freq = FREQ_INIT;
unsigned long freqold, fstep;

long interfreq = IF;

unsigned long freqdiff = 1.05;  //1.05;//tatsächliche LO-Frequenz
war 1.05 kHz zu hoch         CALIBRIEREN
//wird vor dem tunebefehl subtrahiert

unsigned long freq2 = FREQ_INIT2;  //für 2. LO; hier:

long interfreq2 = IF2;            //für zwei Zwischenfrequenz

//long freqrm = IF;                     // für remix an Clock 2

//long freqbfo = 49135;  //
/**************************BFO-Test = aktuelle BFO-
Frequenz


long cal = XT_CAL_F;
unsigned long long pll_freq = 90000000000ULL;
byte encoder = 1;
byte stp;
unsigned int period = 100;   //millis display active
unsigned long time_now = 0;  //millis display active




//Tastenklicks-------------------------------------------------------
```

```
---

#define HUNDERT_PIN 11          //D11
#define KILO_PIN 8              //D8
#define FUENFK_PIN 9            //D9
#define HUNDERTK_PIN 10         //D10

int tastenklick;
int hundert;//für frequency
int eink;//für frequency
int st5k;//für frequency
int hundertk;//für frequency
//int zehn;//für frequency

#define MUTEN_PIN 12        //D12 für mute die Schaltklicks

int muten = 0;
#define BREIT_PIN 4         //D4 Breitbandbetrieb > 1,6 MHz

//-------------------------------------------------------------------------
--
//der Teil für den digitalen Drehko:

#define CLK_PIN 5           //D5  hier: CLK für Cd164
#define DATA_PIN 6          //D6  hier: Daten für CD164
#define CLEAR_PIN 7         //D7  hier: Clear für Cd164

float L = 30;               //in uH angeben!

float cap;                  //für die geschalteten Kapazitaeten

int capa;

//int capalt;       //Zwischenspeichern
```

```
int cstreu = 100;  // Wert der Streukapazität; muss nach der Ermittlung von cap subtrahiert werden;
                   //Cstreu muss durch Messung ermittelt werden

int rest;          //nur für den Test

unsigned long freqakt; //freq als unsigned long muss vom Typ erhalten bleiben

//die Werte für das MW-Radio:

int c7 = 1728;
int c6 = 864;
int c5 = 432;
int c4 = 216;
int c3 = 108;
int c2 = 54;
int c1 = 27;


//------------------------------------------------------------------------
---




ISR(PCINT2_vect) {
  char result = r.process();
  if (result == DIR_CW) set_frequency(1);
  else if (result == DIR_CCW) set_frequency(-1);
}

void set_frequency(short dir) {
  if (encoder == 1) {                    //Up/Down frequency
    if (dir == 1) freq = freq + fstep;       //dir = direction
```

```
    if (freq >= 120000000) freq = 120000000;
    if (dir == -1) freq = freq - fstep;
    if (fstep == 1000000 && freq <= 1000000) freq = 1000000;
    else if (freq < 10000) freq = 10000;

  }
}

//**********************************************************
**********SETUP**********************

void startup_text() {
  display.setTextSize(1);
  display.setCursor(4, 5);
  display.print("Si5351");
  display.setCursor(4, 20);
  display.print(" MW-Superhet ");
  display.setCursor(4, 35);
  display.print("Vers. DF8ZR");
  display.setCursor(4, 50);
  display.print(">> 0,5...1,6 Mc <<");
  display.display();
  delay(3000);//3000
  display.clearDisplay();
}


void setup() {                                //SETUP

//pinMode(ZEHN_PIN, INPUT_PULLUP);
pinMode(HUNDERT_PIN, INPUT_PULLUP);
pinMode(KILO_PIN, INPUT_PULLUP);
pinMode(FUENFK_PIN, INPUT_PULLUP);
pinMode(HUNDERTK_PIN, INPUT_PULLUP);
```

```
pinMode(MUTEN_PIN, OUTPUT );
pinMode(4, OUTPUT);     //Breitband
digitalWrite(4, LOW);


//zehn=0;
hundert=0;
eink=0;
st5k=0;
hundertk=0;
tastenklick=0;//keine Taste betätigt

muten = 0;




  pinMode(5, OUTPUT);     //CLK ++++++++++++++++++++++++++
+++++Digital-Superhet
  pinMode(6, OUTPUT);     //DATA
  pinMode(7, OUTPUT);     //CLEAR


digitalWrite(7, LOW);  //CLEAR
delay(1);
digitalWrite(7, HIGH);


//zunächst alle HIGH setzen
//aber einmal CLEAR senden, damit das register gelöscht ist

digitalWrite(5, HIGH);   //CLK
digitalWrite(6, HIGH);   //DATA
```

```
  Wire.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.display();

  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(tunestep, INPUT_PULLUP);



  startup_text();

  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, cal);
  si5351.output_enable(SI5351_CLK0, 1);                 //1 - Enable /
0 - Disable CLK fü LO
  //si5351.output_enable(SI5351_CLK2, 1);                 // für
remix~~~~~~~~ 3. LO
  si5351.output_enable(SI5351_CLK1, 1);                 //aktiviert für
2. Mischer

  si5351.drive_strength(SI5351_CLK0,
SI5351_DRIVE_2MA);  //Output current 2MA, 4MA, 6MA or
8MA
  si5351.drive_strength(SI5351_CLK1,
SI5351_DRIVE_2MA);  //Output current 2MA, 4MA, 6MA or
8MA
 // si5351.drive_strength(SI5351_CLK2,
SI5351_DRIVE_2MA);  // 2 mA for HB
mixers//~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
 //interfreq2 Ist IF2
 //freq2 ist IF2 455000

  si5351.set_freq_manual((freq2 + (interfreq2 * 100ULL)) *
100ULL, pll_freq, SI5351_CLK1);//LO2
  si5351.set_freq_manual((freq + (interfreq * 100ULL)) *
100ULL, pll_freq, SI5351_CLK1);//LO

  PCICR |= (1 << PCIE2);
  PCMSK2 |= (1 << PCINT18) | (1 << PCINT19);
  sei();

  stp = 3;
  setstep();
  layout();
  displayfreq();
}



//#####################################################
########## LOOP ######################
void loop() {

if (tastenklick == 0){

  if (freqold != freq) {
    time_now = millis();
   // stumm();               //schaltet für 60ms die NF stumm
   digitalWrite(12, HIGH);
   tunegen();
    freqold = freq;

digitalWrite(7, LOW);  //Register = CLEAR
```

```
delay(1);
digitalWrite(7, HIGH);

    kapazitaet();  //bei Frequenzänderung neue kapazität bestimmen
und setzen
    schalten();  //??????????????????????????????

    }
}


  if (digitalRead(tunestep) == LOW) {
    time_now = (millis() + 300);
    setstep();
    delay(300);


}

if (digitalRead(HUNDERT_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;
 hundert = 1;
 tasten();

}



if (digitalRead(KILO_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;
 eink = 1;
 tasten();
```

```
}



if (digitalRead(FUENFK_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;  //Normalbetrieb
//tastenklick=0;//**********************BFO-Test
 st5k = 1;
 tasten();

}

if (digitalRead(HUNDERTK_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;//Normalbetrieb

//tastenklick=0;//**************************BFO-Test
 hundertk = 1;
 tasten();

}







  if ((time_now + period) > millis()) {
    displayfreq();
    layout();
  }
  delay(100);//für tastensetzen, sonst Dauersetzen

muten = 0; //stummschalten Ende
```

```
//nach gemessenen 60 ms
digitalWrite(12, LOW);


}

//#########################################################################################


void tunegen() {

//  freqrm = freq - IF3;//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  //freq = freq-freqdiff;//Abweichung von der tatsächlichen Frequenz, hier gemessen = +1,05 kHz

   //VORLAGE: si5351.set_freq_manual((freq + (interfreq * 1000ULL)) * 100ULL, pll_freq, SI5351_CLK0); //LO
if (freq > 1600000) {
  // überbrücke Vorkreise = Breitbandbetrieb
  digitalWrite(7, HIGH);
}
else if (freq < 2000000) {

  digitalWrite(7, LOW);
}

   ///LO1 setzen:
  si5351.set_freq_manual((freq + (interfreq * 1000ULL)) * 100ULL, pll_freq, SI5351_CLK0);           //LO1

  si5351.set_freq_manual( (interfreq2  *  1000ULL) * 100ULL, pll_freq, SI5351_CLK1); //                  LO2 wird im Status()
```

```
gesezt!

  //si5351.set_freq_manual(freqrm * 100ULL, pll_freq,
SI5351_CLK2);// für remix ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
}




void tasten(){

if (hundert == 1) {

stp = 3;
     fstep = 100;
}


if (st5k == 1) {

stp = 5;
     fstep = 5000; // statt 10kHz-steps werden 5kHZ-steps
eingestellt!

    // bfo();//***************
}




if (eink == 1) {

stp = 4;
     fstep = 1000;
}
```

```
if (hundertk == 1) {

stp = 6;
    fstep = 100000;

//bfo();       //**********************************BFO-Test

}

//zehn=0;
hundert = 0;
eink=0;
st5k=0;
hundertk=0;
tastenklick=0;

tunegen();
freqold = freq;

//digitalWrite (ZEHN_PIN,HIGH);
//digitalWrite (HUNDERT_PIN,HIGH);
digitalWrite (KILO_PIN,HIGH);
digitalWrite (FUENFK_PIN,HIGH);
digitalWrite (HUNDERTK_PIN,HIGH);

}



//00000000000000000000000000000000000000000000000000000000000

void displayfreq() {
```

```
  unsigned int m = freq / 1000000;
  unsigned int k = (freq % 1000000) / 1000;
  unsigned int h = (freq % 1000) / 1;

  display.clearDisplay();
  display.setTextSize(2);

  char buffer[15] = "";
  if (m < 1) {
    display.setCursor(41, 1); sprintf(buffer, "%003d.%003d", k, h);
  }
  else if (m < 100) {
    display.setCursor(5, 1); sprintf(buffer, "%2d.%003d.%003d", m,
k, h);
  }
  else if (m >= 100) {
    unsigned int h = (freq % 1000) / 10;
    display.setCursor(5, 1); sprintf(buffer, "%2d.%003d.%02d", m,
k, h);
  }
  display.print(buffer);
}




void setstep() {


  switch (stp) {
    case 1:
      stp = 2;
      fstep = 10;
      break;
    case 2:
```

```
      stp = 3;
      fstep = 100;
      break;
    case 3:
      stp = 4;
      fstep = 1000;
      break;
    case 4:
      stp = 5;
      fstep = 5000;
      break;
    case 5:
      stp = 6;
      fstep = 100000;
      break;
    case 6:
      stp = 1;
      fstep = 1000000;
      break;
  }
}


void layout() {
  display.setTextColor(WHITE);
  display.drawLine(0, 20, 127, 20, WHITE);
  display.drawLine(0, 43, 127, 43, WHITE);
  display.drawLine(105, 24, 105, 39, WHITE);
  display.setTextSize(2);
  display.setCursor(2, 25);
  display.print("ST:");
  if (stp == 2) display.print("10Hz"); if (stp == 3)
display.print("100Hz"); if (stp == 4) display.print("1k");
  if (stp == 5) display.print("5k"); if (stp == 6)
display.print("100k"); if (stp == 1) display.print("1M");
```

```
  display.setCursor(2, 48);


  display.print("0,5-1,6MHz");
  //display.print("k");
  display.setTextSize(1);
  display.setCursor(110, 23);
  if (freq < 1000000) display.print("kHz");
  if (freq >= 1000000) display.print("MHz");
  display.setCursor(110, 33);
  if (interfreq == 0) display.print("VFO");
  if (interfreq != 0) display.print("LO");
  display.display();
}

void layouttest() {

display.clearDisplay();

display.setTextColor(WHITE);

display.setTextSize(2);

display.setCursor(2, 48);

display.print(String(capa));


}



void kapazitaet() {
```

```
//freq = freq / 1000; //ist in kHz angegeben, muss nach MHz umgerechnet werden

freqakt = freq;  //Type unsigned long für freq erhalten!

float f = (float) freqakt;  //nach float convertieren

f = f/1000000; //Frequenz in MHz!!


cap = 25326.7 / (f * f * L);  //25326.7 capa ist eine globale Variable

cap = round(cap);

capa = (int) cap;  // float nach int für die C-Schaltung

capa = capa - cstreu; // die Streukapazität ist Bestandteil der Abstimmung.


//Es muss aber der berechnete Kondensator minus cstreu dazugeschaltet werden. Und das wird jetzt mit schalten(); gemacht.
//Serial.println(capa);

if (cap < 0) cap = 0;

//layouttest();
// delay(1000);
}
```

```
void schalten() {

  //test();

//#define CLK_PIN 5        //D5  hier: CLK für Cd164
//#define DATA_PIN 6       //D6  hier: Daten für CD164
//#define CLEAR_PIN 7      //D7  hier: Clear für Cd164
 //Serial.begin(9600); //für Test



//hier werden die digitalen Ausgänge aktiviert, d.h. auf "1" gesetzt
//zuvor waren alle auf LOW gesetzt

//Serial.println(cap, DEC);  // Drucke im Testbetrieb


//zunächst wird das Register gelöscht:



digitalWrite(7, LOW);  //CLEAR
delay(1);
digitalWrite(7, HIGH);


delay(1);
digitalWrite(6, LOW); //DATA = 0 setzen Grundzustand
delay(1);

 //CLK = übernimmt die Daten von PIN 6

if (capa > c7) {digitalWrite(6, HIGH);
```

```
capa = capa - c7;

//Serial.print("c9 = ");
//Serial.println(c9, DEC);  // Drucken im Test

}
//falls DATA = 0 war, wird eine 0 gespeichert
//falls DATA eine 1 war, wird eine 1 gespeichert


impuls();


if (capa > c6)
{
  digitalWrite(6, HIGH);
capa = capa - c6;

//Serial.print("c6 = ");
//Serial.println(c6, DEC);  // Drucken
}

impuls();



if (cap > c5) {digitalWrite(6, HIGH);
capa = capa - c5;
//Serial.print("c5 = ");
//Serial.println(c5, DEC);  // Drucken
}

impuls();
```

```
if (capa > c4) {
  digitalWrite(6, HIGH);

capa = capa - c4;

//Serial.print("c4 = ");
//Serial.println(c4, DEC);  // Drucken
}

impuls();

if (capa > c3){
  digitalWrite(6, HIGH);

capa = capa - c3;

//Serial.print("c3 = ");
//Serial.println(c3, DEC);  // Drucken
}

impuls();

if (capa > c2){
  digitalWrite(6, HIGH);

capa = capa - c2;

//Serial.print("c2 = ");
//Serial.println(c2, DEC);  // Drucken
}

impuls();

if (cap > c1) {
  digitalWrite(6, HIGH);   //ACHTUNG?
```

```
  capa = capa - c1;

//Serial.print("c1 = ");
//Serial.println(c1, DEC);  // Drucken
}

impuls();


impuls();    // OUT 0 wird nicht benutzt, daher muss eine Zelle weiter geschoben werden

if (capa < 0) capa = 0;

//rest = capa;//mnur für test
//Serial.print("rest = ");
//Serial.println(rest, DEC);

//layouttest();
//delay(1000);

}


void impuls(){

//Schieberegister CD164
//Ausgabe eines Taktimpulses mit Übernahme der aktuellen Date an  PIN 6

//#define CLK_PIN 5        //D5  hier: CLK für Cd164
//#define DATA_PIN 6       //D6  hier: Daten für CD164
//#define CLEAR_PIN 7      //D7  hier: Clear für Cd164
```

```
//digitalWrite(5, HIGH);   //CLK = übernimmt Date von PIN 6

delay(1);
digitalWrite(5, LOW);      //Impuls erzeugen
delay(1);
digitalWrite(5, HIGH);
delay(1);
digitalWrite(6, LOW);      //DATA auf 0 setzen
delay(1);
// layouttest();
//delay(1000);
}


void stumm(){

digitalWrite(12, HIGH);

}

/*
void test(){


digitalWrite(5, LOW);
delay(1000);
digitalWrite(5, HIGH);

delay(2000);

digitalWrite(5, LOW);
delay(1000);
digitalWrite(5, HIGH);
```

```
digitalWrite(5, LOW);
delay(1000);
digitalWrite(5, HIGH);

delay(2000);

digitalWrite(5, LOW);
delay(1000);
digitalWrite(5, HIGH);
  digitalWrite(5, LOW);
}
```