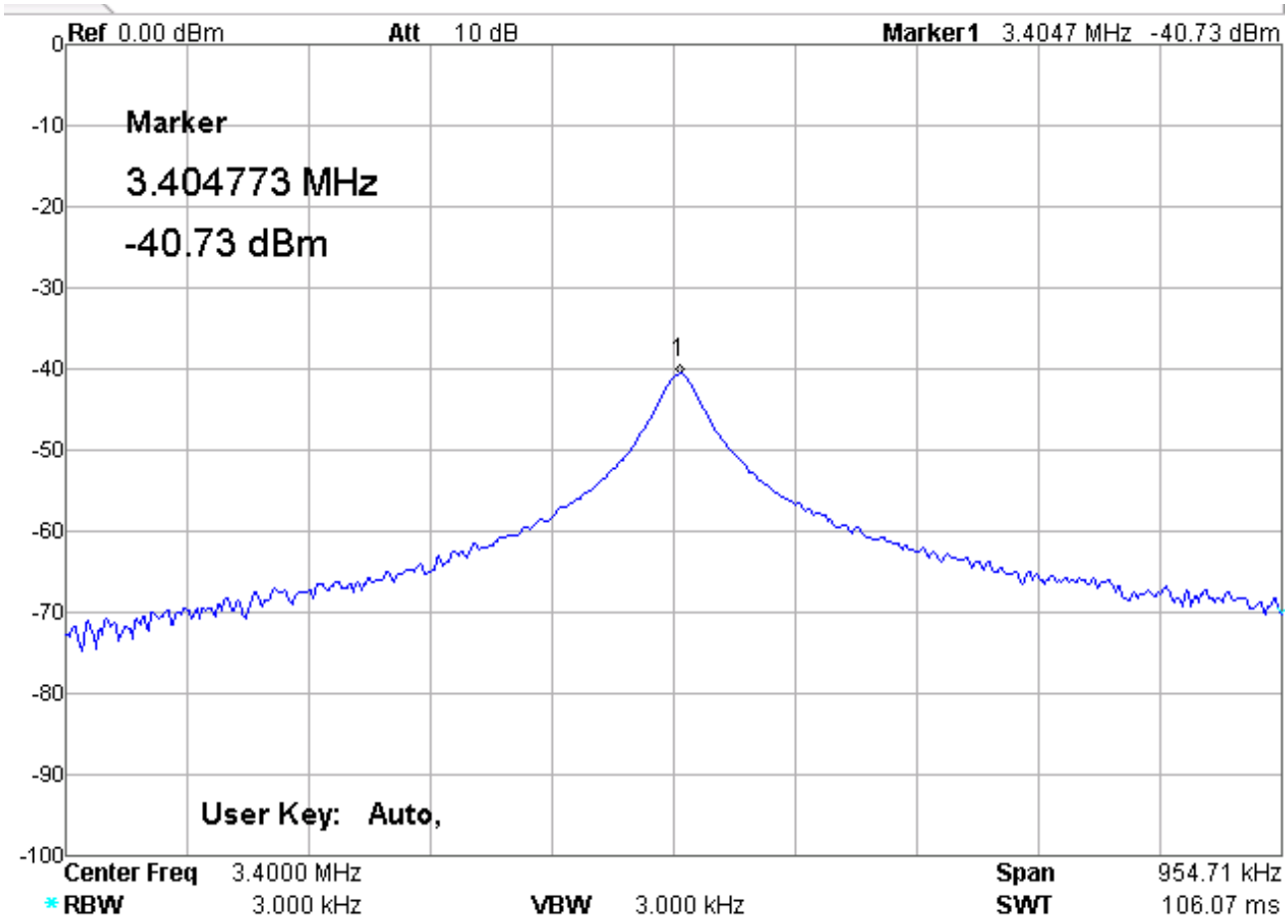
Zwei Transistoren + ein Arduino

Ein Direktmischer für das 80m-Band. Das Konzept(„Berta“) wollte ich modernisieren und ohne frei abstimmbaren LC-Oszillator aufbauen. Allerdings sollte auch eine gute Selektivität mit nur einem Kreis erreicht werden. Nun, das ist nur halbwegs gelungen. Denn es sind eigentlich zwei Schwingkreise, die hier ihren Dienst tun. Der resonante Schwingkreis vor dem Gate des ersten MOSFETs erhält auch einen Drehkondensator. Zusammen bilden die beiden ein Bandfilter. Der Serienkreis zieht die Resonanz des Parallelkreises mit. Man kann eine minimale Bandbreite von 20 kHz erzielen. Es ist eine relativ feste Kopplung, hilft aber die starken Träger des Rundfunks stark zu dämpfen. Für eine „hochohmige Antenne“(ein längerer Draht) ist ein kapazitiver Anschluss vorhanden.

Selektivität
Die beiden Kreise sind großzügig von 3,3 MHz bis über 4,5 MHz abstimmbar. Mit beiden Drehkos kann man ein optimales Ergebnis einstellen. Über das ganze Band gesehen, ist bei einer Fehlabstimmung die Dämpfung nicht zu hoch. Man hört die Stationen und versucht danach das Optimum zu finden. Zusätzlich ist im Sourcekreis des ersten MOSFETs ein Poti, mit dem man die Gesamtverstärkung regelt. Es kann vorkommen, dass die Schaltung schwingt. Dann muss man den Regler zurück drehen. Nur bei optimaler Abstimmung wird die Resonanz recht spitz. Ohne den Serienkreis an der Antenne wäre der Schwingkreis zu breit für einen störungsfreien Empfang. Der sehr kleine Kopplungskondensator(2pF) für Drahtantennen unterdrückt Einstreuungen des Netzbrumms. Ansonsten habe ich die Kurven an einer Quellenimpedanz von 50 Ohm aufgenommen. Der Empfänger hat daher eine BNC-Buchse.

Resonanz

Das Bild zeigt bei der niedrigen Frequenz folgende Kurve:



Die Bandbreite ist 27 kHz, die Güte Q = 126. Mehr war nicht zu erreichen.

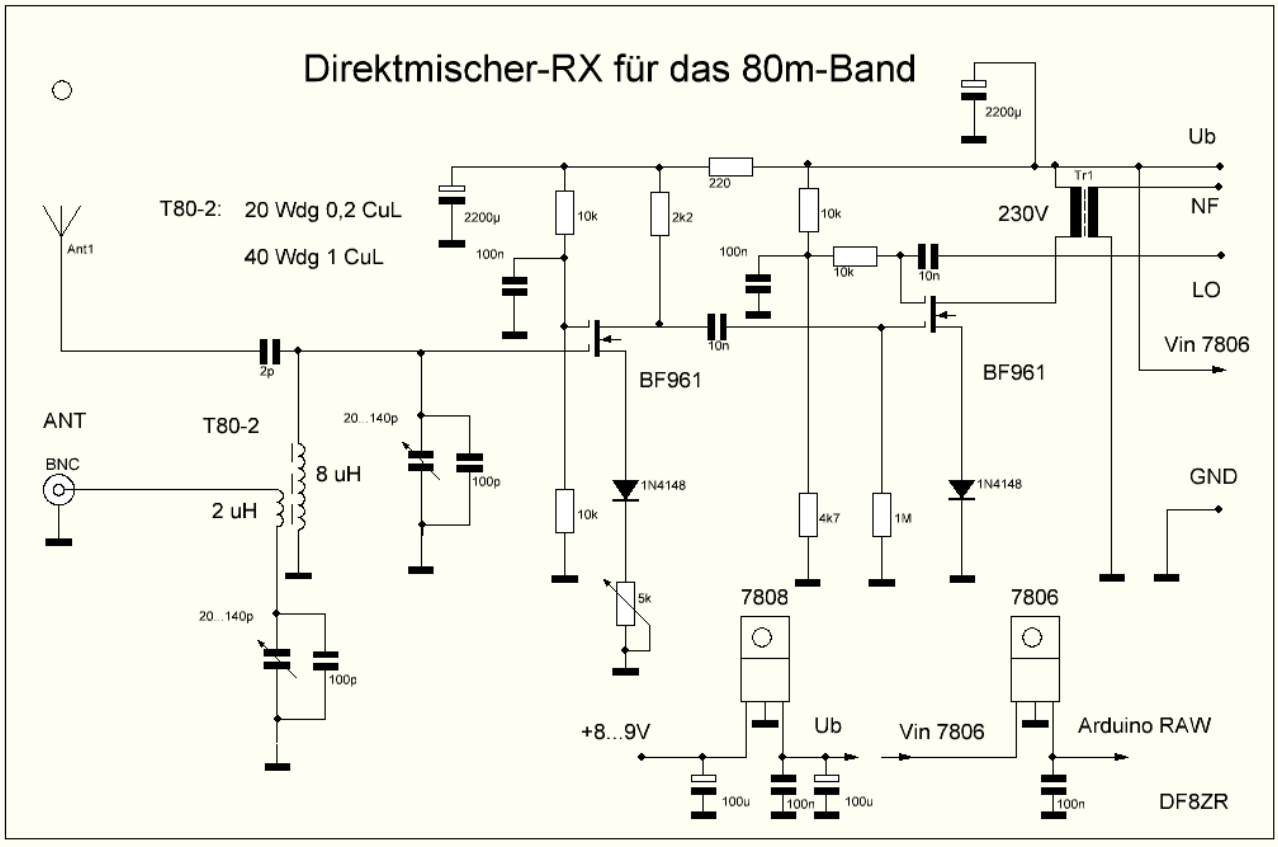
Wenn die Schaltung durch ungewollte Rückkopplung schwingt, ist der Empfang blockiert. Es gibt hier keine Entdämpfung wie bei einem Audion durch Zuführung von Energie. Da ein Mitziehen durch die Empfangsfrequenz wegen des unabhängigen Oszillators(Arduino-Generator) nicht möglich ist, würde es zu lästigen Pfeiftönen kommen. Bei -54 dBm Eingangsleistung wurden durch Resonanzüberhöhung -34 dBm am Gate gemessen.

Der Antennenkreis ist dabei nicht in Resonanz! Mit diesem Drehko regelt man die Ankopplung. Ist das C niedrig, wird sich

eine Dämpfung von -3 dB einstellen. Die Bandbreite verringert sich aber deutlich, das Signal wird „sauber“.

Der Mischer

Die Mischung erfolgt mit dem zweiten MOSFET. Er wird vom Generator mit 3Vss vollkommen durchgesteuert. Dieser Schaltbetrieb erfolgt mit Rechteckspannungen. Dem Si5351 wird dabei weniger als 2mA entnommen. Doch bitte nicht bei eingeschalteter Stromversorgung an dem Anschluss löten! Der Baustein reagiert empfindlich auf Kurzschlüsse, die durch einen geerdeten Lötkolben entstehen können.



Ich habe für die Wicklungen 0,5mm Schaltdraht genommen. Mit der Isolierung war der etwa 1mm dick. Für die Antennenwicklung nahm ich 0,2mm isolierten Wirewrapp-Draht und wickelte diesen ans kalte Ende. Falls der Draht zu dick sein sollte, kann man die Restlänge auch übereinander wickeln.

Siebung
Als ich den Arduino an die Stromversorgung mit +8V(Stabi 7808) anschloß, hörte man heftige Schwingungen. Erst durch den Anschluss einer 9V-Blockbatterie anstelle der gemeinsamen Versorgung blieben diese weg. Es gelang mir nicht in die innere Schaltung des Arduino-Boards zu blicken. Das Problem der internen Stromverteilung lässt sich ohne weitgehendes Studium der verfügbaren Hinweise im Netz kaum lösen. Daher versuchte ich es mit einer aufwendigen Siebung, wie man dem Schaltbild entnehmen kann. Erst danach war Ruhe im Gerät. Die Batterie konnte ich zwischenzeitlich mit ähnlichem Erfolg durch ein zweites Labornetzgerät ersetzen. Dennoch musste ich daran eine Mindestspannung von +5 V einstellen. War sie höher als 7V, kamen wieder Schwingungen auf. Es muss etwas mit der Stromversorgung auf dem Board zu tun haben. Jedenfalls kann ich die Schaltung empfehlen, die ich hier dokumentiert habe.

Der Arduino zieht allein > 45 mA! Der eigentliche RX nur etwas 2...4 mA. Man kann das Radio aber auch mit 12V...15V betreiben.

Der Arduino
Die Software nahm ich vom Netz. Dem Urheber Julio Cesar sei gedankt. Für die bequeme Bedienung habe ich einige Ergänzungen eingearbeitet. Man kann die Frequenzsteps mit dem Druck auf den Impulsgeber verändern oder auch einfach auf die Tasten klicken. Von 10 Hz bis 100kHz sind sie zusätzlich vorgesehen. Manchmal möchte man schnell den Frequenzbereich wechseln. Mit dem Fortschreiten über den Impulsgeber ist das etwas umständlich. Mit den Tasten geht das direkt und sofort kann man mit 100Hz-Schritten für eine klare Verständlichkeit fein abstimmen.

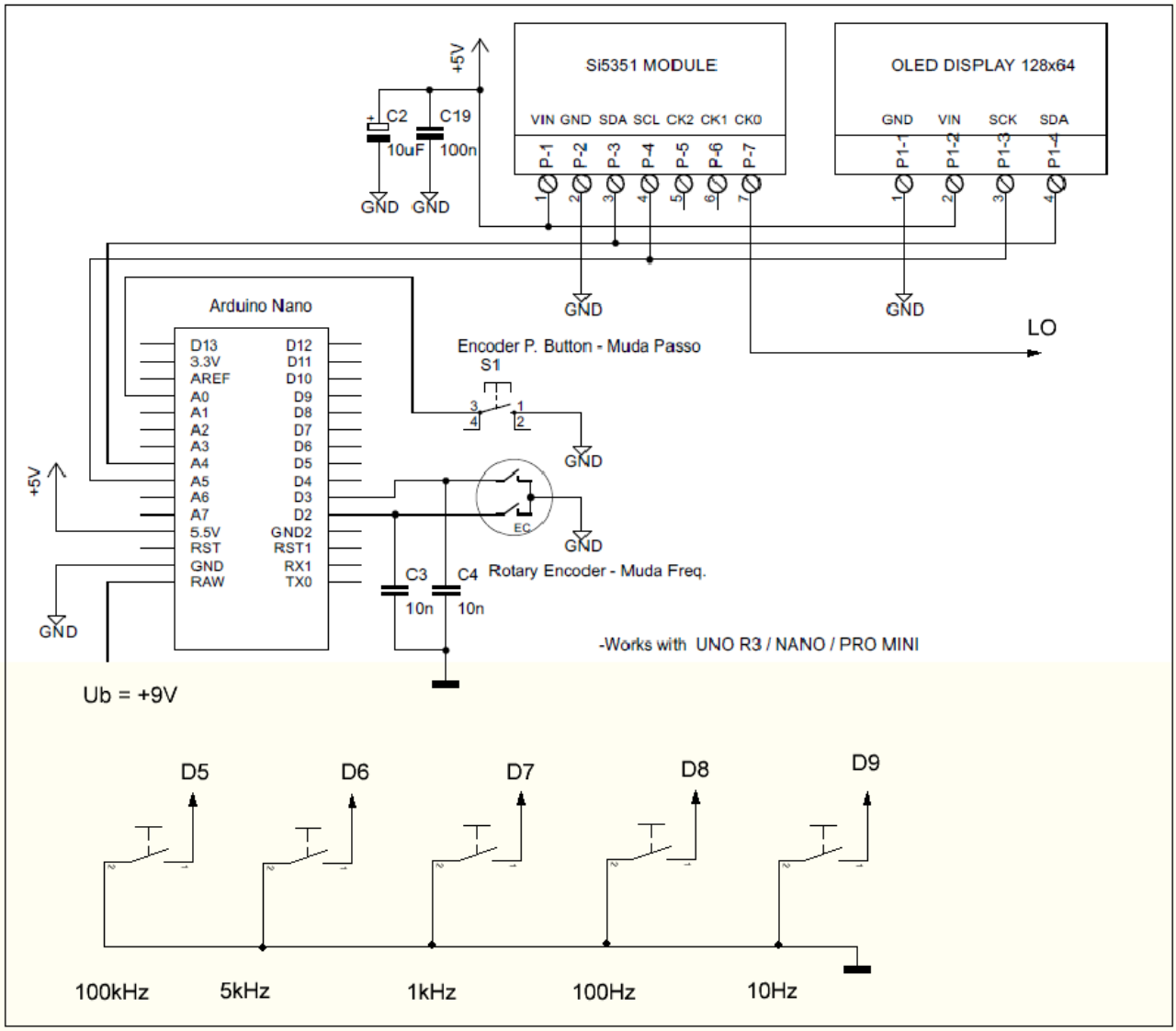
Was brauchen wir noch?
Einen Arduino-Nano. Einen LO-Baustein Si5351 und ein Oled. Die Teile sind sehr preiswert aus China zu beziehen. In DL zahlt man mehr als das Doppelte. Dazu noch die 5 Drucktaster und eine

Kopfhörerbuchse. Einen Lautsprecherverstärker habe ich nicht vorgesehen. Man kann aber einen LM386 nachrüsten.

Das Schema und die Software sind im Anhang.

Bei YouTube ist hier ein Film:
https://www.youtube.com/watch?v=n_-xWK8g8co

DF8ZR; im Januar 2022



Sketch:

Den Sketch bitte mit einem Editor in die Entwicklungssoftware vom Arduino-Nano kopieren.

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

//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  0                //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.
#define FREQ_INIT  3500000   //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 ZEHN_PIN 9           //D9 Tasten für Steps
#define HUNDERT_PIN 8        //D5
#define KILO_PIN 7           //D6
#define FUENFK_PIN 6         //D7
#define HUNDERTK_PIN 5       //D8
//--------------------------------------------------------------------------
----------------------------

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;
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

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


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


void setup() {


  /*Serial.begin(9600); // Öffnet die serielle Schnittstelle bei 9600
Bit/s:
  Serial.print("BEGINN");
  Serial.print("\t");
  */
```

```
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);


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

  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);

  pinMode(st5k, INPUT_PULLUP);///

  statup_text();

  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, cal);
  si5351.output_enable(SI5351_CLK0, 1);                 //1 - Enable /
0 - Disable CLK
  si5351.output_enable(SI5351_CLK1, 0  );
  si5351.output_enable(SI5351_CLK2, 1);                 //aktiviert

  si5351.drive_strength(SI5351_CLK0,
```

```
SI5351_DRIVE_2MA);  //Output current 2MA, 4MA, 6MA or
8MA

  si5351.drive_strength(SI5351_CLK2,
SI5351_DRIVE_2MA);   // 2 mA for HB mixers///eingefügt für
Superhetbetrieb


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

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

void loop() {

 if (tastenklick == 0){
  if (freqold != freq) {
    time_now = millis();
    tunegen();
    freqold = freq;
    }


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

 }
```

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

 zehn = 1;
 tastenklick=1;

 tasten();

}

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

 hundert = 1;
 tastenklick=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;

 st5k = 1;
 tasten();
```

```
}

if (digitalRead(HUNDERTK_PIN) == LOW) {
    time_now = (millis() + 300);
tastenklick=1;
 hundertk = 1;
 tasten();

}

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

}

void tunegen() {
  si5351.set_freq_manual((freq + (interfreq * 1000ULL)) *
100ULL, pll_freq, SI5351_CLK0);
}

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 tasten(){

if (zehn == 1) {

stp = 2;
      fstep = 10; // statt 10kHz-steps werden 5kHZ-steps eingestellt!
}


if (st5k == 1) {

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


if (hundert == 1) {

stp = 3;
      fstep = 100;
}

if (eink == 1) {

stp = 4;
      fstep = 1000;
```

```
}

if (hundertk == 1) {

stp = 6;
      fstep = 100000;
}

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

tunegen();
freqold = freq;

void layout();

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

}

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("IF:");
  //display.print(interfreq);
  display.print("80m-Band");
  //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("L O");
  display.display();
}

void statup_text() {
  display.setTextSize(1);
  display.setCursor(4, 5);
  display.print("Si5351");
  display.setCursor(4, 20);
  display.print(" CW/SSB-Empfaenger ");
  display.setCursor(4, 35);
  display.print("Vers. DF8ZR");
  display.setCursor(4, 50);
  display.print(">> 80m-Band <<");
  display.display();
  delay(3500);//3000
  display.clearDisplay();
}
```