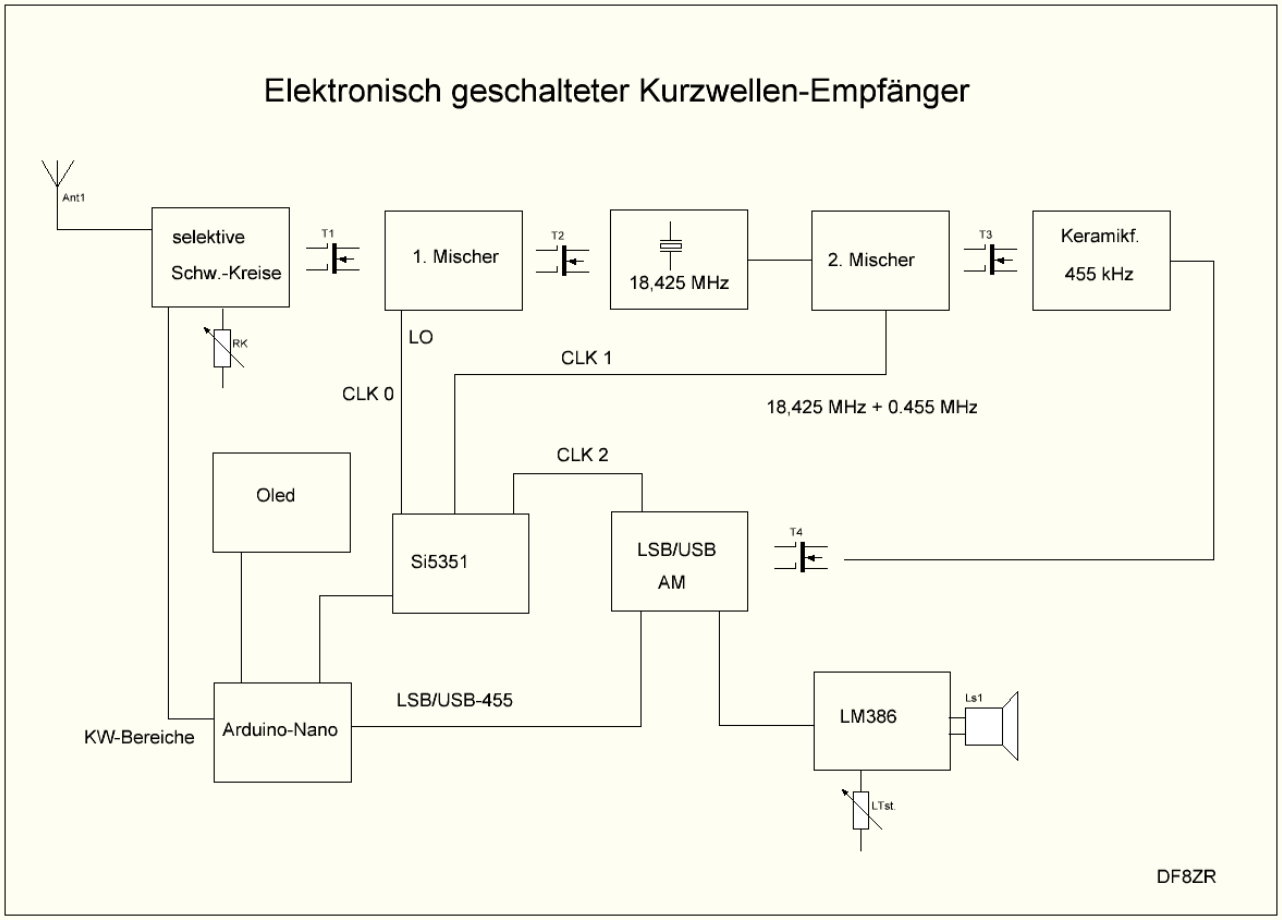
Der RK-Superhet

Nach den guten Erfahrungen mit dem BF962/61 plane ich nun, einen elektronisch geschalteten Kurzwellenempfänger zu basteln. Ich nenne ihn: „RK-Superhet“.

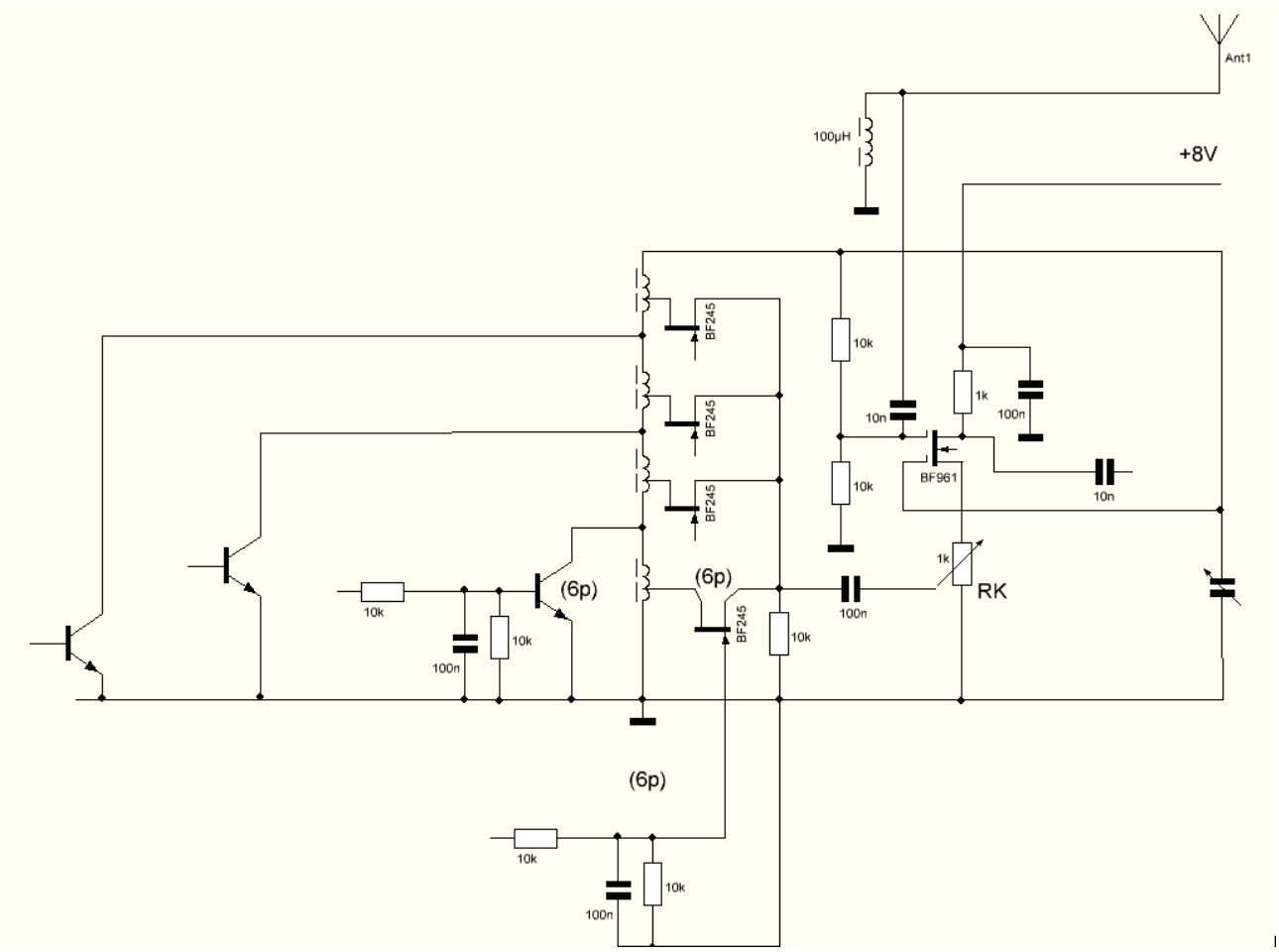


Schwerpunkte der Entwicklung sind hier die elektronische Umschaltung der Empfangsbereiche. Weiterhin die Steigerung der Selektivität durch eine Rückkopplung im Vorkreis.

Es kommt zur Feinabstimmung ein herkömmlicher Drehkondensator zum Einsatz. Mit Transistoren und FETs wird jeweils nur ein Schwingkreis an den Eingangstransistor BF961 geschaltet. Durch die Rückführung von Energie aus dem Sourcekreis wird der Schwingkreis entdämpft und selektiv. Zur Entkopplung des ersten Oszillators LO wird die HF von der Antenne breitbandig auf das Gate2 des BF961 geschaltet. Die

Schaltung arbeitet stabil, wenn man im Drainkreis einen Widerstand anordnet. Hier wird der verstärkte Pegel(ca. 3 x) für den folgenden Mischtransistor abgenommen. Die Energie mit der hohen Schaltfrequenz des LO kommt so nicht in die Antenne. Durch die funktionale Trennung arbeiten die beiden ersten Transistoren optimal.

Die elektronische Umschaltung der Empfangsbereiche wird so ausgeführt:

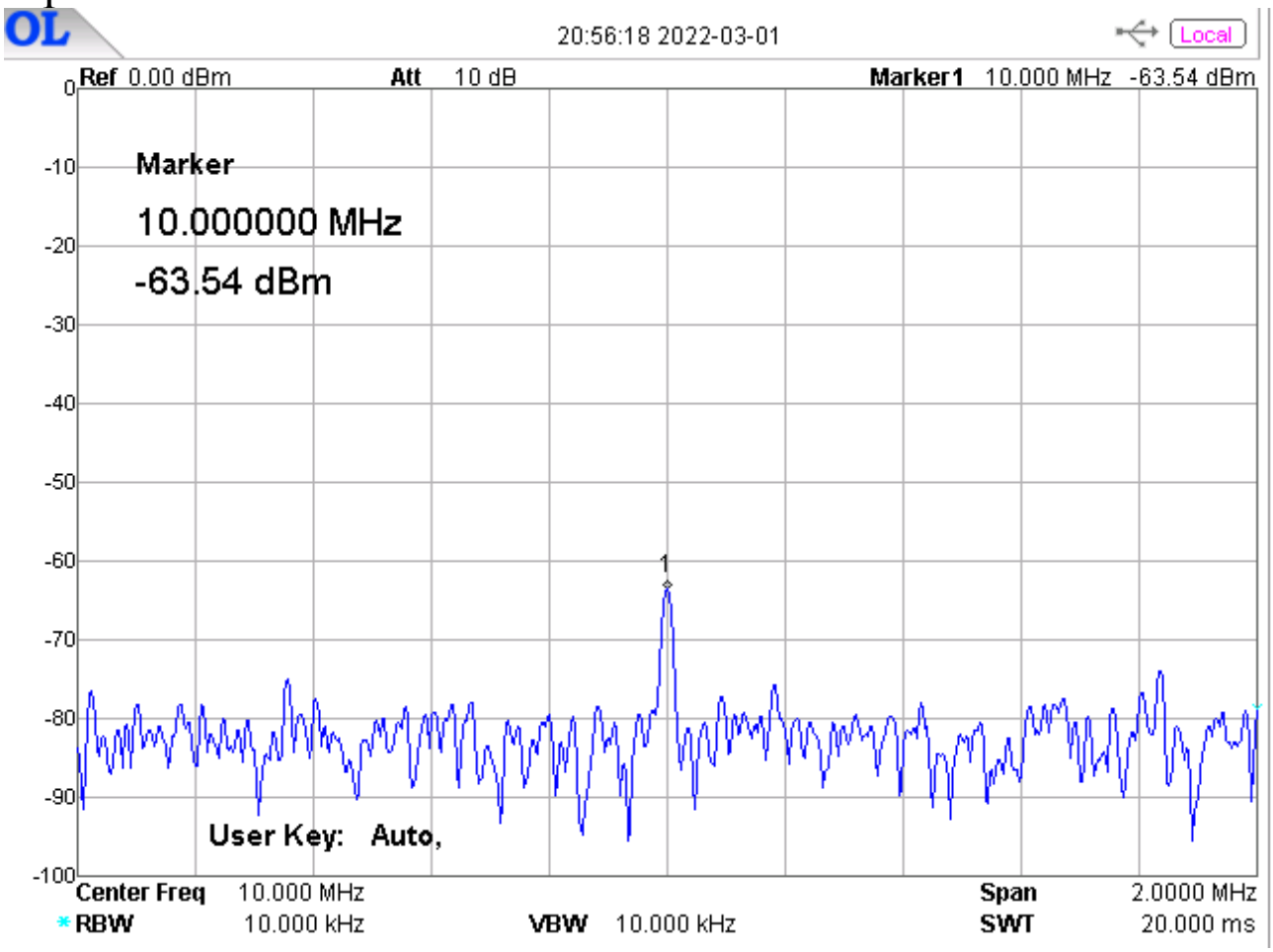


Die Software steuert die Schalttransistoren. Dabei werden die Induktivitäten ggf. durch eine Kombination zweier Spulen gebildet. Entsprechend werden die Steuerspannungen angelegt. Für den teuren BF254C kann man event. Auch den preiswerten BS170 einsetzen. Dieser Schalt-FET darf keine zu hohe Rückwärtskapazität im gesperrten Zustand haben(max. 6p).

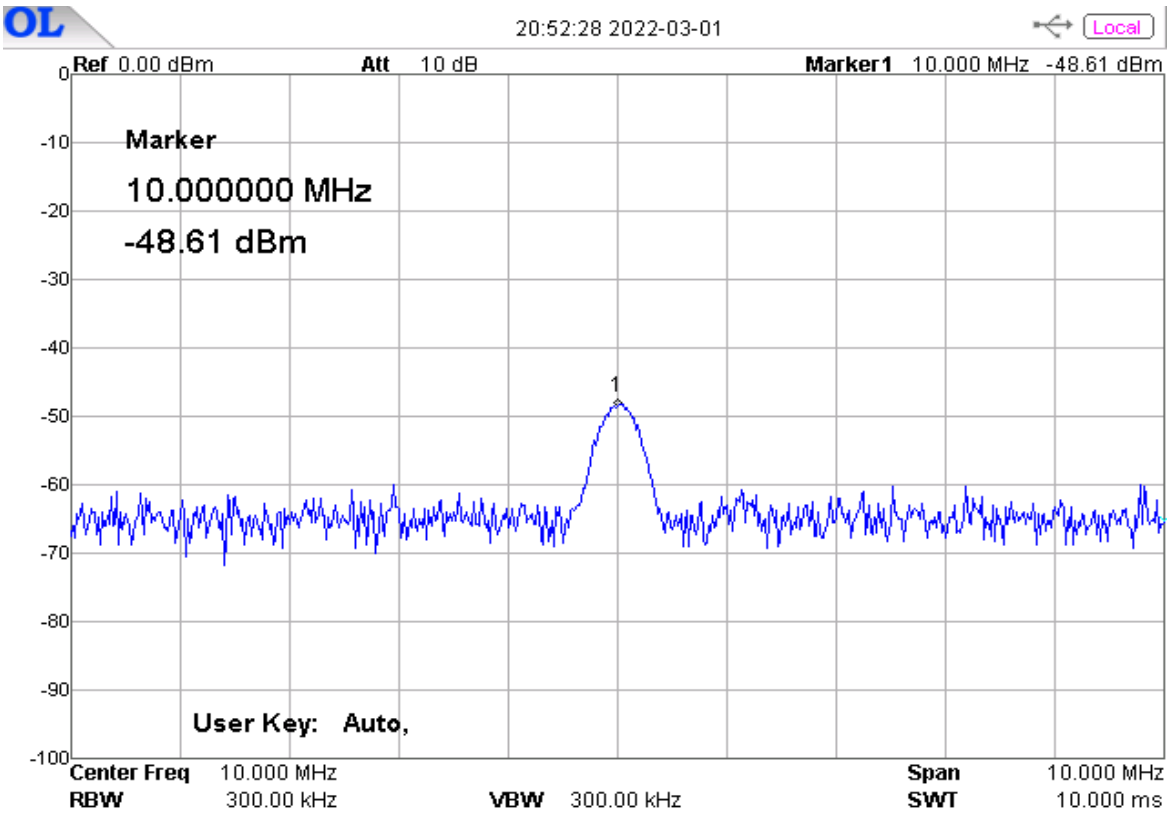
Durch das elektronische Umschalten entfallen die mechanischen Dreh-bzw. Tastenschalter. Allerdings ist hier auch eine gewisser Aufwand an Bauelementen unvermeidbar. Der ganze Spulenblock kann aber kompakt aufgebaut werden. Wobei sich die Schalter ggf. als kleine Module anfertigen lassen.

Hier Messbilder:

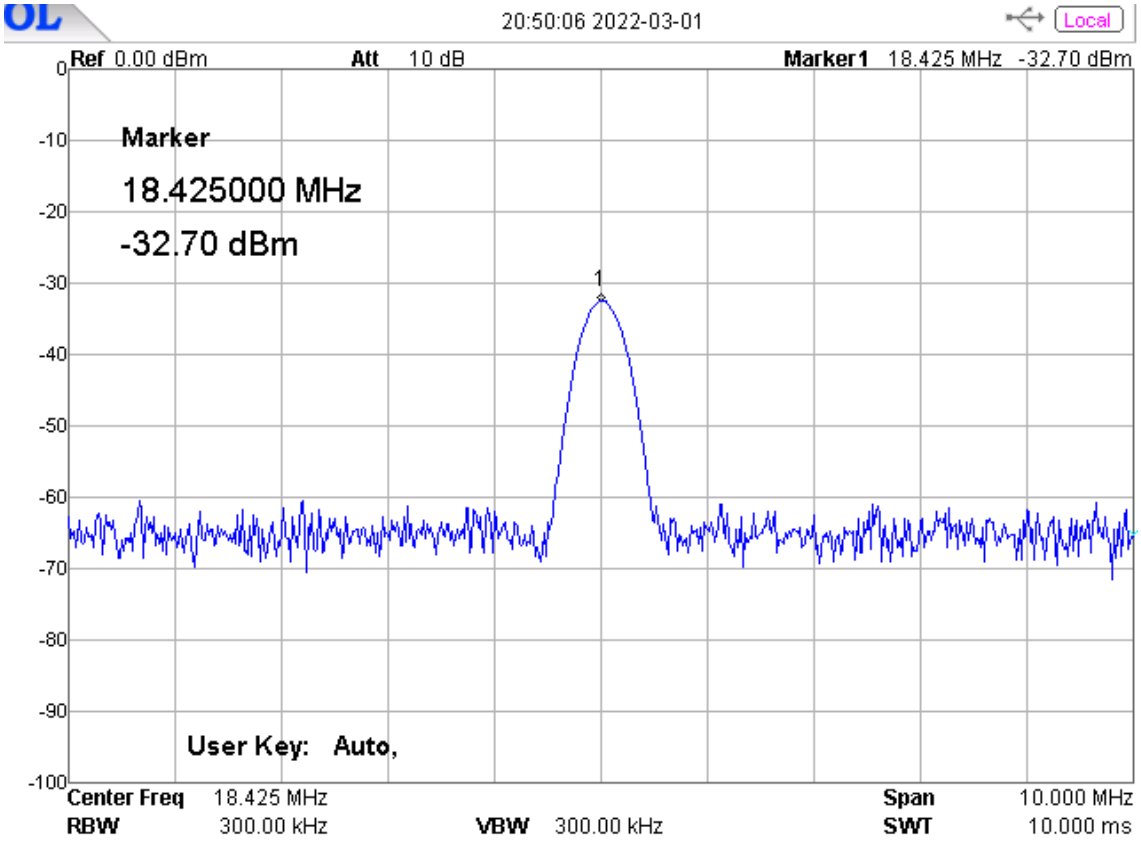
Input vom Messsender am Gate 2 des ersten Transistors



am Drain des ersten Transistors



und am Drain des zweiten Mischers

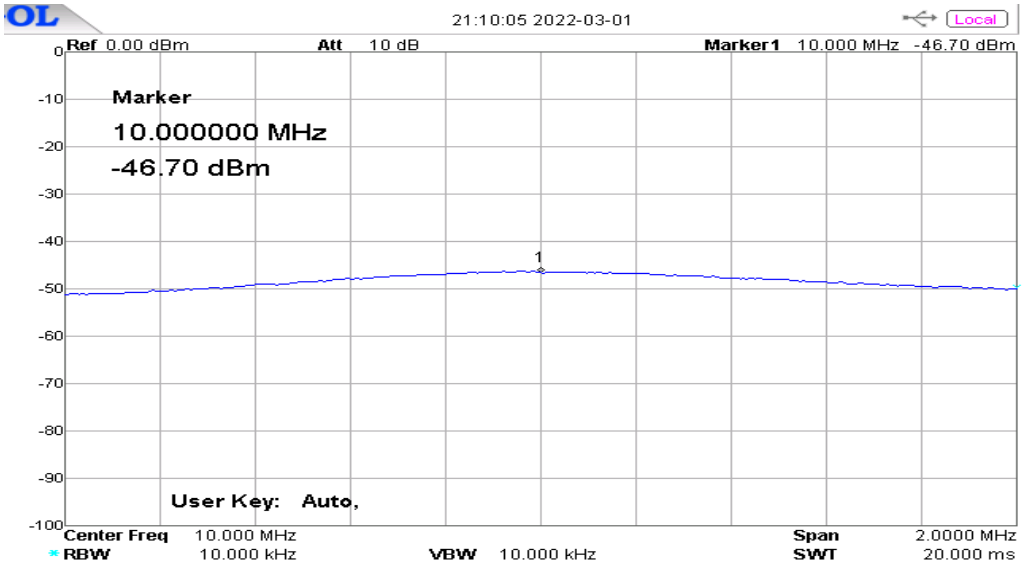


Man erkennt, dass der erste Transistor mit angezogener RK etwa

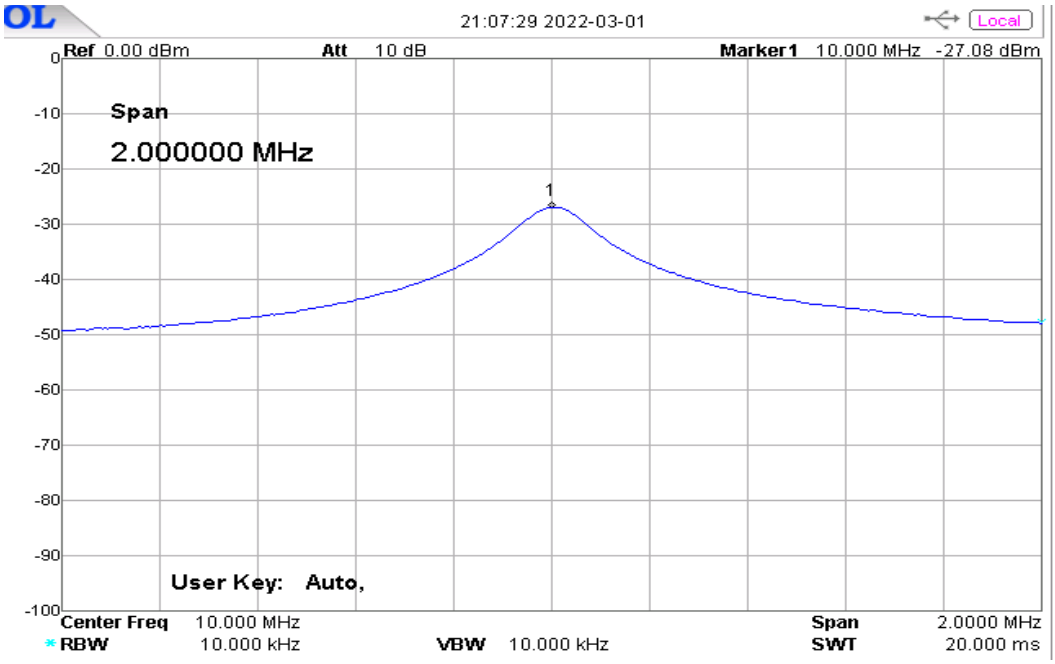
15 dB verstärkt. Der zweite Transistor mischt danach auf einen Pegel, der wieder um 15 dB höher ist. Insgesamt ist das eine mehr als ausreichende Verstärkung im Frontend. Der nachfolgende ZF-Verstärker wäre schon fast übersteuert.

Selektivität

ohne RK:



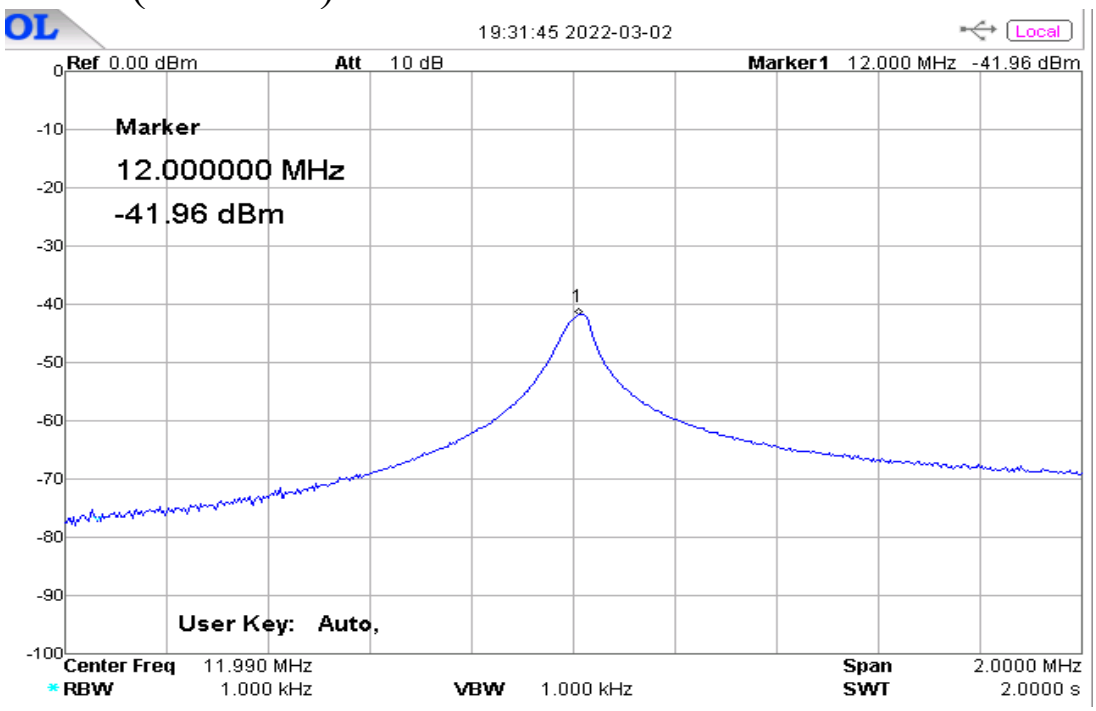
Der Pegel entspricht dem Pegel des Eingangssignals am Gate 2 des ersten Transistors. Die zweite Kurve am Drain des ersten Transistors wurde mit angezogener RK aufgenommen:



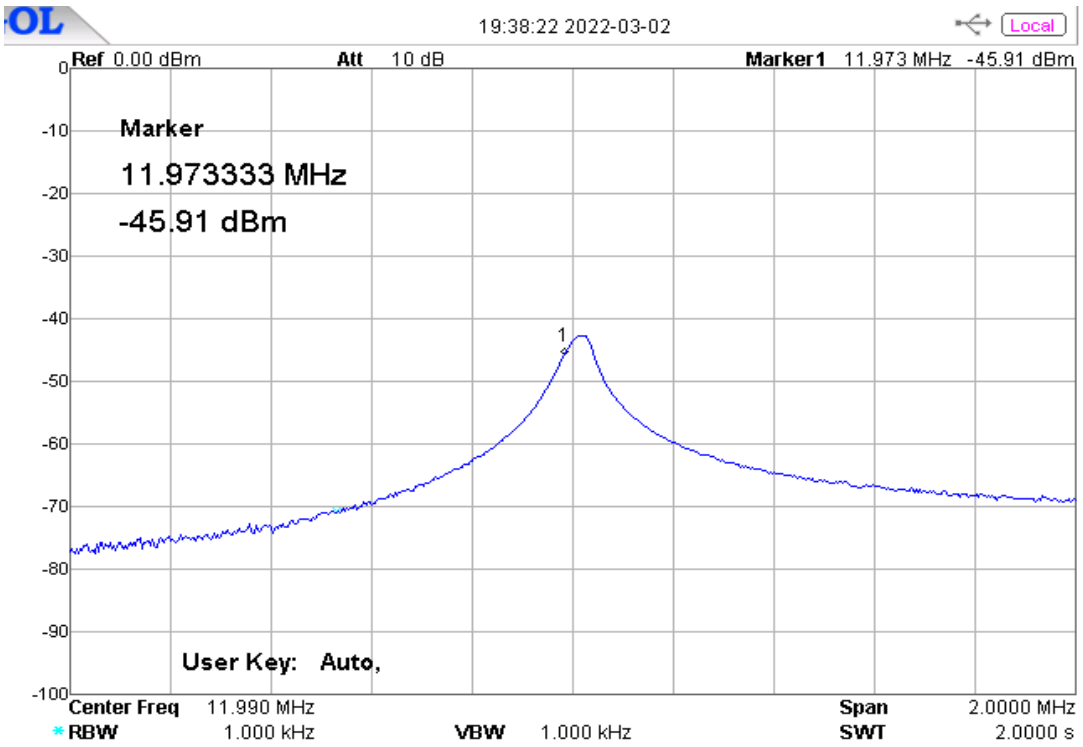
Der Pegel ist >15 dB. Die Güte des Schwingkreises ist ca. 14. Bei niedrigeren Frequenzen wird sie spürbar besser(3MHz/130). Da die -3dB-Bandbreite etwa 100 kHz ist, kann man beim Abstimmen keinen Sender verpassen. Für diesen Frequenzbereich ist die Selektivität als normal zu bewerten. Die Trennschärfe kommt aus den Filtern.

Anmerkung
Die Pegelangaben sind keine absoluten Werte. Es waren diverse Abschwächer am SA wirksam! Da die nicht verändert wurden, gelten die relativen Unterschiede.

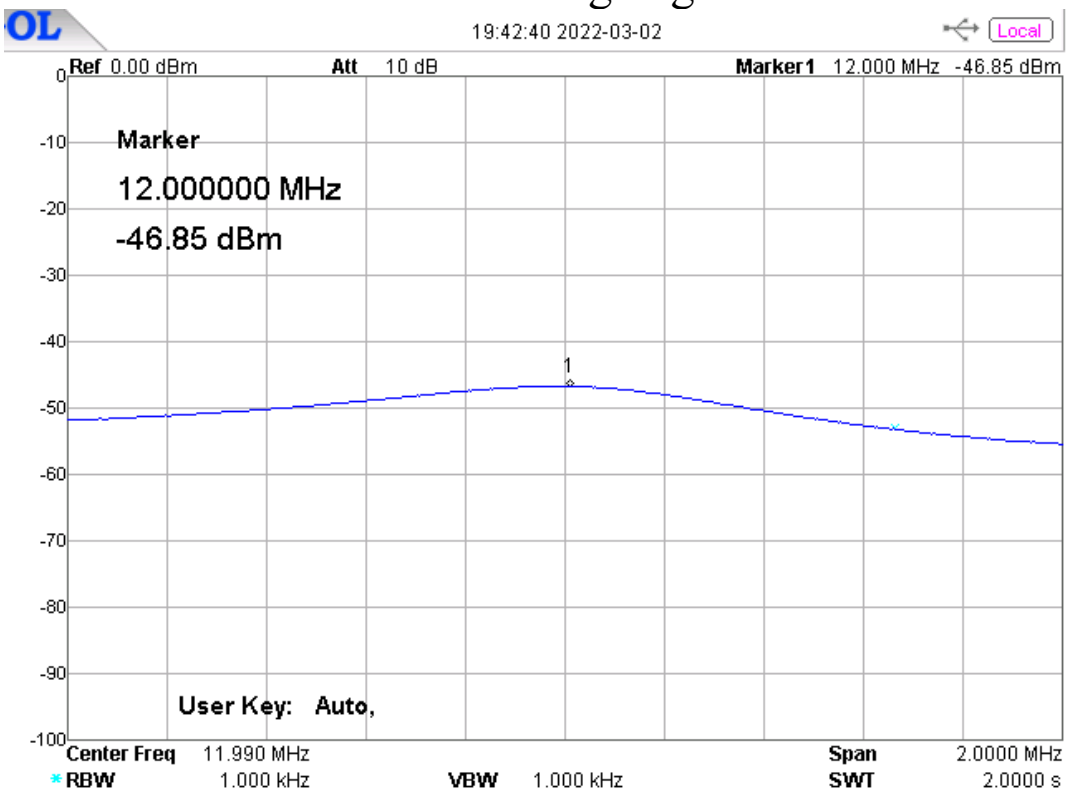
Der Schwingkreis war bei allen Messungen mit Transistoren angeschaltet, um den realen Betrieb zu emulieren. Die Pegelverluste durch die geschalteten Halbleiter sind vernachlässigbar gering. Das ist auch bei hohen Frequenzen(>20MHz) so.



Hier ist nun eine andere Schaltung zur Regelung der RK. Es wird die Gate2-Spannung geregelt. Man sieht die Einstellung kurz vor dem Schwingungseinsatz. Nachfolgend dann die -3dB-Bandbreite:

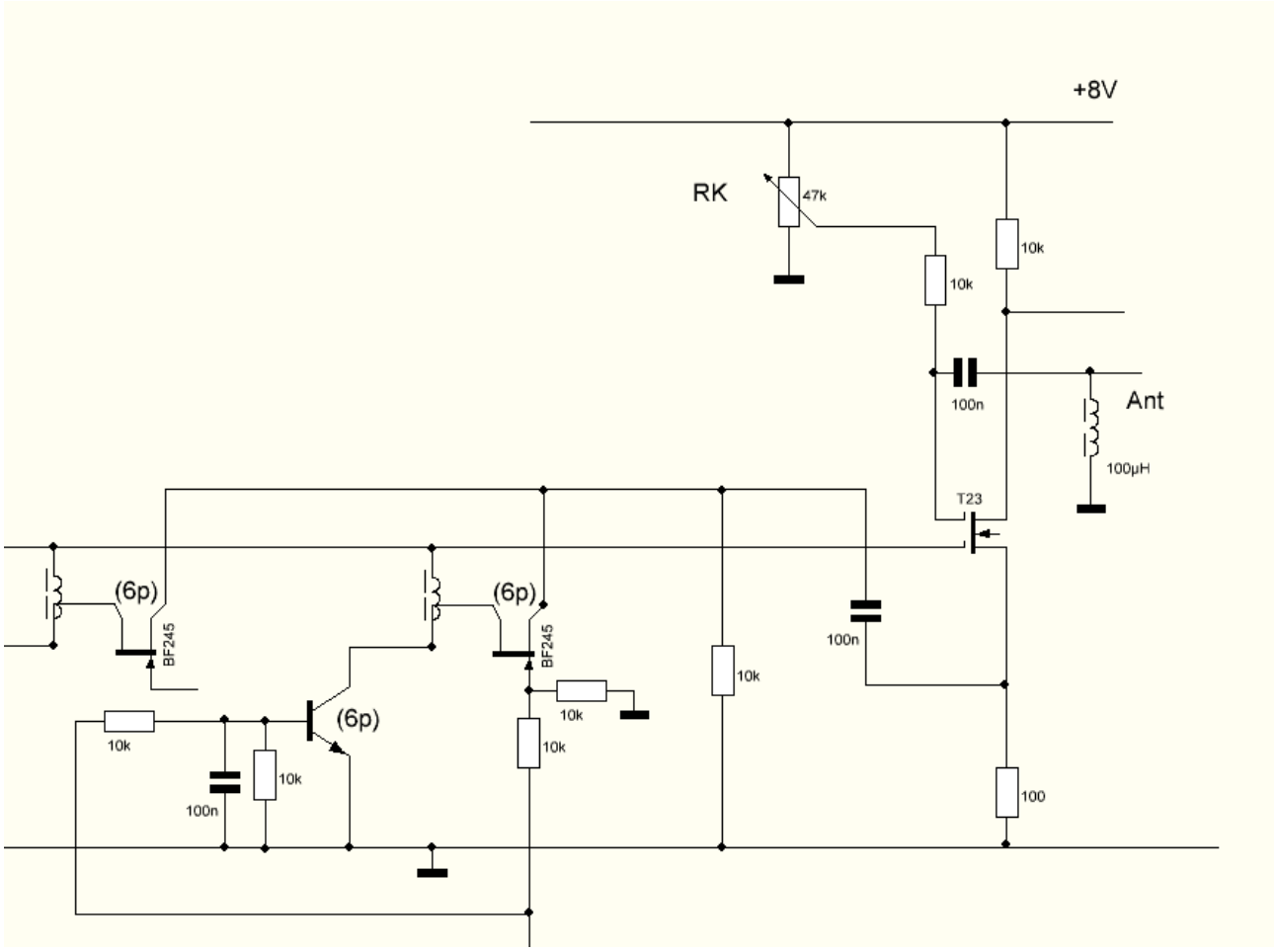


Die bandbreit ist 12/0.027 = 222, also > 200. Das ist ein guter Wert. Und abschließend noch bei abgeregelter RK:



Breit genug, um bei der Suche die Sender nicht zu überhören. Der Eingangspegel(relativ!) war -40 dBm. Die Verstärkung also nahezu 1 bei voller RK.

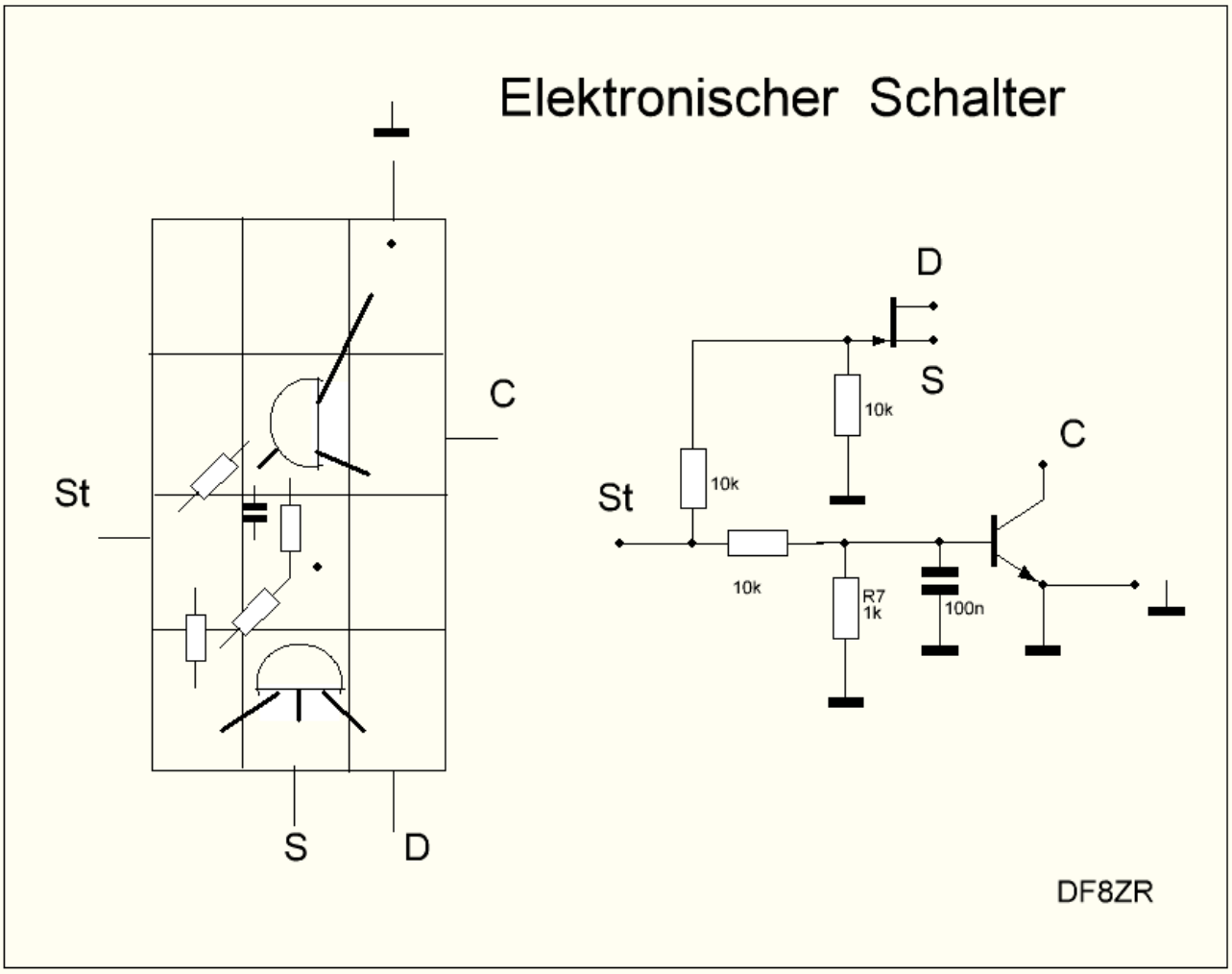
Hier die geänderte Schaltung:



Es ist die klassische Regelung der Rückkopplung, wie man sie auch in Röhrenschaltungen macht. Dort regelt man die Schirmgitterspannung. Allerdings muss man bei der Dimensionierung des Sourcewiderstandes Kompromisse machen. Je nach Frequenzbereich setzt die RK früher oder später ein. Dennoch erlaubt die großzügige Verstellweite der Gate2-Spannung eine bedienerfreundliche Einstellung.

Die Realität

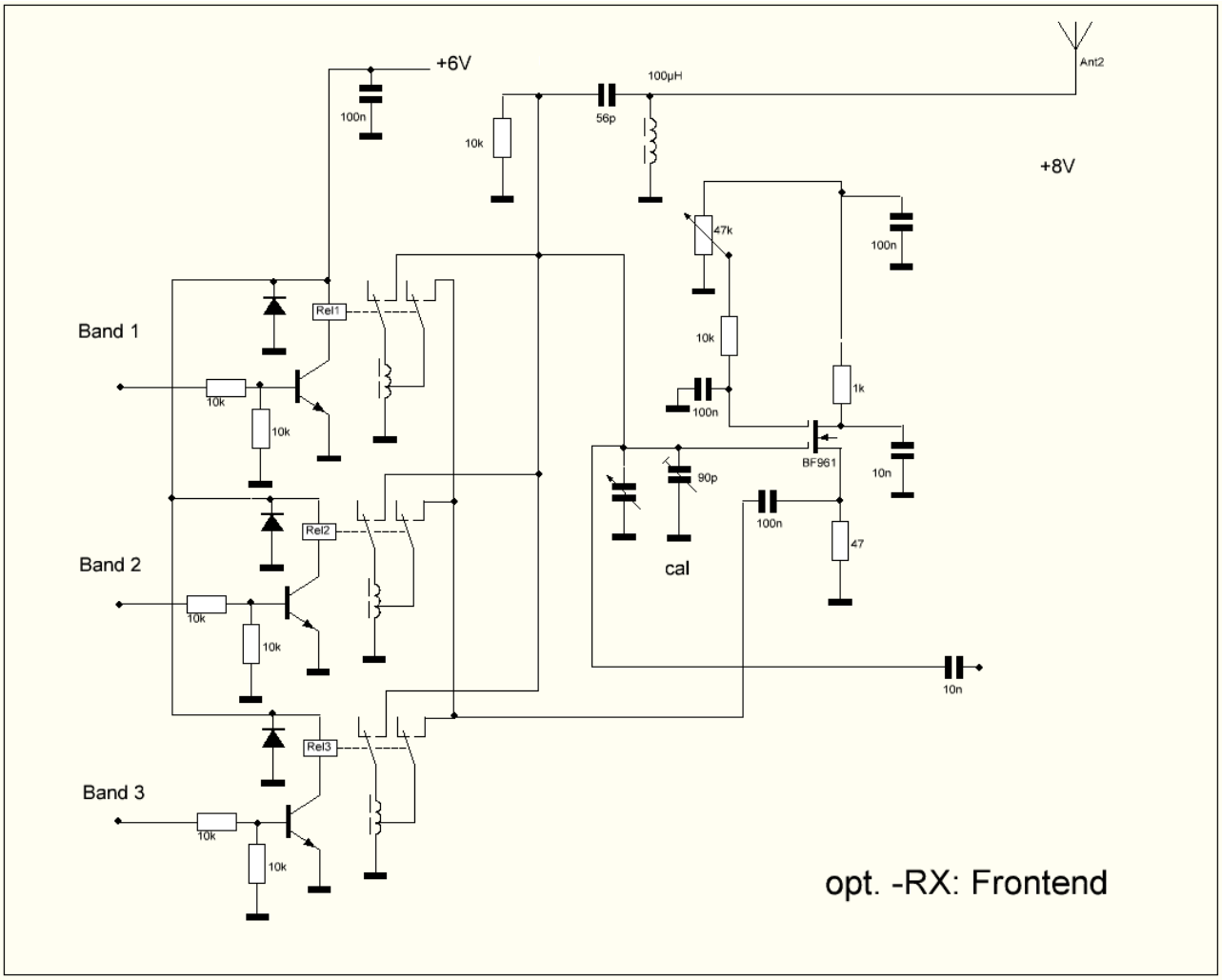
Nachdem ich für das Frontend drei Schaltermodule nach dem folgenden Bild gebaut hatte, kam der Test. Ging leider negativ aus. Man kann die nicht eingeschalteten Schwingkreise mit ihren Spulen nicht einfach parallel ans Gate hängen. Die Verluste in den nicht beteiligten Schwingkreisen(schädliche Kapazitäten) sind zu hoch. Die Resonanzkurven flachen ab. Die oberen Anschlüssen



der Spule mit FETs zu trennen gelang auch nicht zufriedenstellend. Man kann die Schalter aber durchaus an einem einzelnen Schwingkreis ohne Nachteile betreiben. Also habe ich hier die elektronische Umschaltung aufgegeben. Es kommen jetzt Relais zum Einsatz. Das vereinfacht den Aufbau und die positiven elektrischen Eigenschaften der Schwingkreise bleiben erhalten.

Neues Frontend

Nachfolgend das Schaltbild. Für die Kalibrierung des Grundzustandes der Kapazität habe ich einen Trimmer „cal“ vorgesehen. Damit kann man die größte Abstimmkapazität auf genau 100p abstimmen. Dann stellen sich auch die Frequenzbereiche der Bänder1 bis 3 ein.

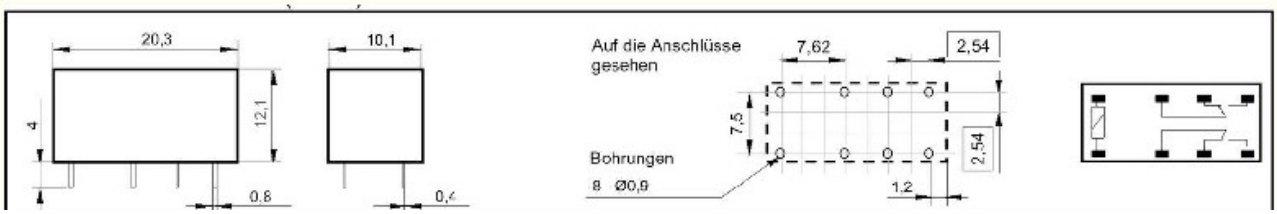


Man sieht die Auskopplung am Gate1. Hier entsteht durch Resonanzüberhöhung ein Gewinn von 10 ...15 dB, der sich am Drain des MOSFETs nicht bemerkbar macht. Die Spannung steuert hochohmig den folgenden Transistor.

Erkenntnis

Das Konzept kann gebaut werden! Die Relais sind bestellt. Jetzt folgt die Arbeit:

maluala M4-5H



Kleines Problem nebenbei
Es hat sich gezeigt, dass die Frequenzbereiche am Beginn des Bandes nicht ganz überlappten. Habe daher einen Drehko mit 265p statt 160p vorgesehen. Jetzt überdeckt dieser in weiten Bereichen die Bänder.

Den C-Trimmer „cal“ kann man weglassen. Lediglich für die hohen Frequenzen müssen die im Drehko eingebauten Trimmer vor dem Einbau auf kleinstes C eingestellt werden.

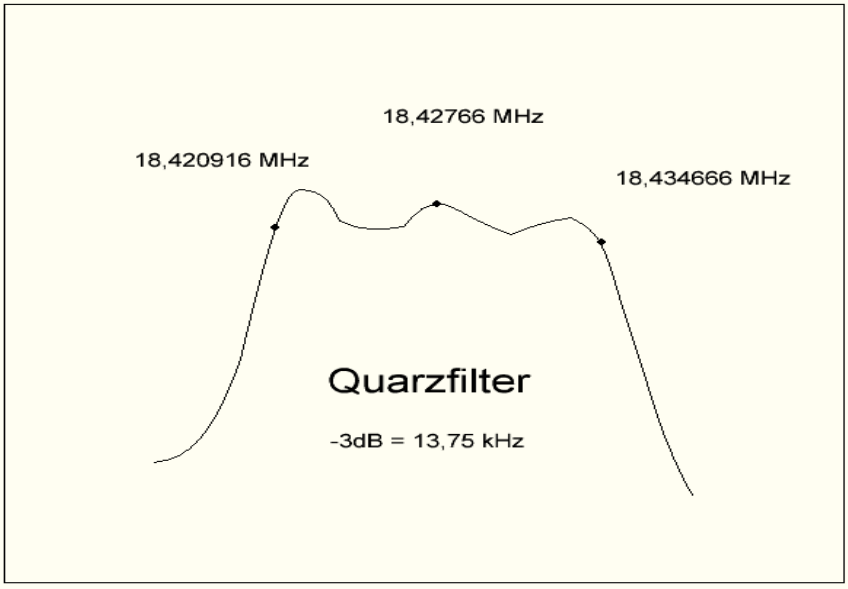
Selektivität
Es hat sich gezeigt, dass bei angezogener RK und optimaler Vorkreisabstimmung es bereits bei der geringste Vorspannung am Gate2 die Schaltung schwingt. Ich musste die Vorkreise daher mit einem 10k dämpfen. Dadurch wird auch die Suche erleichtert, weil man über einen weiten Bereich(0,5MHz bei 6 MHz) mit ausreichender Empfindlichkeit empfangen kann. Hört man einen Sender, dann kann man die RK erhöhen und die Lautstärke steigern. Wegen des sehr geringen Rauschens dieses Empfängers ist es nicht so einfach, eine eindeutige Abstimmung des Vorkreises zu finden. Wenn man dazu die RK steigert, stellt sich zunächst ein lauteres Rauschen ein, dann aber gerät das Frontend ins Schwingen. Man nimmt also die RK wieder zurück, bis die Schwingungen abreißen. Hier ist dann der optimale Abstimmbereich gefunden.

Aber manchmal ist es nicht zu vermeiden, dass man auf ein Pseudooptimum hereinfällt. Das passiert dadurch, dass der Vorkreis derart die Signalstärke eines anderen Senders steigert, sodass hier Eigenschwingungen eintreten. Der falsche Sender wird durch die schwingende RK in den Empfangsbereich gezogen. Zusammen mit diesen Eigenschwingungen setzt sich das Pseudosignal durch und ist laut zu hören. Es ist aber auf der falschen Frequenz und stimmt nicht mit der Anzeige überein.

Fazit: man muss den richtigen Umgang mit diesem Empfänger üben, um die Vorzüge zu genießen.

Das Quarzfilter

Der Aufbau ist wie beim MW-RX. Hier mal die Durchlasskurve:



Empfindlichkeit

Sie ist hoch genug, um sogar Amateurfunk zu empfangen. Allerdings hier nicht dekodierbar. Bei 6 MHz habe ich eine Grenzempfindlichkeit von 10 uV gemessen. Das ist für den Rundfunkempfnag mehr als ausreichend. Wien1, den Sender des ORF, empfange ich mit 1m Draht im Keller lautstark und bei angezogener RK rauschfrei.

Die Trennschärfe wird überwiegend durch das Quarzfilter(B=10kHz) bestimmt. Allerdings macht das nachfolgende Keramikfilter den Kanal wieder schmaler. Etwa 4...5 kHz ist die AM-modulierte Station mit akzeptabler Qualität zu hören. Eher etwas dunkler im Ton, weil die Höhen fehlen. Aber dadurch wird auch das lästige Rauschen geringer. Jedenfalls ist „Ruhe“ im Kasten, wenn man Sender sucht. Kommt aber einer rein, dann hört man ihn ganz sicher. Selbst auch dann, wenn der

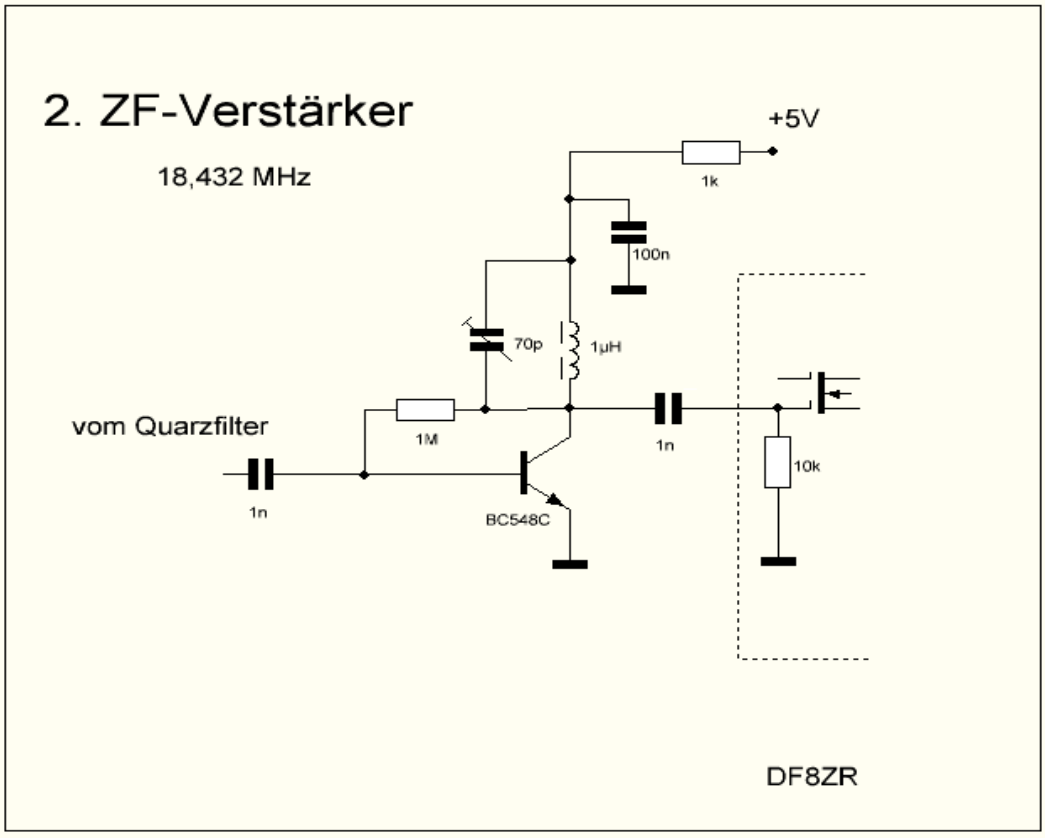
RK-Regler ganz links steht. In der Regel möchte man aber doch nur in einem Bereich einen Rundfunksender bei bekannter Frequenz einstellen. Dann dreht man soweit, bis die Anzeige stimmt. Erst dann betätigt man die Vorkreisabstimmung. Und zuletzt ganz vorsichtig die RK.

Handhabung

Durch das hier realisierte Prinzip ist die Bedienung des Radios unüblich. Nach kurzer Zeit aber hat man es raus, damit umzugehen. Mein Ziel war es ja, das Rauschen gerade beim Empfang von schwachen Sendern gering zu halten. Das ist mir vielleicht gelungen. Jedenfalls ist mir in meiner langjährigen Praxis eigentlich noch nie ein Superhetempfänger mit einer Rückkopplung im Vorkreis untergekommen. Will aber nicht ausschließen, dass man das bereits in den frühen Jahren der Empfängerentwicklung überdacht hatte. Schließlich war es ja der Wunsch, dem Konsumenten eine bequeme „Einknopfbedienung“ anzubieten. Das ist aber nicht meine Priorität beim Basteln der Radios. Mir geht es darum, Schaltungen zu testen, die mit einfachen Mitteln gute Ergebnisse liefern. Die Neugier auf das technisch Machbare verführt mich immer noch, den Lötkolben in die Hand zu nehmen.

Pegelplan

Wenn man ein Radio entwickelt, muss man sich über einen Pegelplan Gedanken machen. Zunächst dachte ich, den brauchste nicht. Doch dann zeigte sich eine zu geringe Empfindlichkeit. Der zweite Mischer dämpfte um 10 dB. Der Schaltkreis TA7640 für die zweite ZF brachte nicht die erforderliche Verstärkung. Also fügte ich zwischen dem Ausgang des Quarzfilters und dem zweiten Mischer einen kleinen ZF-Verstärker ein, der die Dämpfung wieder kompensierte. Erst danach hatte der Empfänger eine ausreichende Grundempfindlichkeit auf allen drei Bändern.

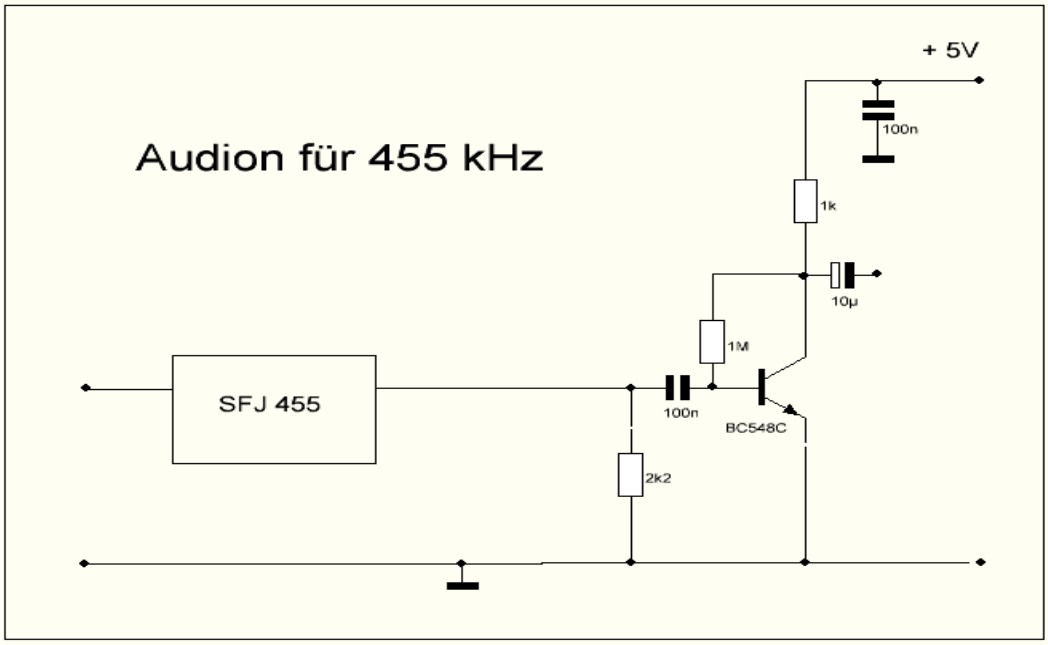


Der 10k am Gate1 des folgenden zweiten Mischers dämpft den Schwingkreis am Kollektor. Er macht dem Kreis breitbandig und verhindert das Eigenschwingen. Die Verstärkung ist genau +10dB.

Erweiterungen
Einen Demodulator für LSB/USB hatte ich ursprünglich vorgesehen, wollte ich aber doch nicht mehr nachrüsten. Für den Empfang der SSB-Stationen habe ich andere Empfänger. Dieser hier soll ein Kurzwellenempfänger für den Rundfunk sein, mehr nicht.

Weg mit dem IC !
Nachdem ich nochmal einen Versuch mit einem anderen IC machte(TDA1046), habe ich diese Schaltkreise einfach mal weggelassen und durch ein Audion ersetzt. Der Ton wurde jetzt viel klarer, kaum noch Verzerrungen, jedoch keine Schwundregelung. Doch darauf kann ich verzichten. Ich habe ja noch den HF-Pegelregler. Und bei den derzeitigen schlechten

Empfangsbedingungen muss man sich mit dem Schwund abfinden. Das Audion hat am Eingang einen auf 455 kHz abgestimmten Schwingkreis. Am Kollektor eines BC548C ist ein 1k. An dem wird die NF abgenommen.

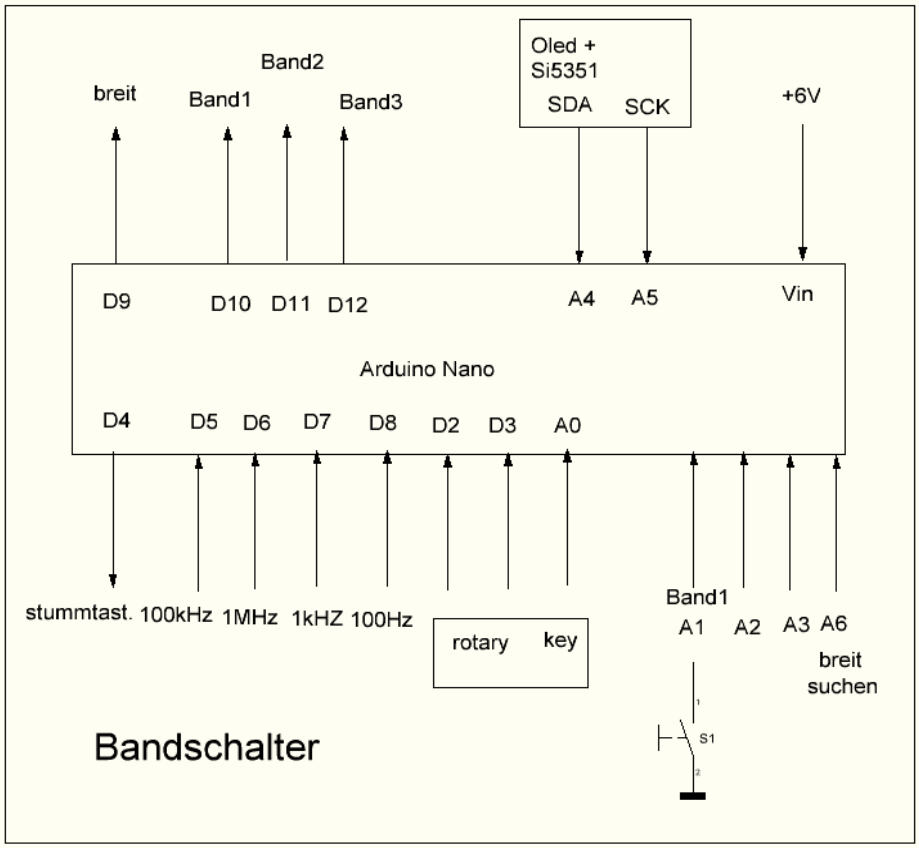


Es hat sich gezeigt, dass große Probleme mit dem LM386 entstehen können. Er verstärkt auch andere Frequenzen und über das Audion stellt sich dann eine Rückkopplung für alle möglichen Frequenzen ein. Ich konnte diese Schwingungen nur dadurch unterdrücken, in dem ich vor das Audion noch ein keramisches Filter einfügte. Jetzt kam nur noch das Nutzsignal durch und den NF-Ausgang des Audions filterte ich noch zusätzlich mit einer Drossel 1mH. Danach war Ruhe im Kasten und ich konnte die nun viel sauberer klingende NF genießen.

Schaltungen

Die weiteren Schaltungen(2. Mischer, 2. ZF-Verst und das NF-Teil) sind baugleich wie beim MW-RX. Bitte dort die Einzelheiten entnehmen.

Die Belegung des Arduino Nano
Die Anschlüsse(Ports) wurden knapp. Mit Einschränkungen könnte man gerade noch ein S-Meter hinzufügen. Doch ich verlasse mich lieber auf meine Ohren. Man braucht keine optische Hilfe beim Abstimmen. Hier die Belegung:



Ursprünglich wollte ich den Vorkreis „breitbandig“ schalten(A6), um die Sendersuche zu erleichtern. Aber die Dämpfung mit 10k reicht aus, um einen größeren Frequenzbereich ohne Nachstimmen zu überstreichen.

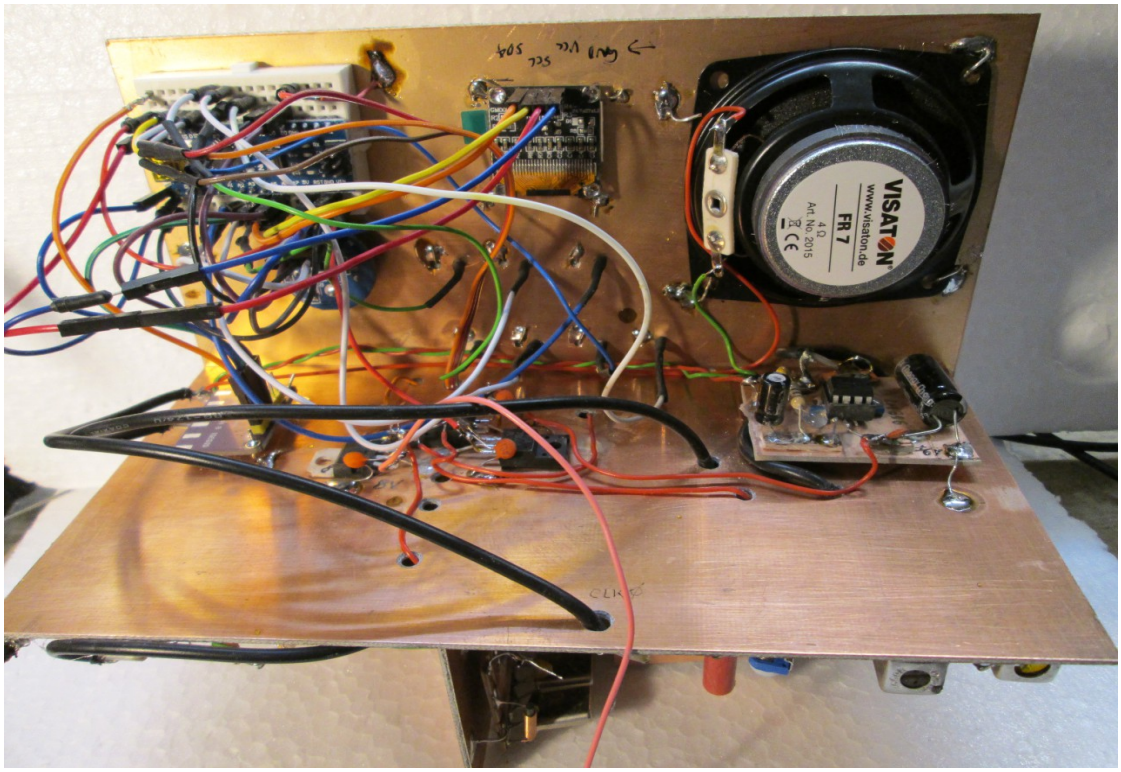
Und die Demo bei YouTube:

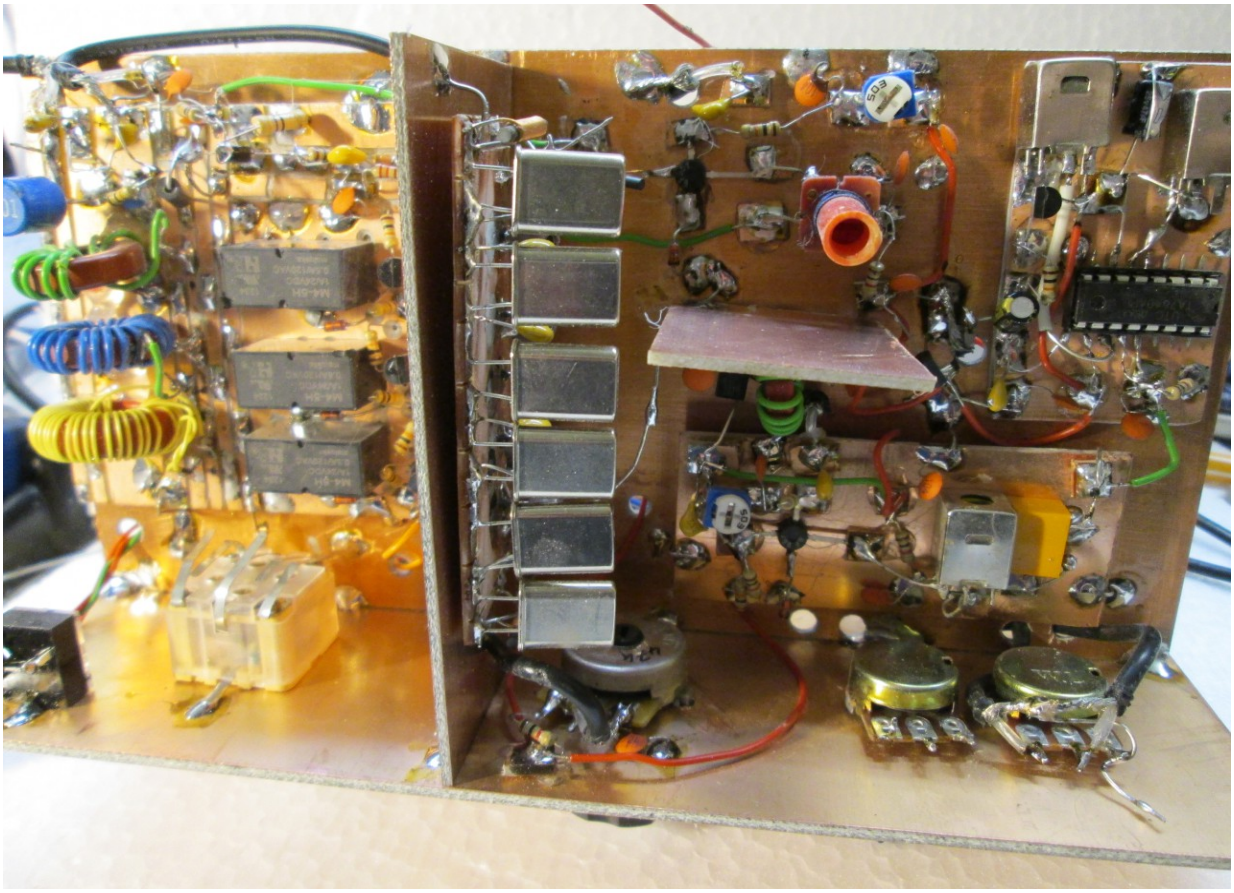
https://www.youtube.com/watch?v=1-7DbhJE2nI

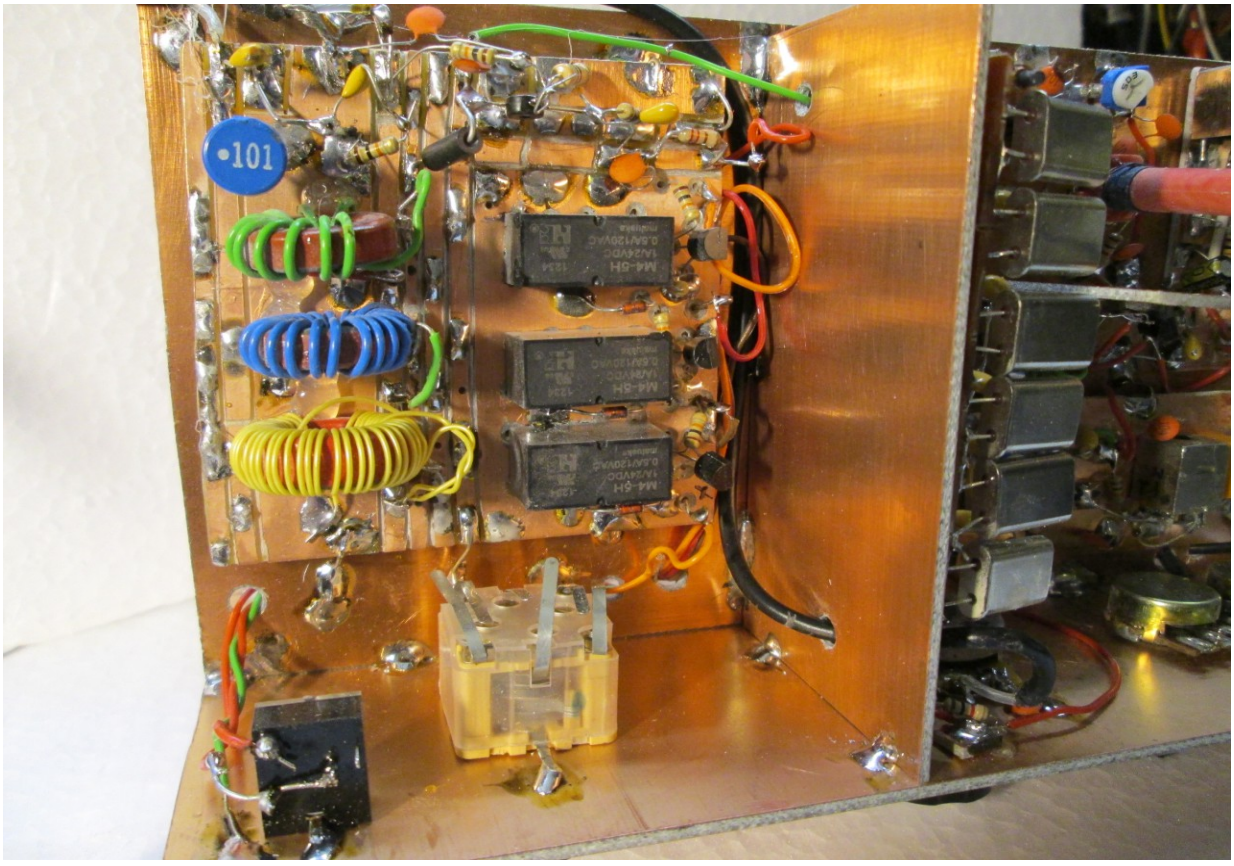
Viel Spaß beim Nachbau

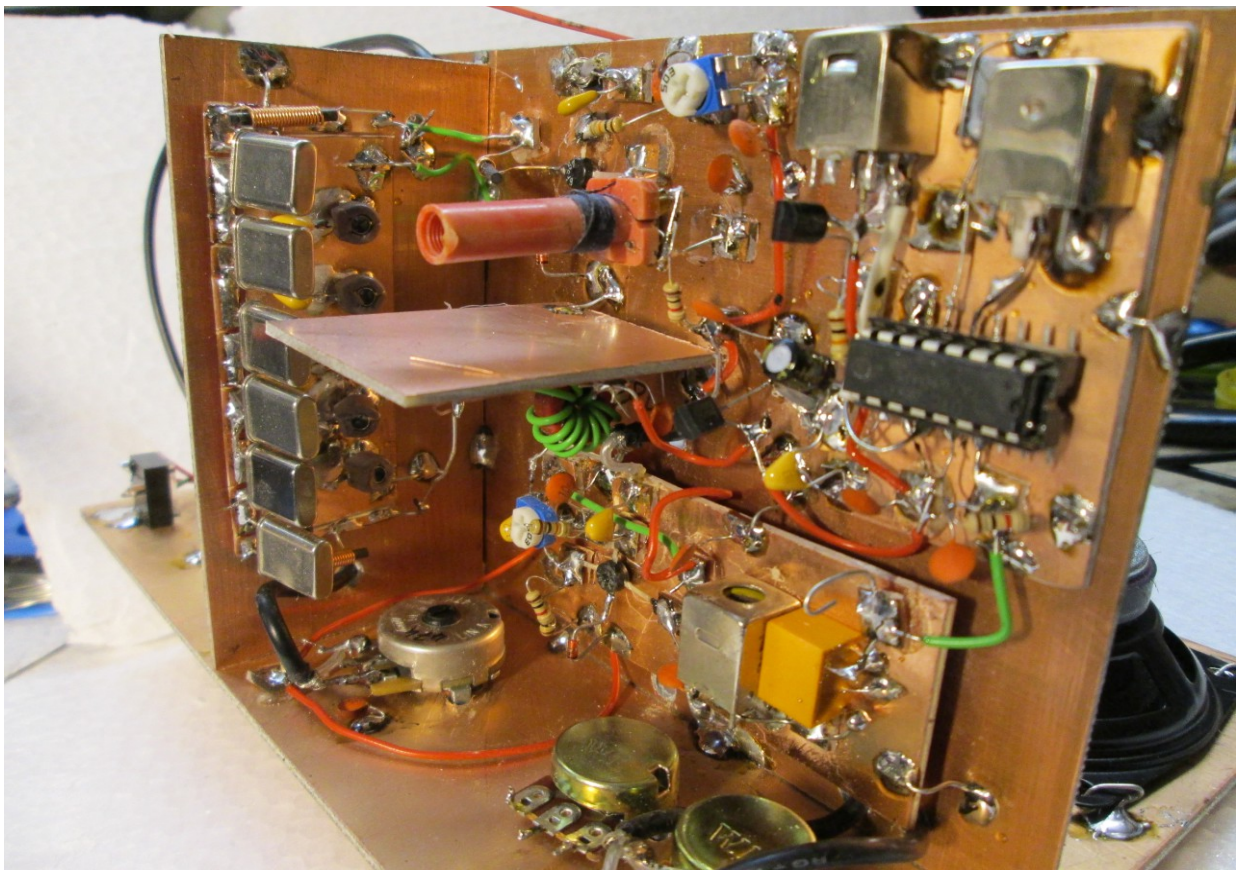
DF8ZR; im Febr. 2022

...es folgen Fotos und die Software:






•101



Software:

/
****
****

RK-Superhet --> Doppelsuper mit Rückkopplung im Frontend

10kHz to 120MHz VFO / RF Generator with Si5351 and Arduino Nano, with Intermediate Frequency (IF) offset (+ or -). See the schematics for wiring details. By J. CesarSound - ver 1.0 - Dec/2020.

Private Anwendung und Anpassung bei DF8ZR im Jan. 2022: Direktmischer für das 80m-Amateurfunkband.
Mit Dank an den Autor J. Cesar, bei dem alle Rechte sind.

BFO-Test: Hier werden die Tasten 100k und 5k für die Feinabstimmung der BFO-Frequenz verwendet. Im BFO-Betrieb

```
  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.
**********************************************************
**********************************************************/

//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  3000000 //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 18426       // 2. ZF in kHz angeben
#define IF2 18882.7              // 2. LO  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 = 1;//????????????????????????
unsigned int period = 100;//100;   //millis display active
unsigned long time_now = 0;  //millis display active




//Tastenklicks-------------------------------------------------------------
---
#define MEGA_PIN 5            //D5
#define HUNDERTK_PIN 6        //D6
#define KILO_PIN 7            //D7
#define HUNDERT_PIN 8         //D8

#define MUTEN_PIN 4        //D4 für mute die Schaltklicks

#define BREIT_PIN 9        //D9 für Breitbandbetrieb: Suchen

#define SUCHEN_PIN A6      //A6 Input-Taste für Breitbandbetrieb: Suchen
int breitband=0;

int tastenklick;
int hundert = 0;//für frequency
int eink = 0;//für frequency
int mega = 0;//für frequency
int hundertk = 0;//für frequency
//int zehn;//für frequency
```

```
int bnd1 = 0; //Merkmal für Bandbereich
int bnd2 = 0;
int bnd3 = 0;




//------------------------------------------------------------------------
--
//hier nun der Teil des Bandumschalters

//die Eingabepins
#define Band1 A1
#define Band2 A2
#define Band3 A3



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



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 setup() {

//pinMode(ZEHN_PIN, INPUT_PULLUP);
pinMode(HUNDERT_PIN, INPUT_PULLUP);
pinMode(KILO_PIN, INPUT_PULLUP);
pinMode(MEGA_PIN, INPUT_PULLUP);
pinMode(HUNDERTK_PIN, INPUT_PULLUP);


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

//Band setzen
 pinMode(Band1, INPUT_PULLUP);
 pinMode(Band2, INPUT_PULLUP);
 pinMode(Band3, INPUT_PULLUP);
```

```
//die Ausgabepins für das Frontend

  pinMode(10, OUTPUT);     //Band 1 schalten
  pinMode(11, OUTPUT);     //Band 2 schalten
  pinMode(12, OUTPUT);     //Band 3 schalten

pinMode(4, OUTPUT);     //stummschalten
digitalWrite(4, LOW);    //  "

pinMode(9, OUTPUT);     //suchen = Breitbandbetrieb
digitalWrite(9, LOW);   //  "

pinMode(A6, INPUT);     //suchen = Breitbandbetrieb
digitalWrite(9, HIGH);   //  "


//zunächst alle LOW setzen


digitalWrite(10, LOW);   //Band 1
digitalWrite(11, LOW);   //Band 2
digitalWrite(12, LOW);   //Band 3




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

  pinMode(2, INPUT_PULLUP);//rotary D2
  pinMode(3, INPUT_PULLUP);//   "   D3
```

```
  pinMode(tunestep, INPUT_PULLUP);//A0



  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();



//-----------------------------------------------
  //Anfangsanzeigen:

  stp = 3;            //1kHz setzen und anzeigen am Start

  setstep();

  bnd1 = 1;//Bandanzeige auf Band1


  displayfreq();


  layout();

}

//_____ENDE
SETUP_____________________________________________________
_________
```

```
//#########################################################
```

```
######### LOOP ######################

void loop() {


if (tastenklick == 0){

  if (freqold != freq) {
    time_now = millis();
   stumm();
   tunegen();
    freqold = freq;

    }
   digitalWrite(4, LOW);
}


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


}

 /* if (digitalRead(A6) == LOW) {   //suchen gedrückt
FEHLER!!!
    test();
    suchen();
    delay(300);


}*/
```

```
//Abfrage der Tasten für Stepfrequenz



if (digitalRead(HUNDERT_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;
 hundert = 1;
 eink=0;
 hundertk=0;
 mega=0;

 tasten();


}



if (digitalRead(KILO_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;
 eink = 1;
 hundert = 0;
 hundertk=0;
 mega=0;
 tasten();

}



if (digitalRead(MEGA_PIN) == LOW) {
    time_now = (millis() + 300);
```

```
tastenklick=1;  //Normalbetrieb
//tastenklick=0;//**********************BFO-Test
 mega = 1;
 hundert = 0;
 eink=0;
 hundertk=0;

 tasten();

}

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

//tastenklick=0;//**************************BFO-Test
 hundertk = 1;
  mega = 0;
 hundert = 0;
 eink=0;

 tasten();

}




//jetzt die Bandabfrage

if (digitalRead(Band1) == LOW) {
    time_now = (millis() + 300);
bnd1=1;//
bnd2=0;
bnd3=0;
```

```
freq = 3000000;
tunegen();
eink = 1;
 hundert = 0;
 hundertk=0;
 mega=0;
 tasten();
displayfreq();
bandschalten();

}

if (digitalRead(Band2) == LOW) {
    time_now = (millis() + 300);
bnd2=1;//Normalbetrieb
bnd1=0;
bnd3=0;
freq = 5000000;
tunegen();
eink = 1;
 hundert = 0;
 hundertk=0;
 mega=0;
 tasten();
displayfreq();

bandschalten();

}


if (digitalRead(Band3) == LOW) {
    time_now = (millis() + 300);
    bnd3=1;
    bnd1=0;
```

```
    bnd2=0;
    freq = 10000000;
tunegen();
eink = 1;
 hundert = 0;
 hundertk=0;
 mega=0;
 tasten();
displayfreq();

bandschalten();

}




  if ((time_now + period) > millis()) {
    displayfreq();
    layout();
  }

  delay(100);//für Tasten setzen, sonst Dauersetzen


}

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

void test(){
```

```
display.clearDisplay();

  display.setTextSize(2);
  display.setCursor(4, 50);//ganz unten


display.print("OK");

display.display();



  exit(0);


}



void startup_text() {
display.clearDisplay();

  display.setTextSize(2);
  display.setCursor(5, 1);
  display.print(" RK-Super");
  display.setTextSize(2);
  display.setCursor(4, 20);
  //display.print(" RK-Superhet ");
  display.setCursor(4, 35);
  display.print("  DF8ZR");


  display.display();
```

```
  delay(2000);//3000

}


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("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.setTextSize(2);
 display.setCursor(4, 50);
```

```
 display.print("");



if(bnd1 == 1) display.print(" 3 - 9 MHz");
if(bnd2 == 1) display.print("5 - 15 MHz");
if(bnd3 == 1) display.print(" 10-30 MHz");

display.display();//aktualisieren


}



void bandschalten(){

if (bnd1 == 1) digitalWrite(10, 1);
else digitalWrite(10, 0);

if (bnd2 == 1) digitalWrite(11, 1);
else digitalWrite(11, 0);

if (bnd3 == 1) digitalWrite(12, 1);
else digitalWrite(12, 0);


}


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
//interfreq= 18425.0 - 0.4918;  // 18.9258 gemessen
//interfreq2  = 17970.0 - 1.7; // 17.9697

//interfreq= 18424.5;  // 18.9258
//interfreq2  = 17968.3; // 17.9697

//interfreq= 18424;  //
//interfreq2  = 17968; //

   ///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 (mega == 1) {

stp = 1;//5; //1 für MHz
      fstep = 1000000; // statt 10kHz-steps werden 1MHZ-steps
eingestellt!


}



if (eink == 1) {

stp = 4;
      fstep = 1000;
}

if (hundertk == 1) {

stp = 6;
     fstep = 100000;

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

}

//nach dem Setzen und Anzeigen:
//zehn=0;
//hundert = 0;
```

```
//eink = 0;
//mega = 0;
//hundertk = 0;

tastenklick = 0;

tunegen();//den generator auf die Frequenz setzen
freqold = freq;

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

}



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 stumm(){

digitalWrite(4, HIGH);

}

void suchen(){

if (breitband == 1){ breitband = 0;
 digitalWrite (BREIT_PIN,LOW);
}
else if(breitband == 0 ) {
  digitalWrite (BREIT_PIN,HIGH);//eingeschaltet, LED leuchtet
breitband = 1;
}
}



//hier folgt Code, der nur für die Optimierung der BFO=-Frequenz
unkommentiert werden soll /**********BFO-Test

/*void bfo(){

//#define IF2 49137             //LSB-BFO-Frequenz
```

```
if (stp==6) {

freqbfo = freqbfo + 1;

si5351.set_freq_manual((0 + (freqbfo * 100ULL)) * 100ULL,
pll_freq, SI5351_CLK1);

}


if (stp==5) {

freqbfo = freqbfo - 1;

si5351.set_freq_manual((0 + (freqbfo * 100ULL)) * 100ULL,
pll_freq, SI5351_CLK1);


}*/
```
