# *60W 144MHz FM Power Amplifier*

by Simon Dorrer, OE3SDE

May 18, 2021

**Abstract**

This 60W 144MHz FM Power Amplifier is the perfect "Add-On" for my previously released 1W 144MHz FM Transceiver. This power amplifier adds 30dB gain in the TX-Path and a lot of important analog RF filtering. Furthermore, the RX-Path is amplified with a gain of 23dB and is filtered with high order filters to improve the signal noise ration for better understanding of the other transmitters. All RX / TX signals are switched with PIN diodes to avoid expensive radio frequency shielded relais.

# Contents

# Chapter 1

# Overview

## 1.1 Description

On the following pages the most important details for the 60W 144MHz FM power amplifier are shown. This document includes circuits, PCB-Designs, bill of material, microcontroller codes, measurements and final result pictures.

## 1.2 Software

The theoretical work like circuit simulation, PCB Design and Microcontroller (MCU) programming was done with following software:

- Multisim & LTSpice: Circuit Simulation
- ELSIE: Analog Filter Design
- Autodesk EAGLE: Circuit- & PCB-Design
- Autodesk Fusion 360: 3D Design
- Arduino: MCU Code

## 1.3 Devices

The practical work like PCB assembly, fault finding and measurements are done with the following devices:

- FLUKE 179 True RMS Multimeter
- Weller WT2020M Soldering Station
- Zhongdi ZD-939L Hot Air Rework Station
- Andonstar AD407 Digital Microscope
- Siglent SDS2352X-E Oscilloscope (2xCH, 350MHz)
- Siglent SDG2042X Function Generator (2xCH, 120MHz - hacked)
- Siglent SVA1015X Spektrum & Vector Network Analyser (1.5GHz)
- Siglent SPD3303X-E Power Supply (2x0-32V with 3.2A, 1x2.5V/3.3V/5V with 3.2A)
- QJ3005EIII 300W Power Supply (2x0-30V with 5A, 1x5V with 3A)
- Anycubic i3 Mega S 3D Printer

A complete list of all my labor equipment can be found here: `https://www.oe3sde.com/Workstation.html`

# Chapter 2

# Circuit

## 2.1 Filters

In ham radio applications it is very important, that transmitted signals are only in the licensed band. This power amplifier works for the 2m band, that means a frequency range of 144MHz to 146MHz. Of course, also the received signals should be filtered to avoid any unwanted noise.

### 2.1.1 Air Coil Calculation

The air coils for the TX-OUT and RX-IN filters according to circuit 2.3 and 2.5 can be calculated with the following formula:

$$L = \frac{N^2 \cdot d^2}{18 \cdot d + 40 \cdot l} \tag{2.1}$$

Where:

- L = Inductance
- N = Number of turns
- d = Coil outside diameter
- l = Length of the coil

For a faster result, also an online calculator[1] [1] can be used.

[1]https://m0ukd.com/calculators/air-cored-inductor-calculator/

### 2.1.2 TX-IN Filter

This filter is build up with SMT components because of the low input power of 0.5W. The TX-IN filter is a seventh order chebyshev low pass filter with a cutoff frequency of 175MHz. The high order is chosen because of the high harmonic distortion of the DRA818V **TRX!** at the input.

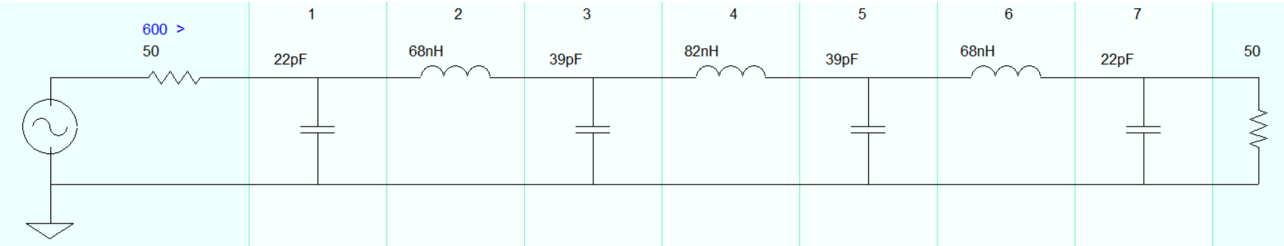



Figure 2.1: Circuit of the TX-IN Filter

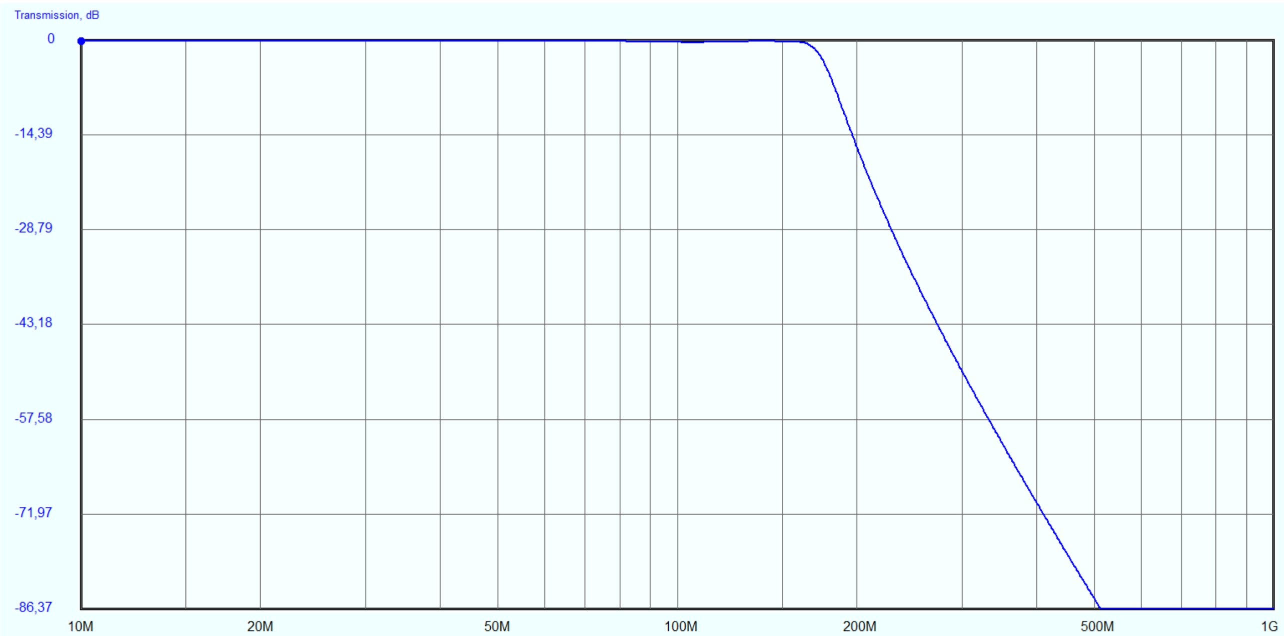



Figure 2.2: Transfer Function of the TX-IN Filter

### 2.1.3 TX-OUT Filter

This filter is build up with self-wound inductors because of the higher output power of 60W. The TX-OUT filter is a fifth order chebyshev low pass filter with a cutoff frequency of 182MHz.

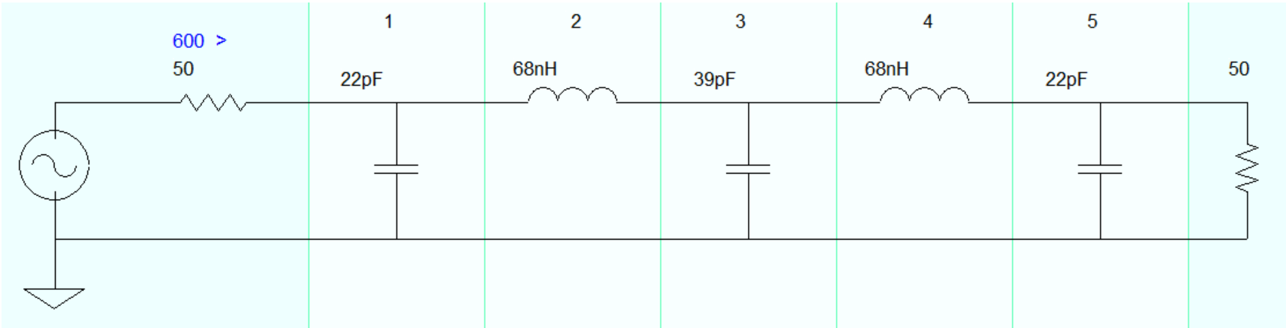



Figure 2.3: Circuit of the TX-OUT Filter

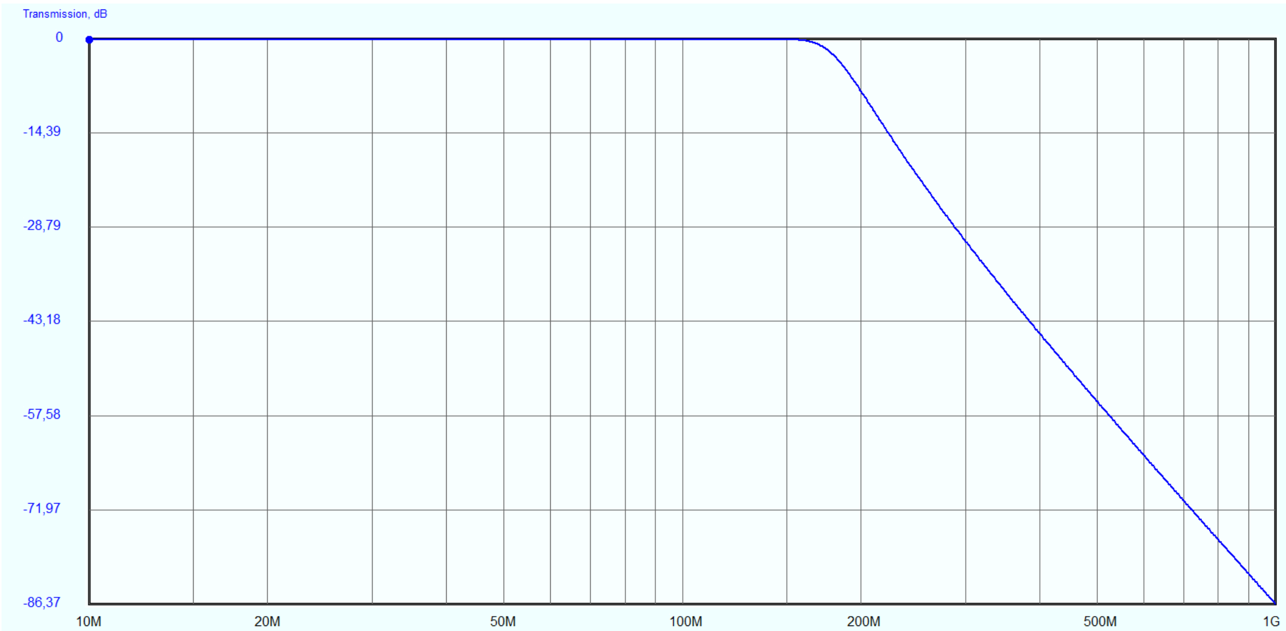



Figure 2.4: Transfer Function of the TX-OUT Filter

### 2.1.4 RX-IN Filter

This filter can be build up with SMT components. However, self-wound inductors can be changed easier and the filter band can be adjusted slightly. The RX-IN filter is a third order chebyshev band pass filter with a lower cutoff frequency of 120MHz and a upper cutoff frequency of 172MHz.

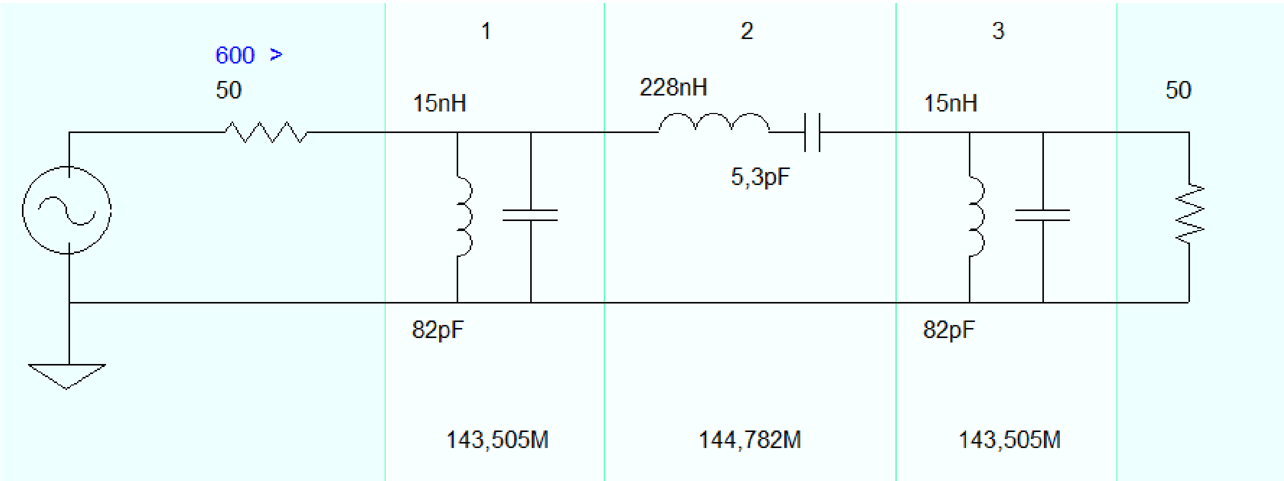



Figure 2.5: Circuit of the RX-IN Filter

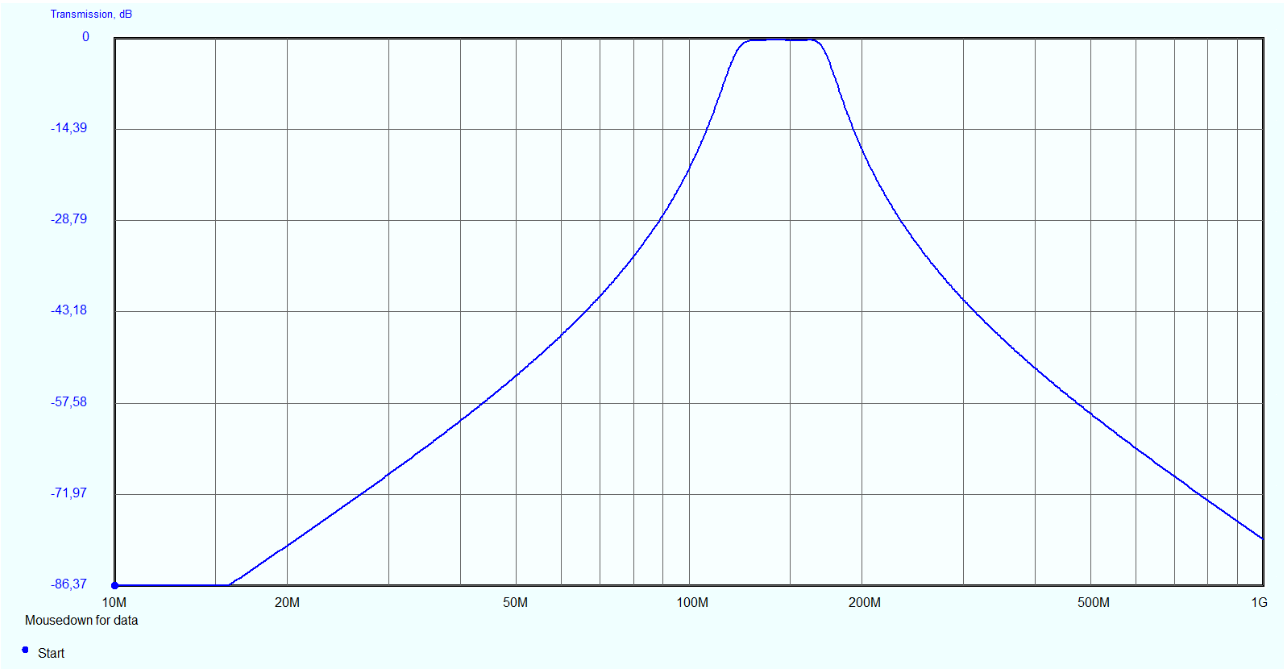



Figure 2.6: Transfer Function of the RX-IN Filter

## 2.2 PIN-Diode Switches

In this project, PIN-Diodes are used instead of relais to switch between RX and TX path.

### 2.2.1 RX/TX Input-Switch

ToDo Text

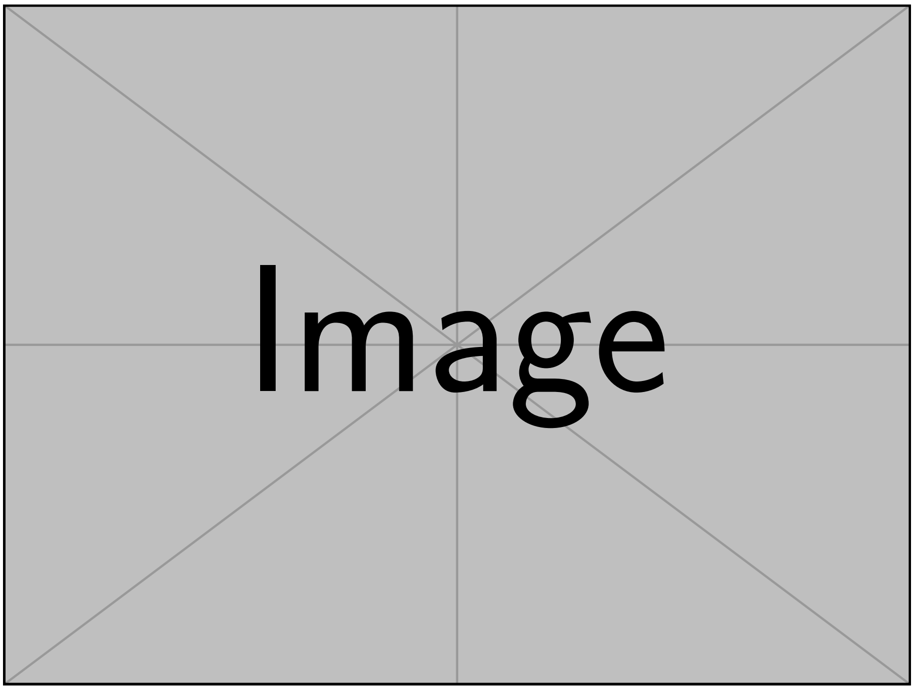



Figure 2.7: Circuit of the RX/TX Input-Switch

### 2.2.2 RX/TX Output-Switch

ToDo Text

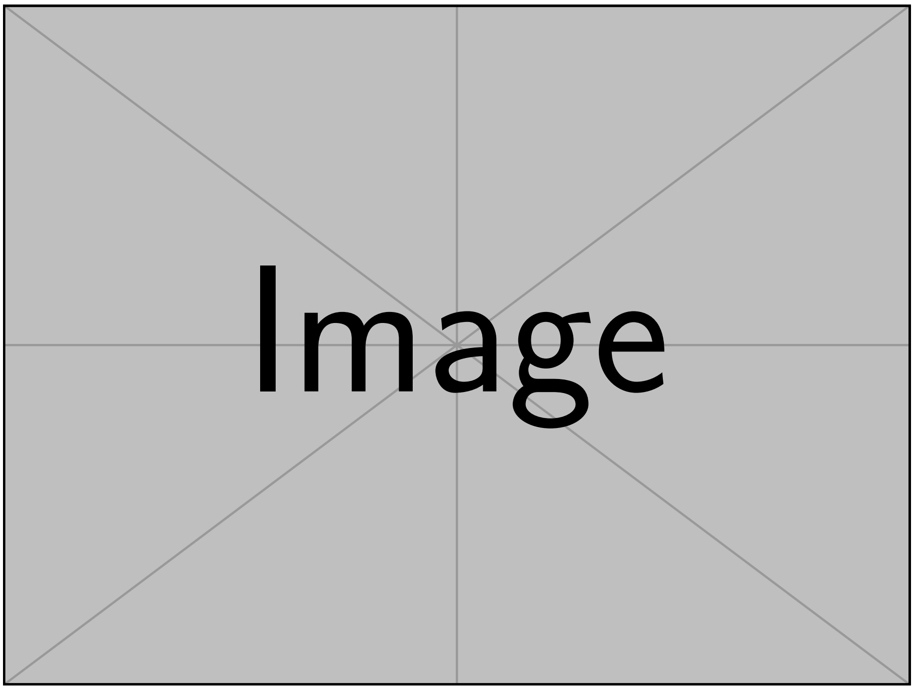
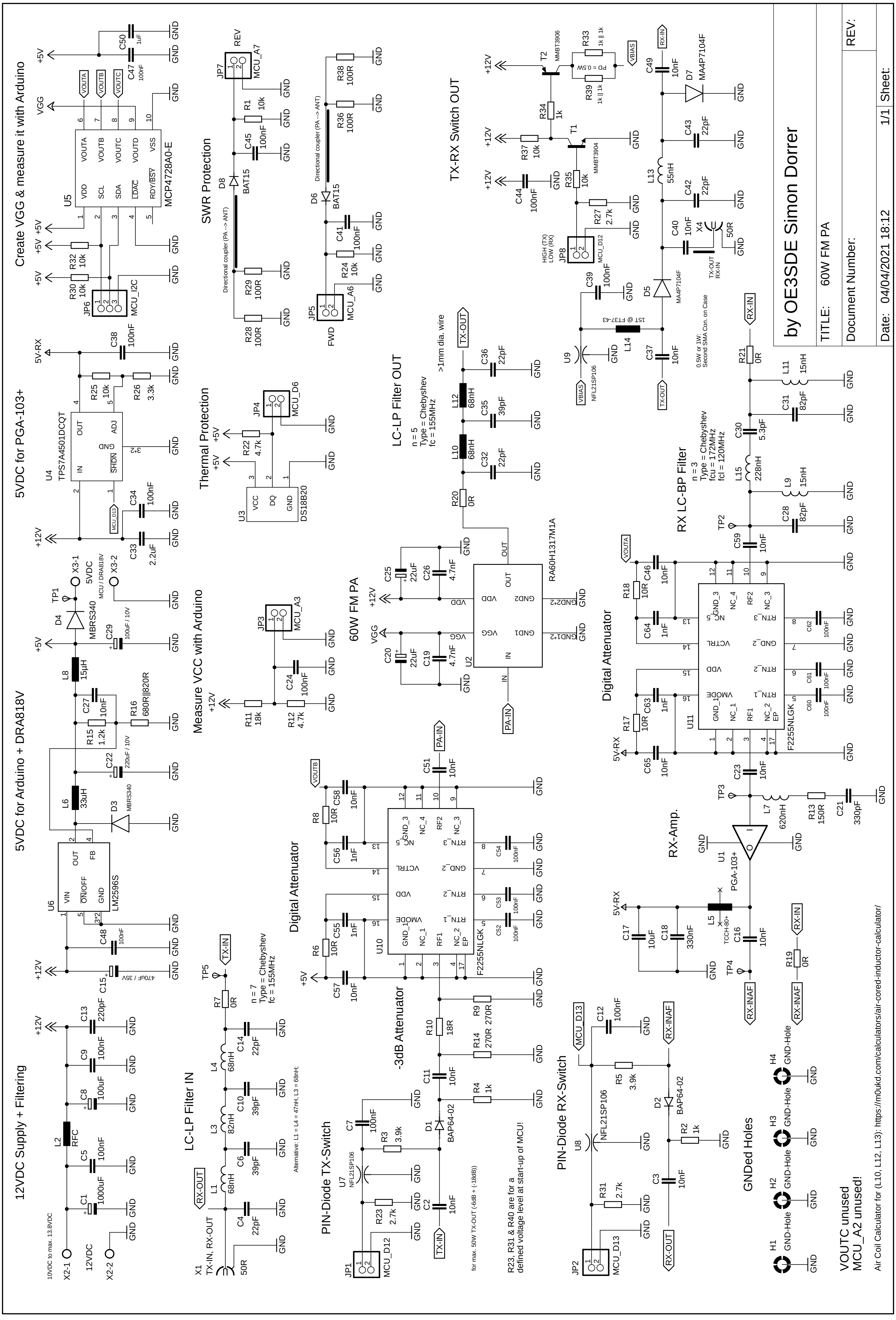



Figure 2.8: Circuit of the RX/TX Output-Switch



12VDC Supply + Filtering
5VDC for Arduino + DRA818V
5VDC for PGA-103+
Create VGG & measure it with Arduino
LC-LP Filter IN
PIN-Diode TX-Switch
-3dB Attenuator
Digital Attenuator
Measure VCC with Arduino
Thermal Protection
SWR Protection
60W FM PA
LC-LP Filter OUT
TX-RX Switch OUT
PIN-Diode RX-Switch
RX-Amp.
Digital Attenuator
RX LC-BP Filter
GNDed Holes
VOUTC unused
MCU_A2 unused!
Air Coil Calculator for (L10, L12, L13): https://m0ukd.com/calculators/air-cored-inductor-calculator/
by OE3SDE Simon Dorrer
TITLE: 60W FM PA
Document Number:
REV:
Date: 04/04/2021 18:12
1/1 Sheet:

# Chapter 3

# PCB

## 3.1 Layer Stackup

This **PCB!** is a two layer PCB. On the top layer the **RF!** signals and SMT components are placed. The bottom layer is a completed GND layer with some supply traces. This PCB is produced by my sponsor JLCPCB [1]. Normally, for bigger RF PCB-Designs a 4 layer PCB is used as follows:

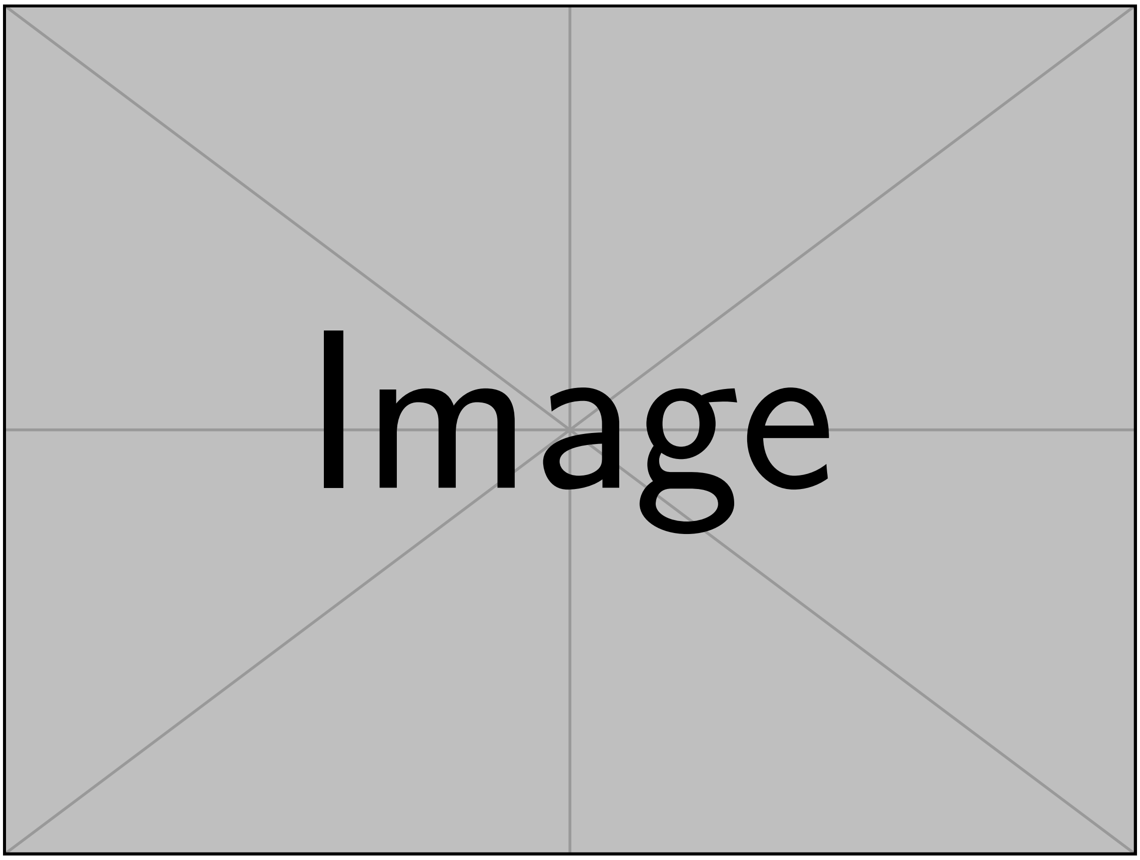



Figure 3.1: Two Layer PCB Stackup

[1] `https://jlcpcb.com/`

## 3.2 Coplanar Waveguide Grounded (CPWG)

In RF PCB-Design applications it is very important, that the trace impedance is matched to 50Ω. Therefore, different techniques like Microstrip, Stripline or Coplanar Waveguide are used. This PCB-Design uses the Coplanar Waveguide method. However, the calculation of this technique is not that easy (elliptical integrals etc.). That is why an online calculator [2] [2] is used. The original equations are in [3] on page 79.

### 3.2.1 RX CPWG Calculation

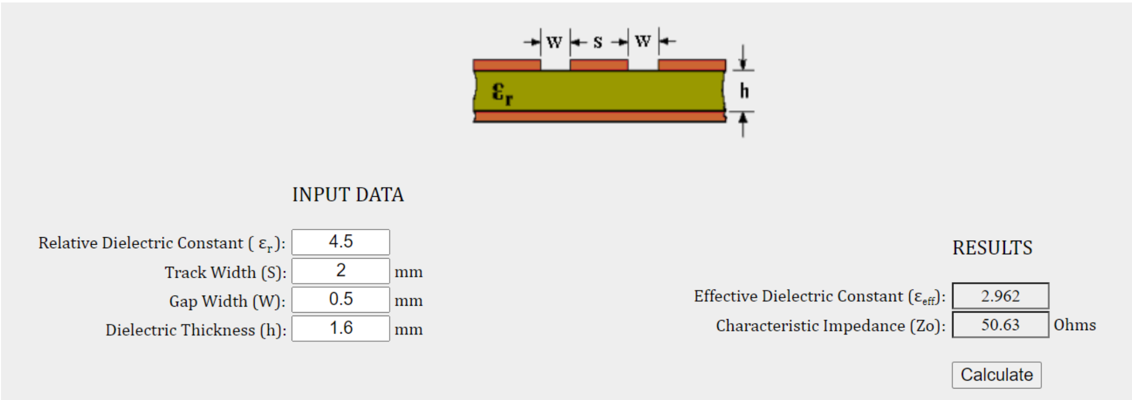



Figure 3.2: RX CPWG Calculation

### 3.2.2 TX CPWG Calculation

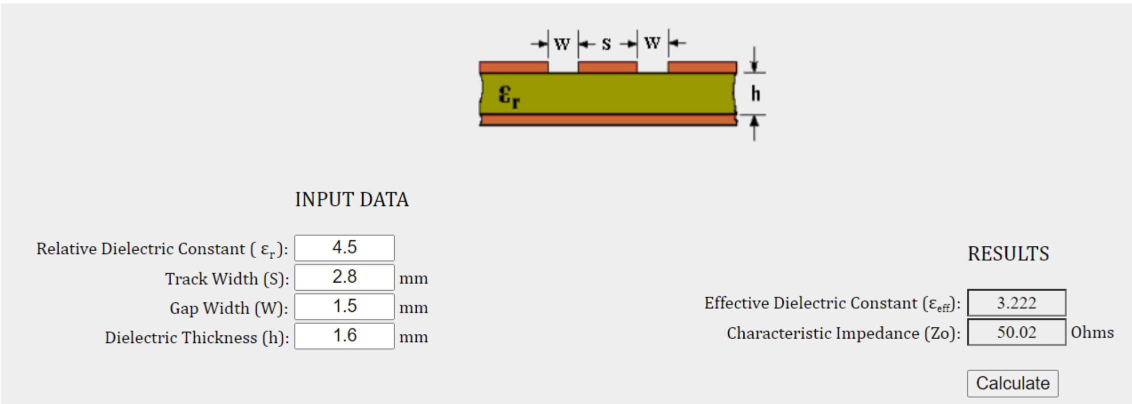



Figure 3.3: TX CPWG Calculation

[2] https://chemandy.com/calculators/coplanar-waveguide-with-ground-calculator.htm

## 3.3 Trace corners

ToDo Text

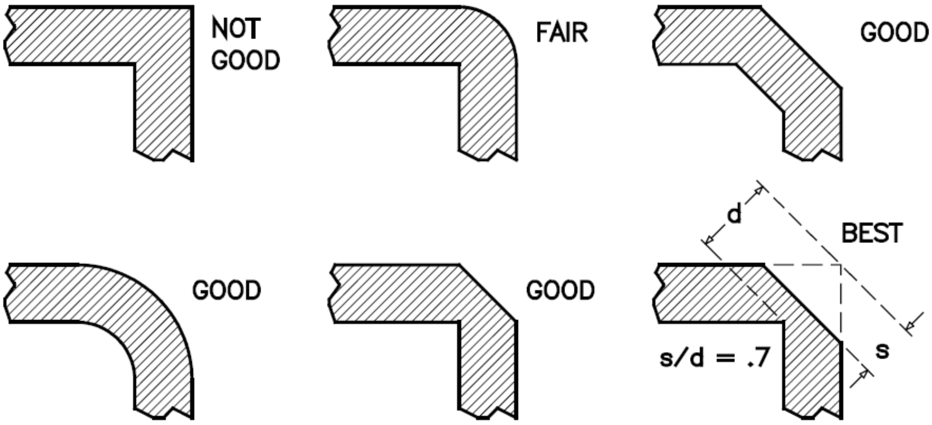



Figure 3.4: Corners in RF traces

## 3.4 3D PCB

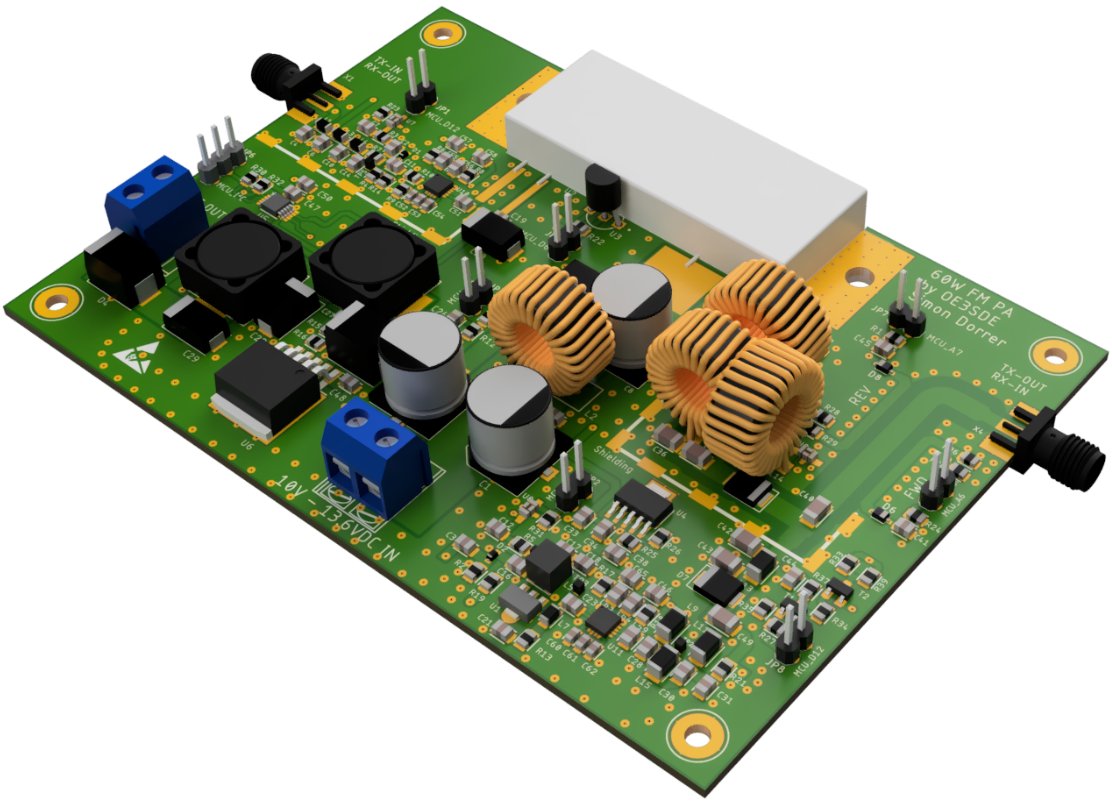



Figure 3.5: Finished **PCB!** 3D view

## 3.5 2D PCB

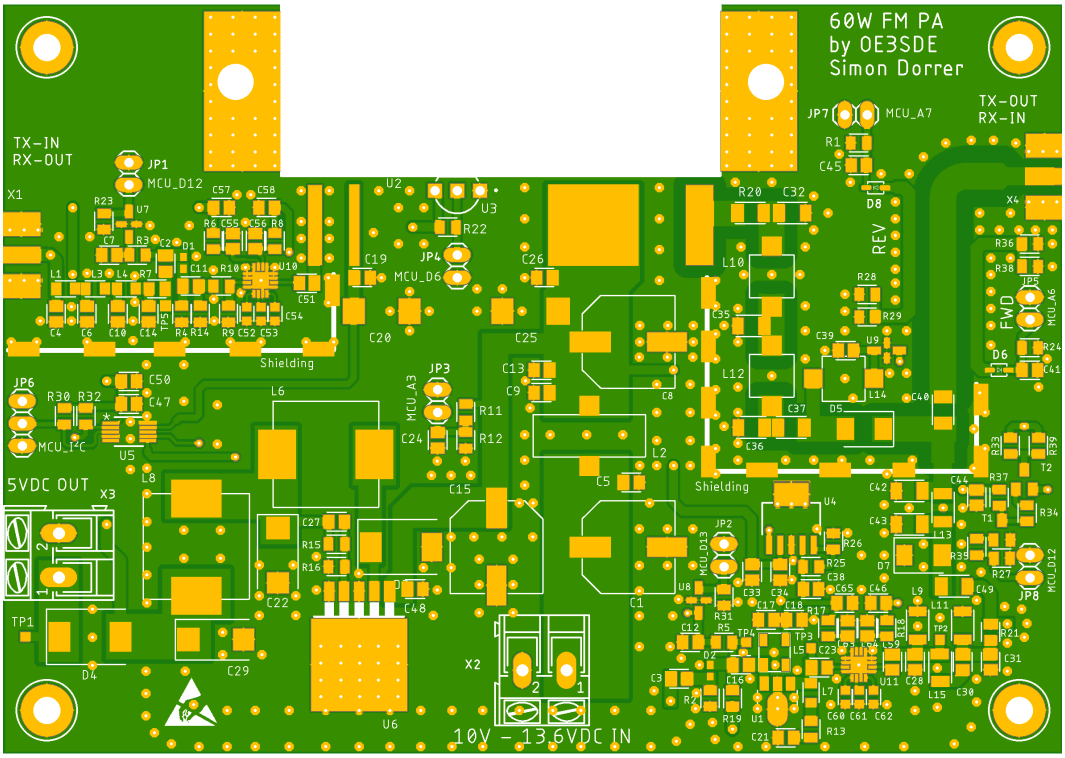



Figure 3.6: Finished **PCB!** 2D top view

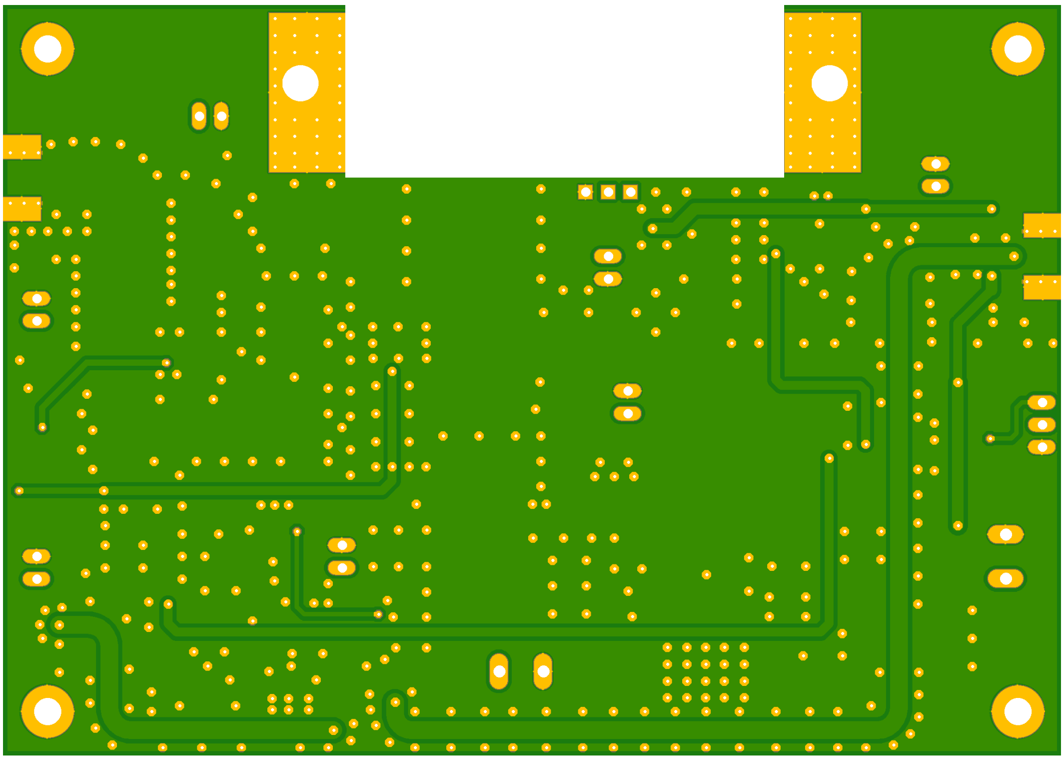


Figure 3.7: Finished **PCB!** 2D bottom view

# Chapter 4

# Bill of Material

| Qty | Value | Parts |
|---|---|---|
| 4 | 0R | R7, R19, R20, R21 |
| 1 | 1.2k | R15 |
| 1 | 1000uF | C1 |
| 4 | 100R | R28, R29, R36, R38 |
| 6 | 100nF | C52, C53, C54, C60, C61, C62 |
| 13 | 100nF | C5, C7, C9, C12, C24, C34, C38, C39, C41, C44, C45, C47, C48 |
| 1 | 100uF | C8 |
| 1 | 100uF / 10V | C29 |
| 4 | 10R | R6, R8, R17, R18 |
| 7 | 10k | R1, R24, R25, R30, R32, R35, R37 |
| 12 | 10nF | C2, C3, C11, C16, C23, C27, C46, C51, C57, C58, C59, C65 |
| 3 | 10nF | C37, C40, C49 |
| 1 | 10uF | C17 |
| 1 | 12VDC | X2 |
| 1 | 150R | R13 |
| 1 | 15T @ FT37-43 | L14 |
| 2 | 15nH | L9, L11 |
| 1 | 15ÁH | L8 |
| 1 | 18R | R10 |
| 1 | 18k | R11 |
| 3 | 1k | R2, R4, R34 |
| 2 | 1k || 1k | R33, R39 |
| 4 | 1nF | C55, C56, C63, C64 |
| 1 | 1uF | C50 |
| 1 | 2.2uF | C33 |
| 3 | 2.7k | R23, R27, R31 |
| 1 | 220pF | C13 |
| 1 | 220uF / 10V | C22 |
| 1 | 228nH | L15 |
| 2 | 22pF | C4, C14 |
| 4 | 22pF | C32, C36, C42, C43 |
| 2 | 22uF | C20, C25 |
| 2 | 270R | R9, R14 |

| Qty | Value | Parts |
|---|---|---|
| 1 | 3.3k | R26 |
| 2 | 3.9k | R3, R5 |
| 1 | 330nF | C18 |
| 1 | 330pF | C21 |
| 1 | 33uH | L6 |
| 2 | 39pF | C6, C10 |
| 1 | 39pF | C35 |
| 2 | 4.7k | R12, R22 |
| 2 | 4.7nF | C19, C26 |
| 1 | 470uF / 35V | C15 |
| 1 | 5.3pF | C30 |
| 1 | 50R | X4 |
| 1 | 55nH | L13 |
| 1 | 5VDC | X3 |
| 1 | 620nH | L7 |
| 1 | 680R\|\|820R | R16 |
| 2 | 68nH | L10, L12 |
| 2 | 68nH | L1, L4 |
| 1 | 82nH | L3 |
| 2 | 82pF | C28, C31 |
| 2 | BAP64-02 | D1, D2 |
| 2 | BAT15 | D6, D8 |
| 1 | DS18B20 | U3 |
| 2 | F2255NLGK | U10, U11 |
| 4 | GND-Hole | H1, H2, H3, H4 |
| 1 | LM2596S | U6 |
| 2 | MA4P7104F | D5, D7 |
| 2 | MBRS340 | D3, D4 |
| 1 | MCP4728A0-E | U5 |
| 1 | MMBT3904 | T1 |
| 1 | MMBT3906 | T2 |
| 3 | NFL21SP106 | U7, U8, U9 |
| 1 | PGA-103+ | U1 |
| 1 | RA60H1317M1A | U2 |
| 1 | RFC | L2 |
| 1 | TCCH-80+ | L5 |
| 1 | TPS7A4501DCQT | U4 |
| 5 | TPSTP10SQ | TP1, TP2, TP3, TP4, TP5 |
| 1 | TX-IN, RX-OUT | X1 |

Table 4.1: Bill of Material

# Chapter 5

# Measurements

## 5.1 TX-Path Transfer Function

ToDo Text (Variable Gain with Vdd of RA60, XY V for every gain)

### 5.1.1 TF of RA60H1317M1A

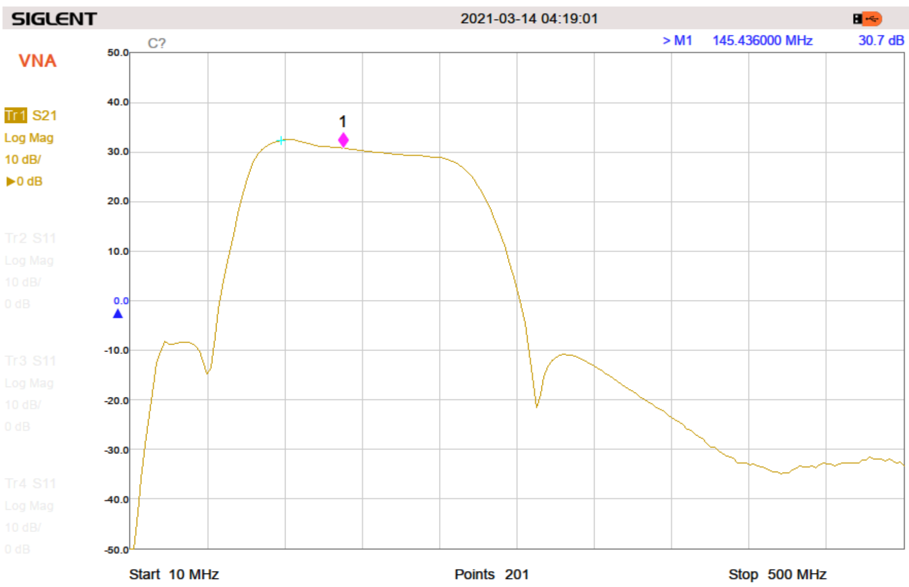



Figure 5.1: Transfer Function of RA60H1317M1A

### 5.1.2 10dB Gain

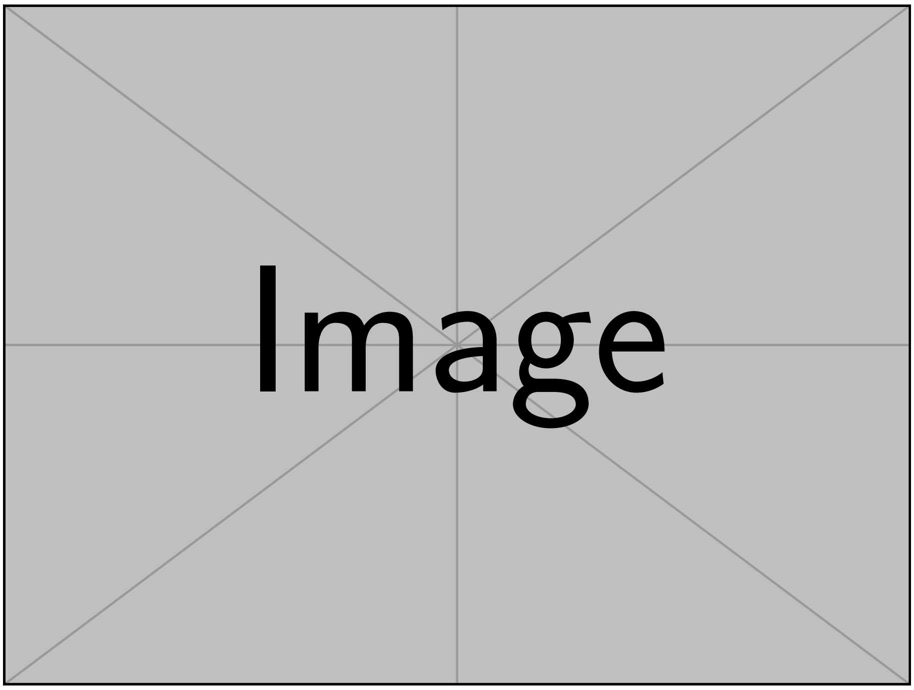



Figure 5.2: TX-Path Transfer Function with 10dB gain

### 5.1.3 30dB Gain

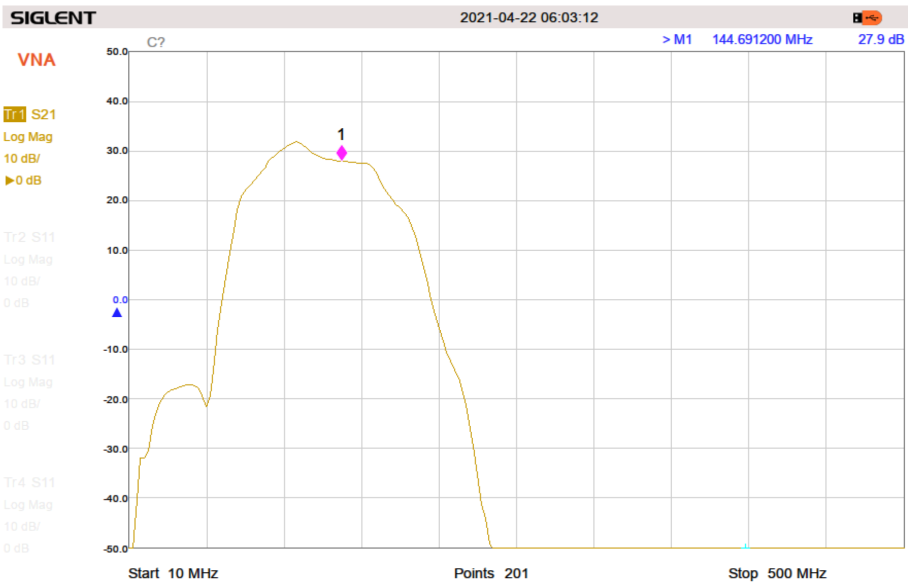



Figure 5.3: TX-Path Transfer Function with 30dB gain

## 5.2 RX-Path Transfer Function

ToDo Text (Variable Attenuation with F2255)

### 5.2.1 F2255NLGK Attenuation Diagram

ToDo Text (What is a F2255)

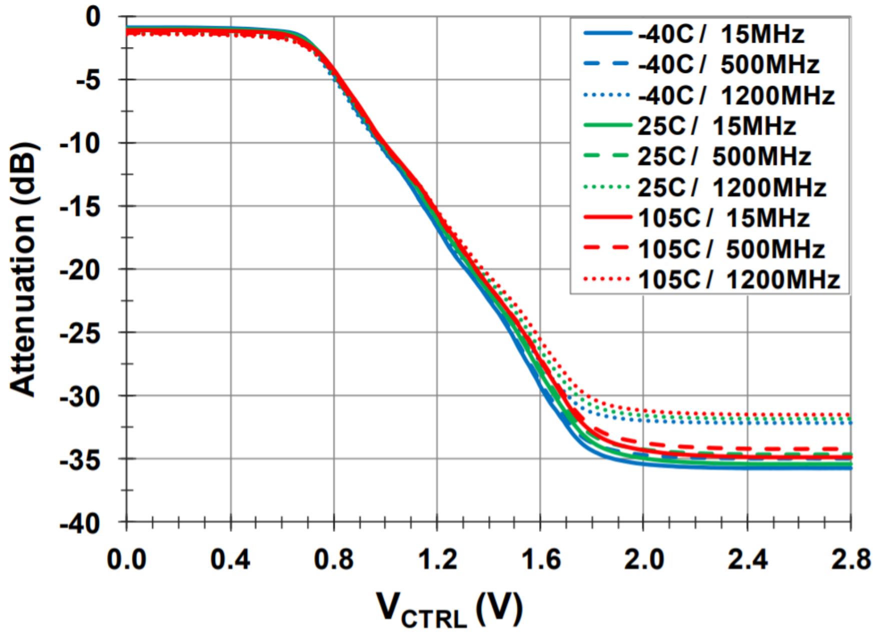



Figure 5.4: F2255NLGK Attenuation Diagram[1][4]

[1] https://www.mouser.at/datasheet/2/698/IDT_F2255_Datasheet_DST_20180209-1997542.pdf

### 5.2.2 No Attenuation

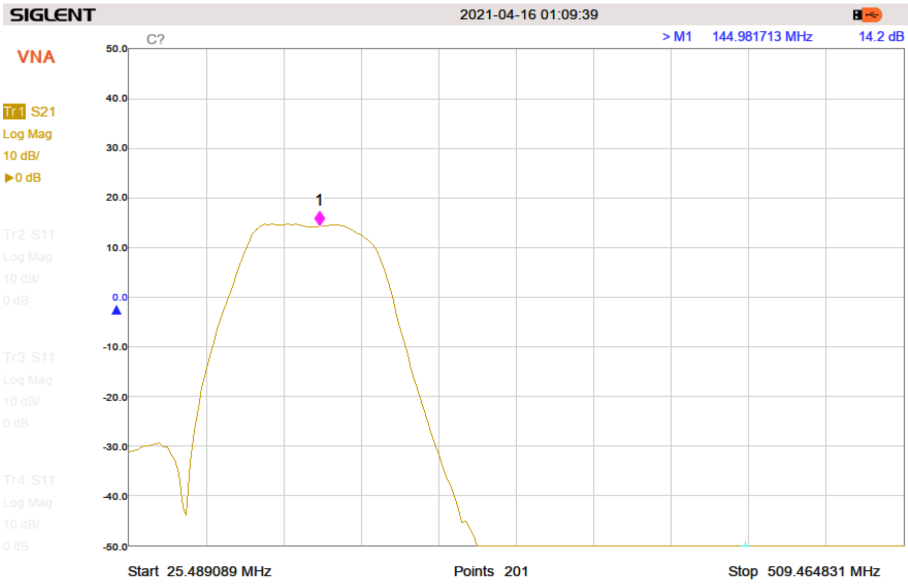



Figure 5.5: RX-Path Transfer Function without attenuation

### 5.2.3 20dB Attenuation

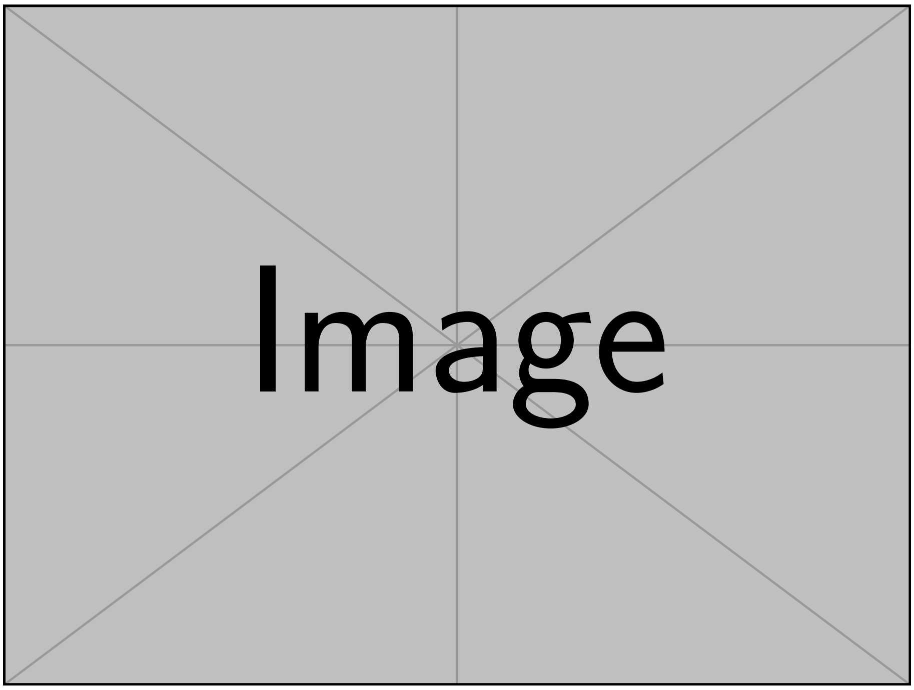



Figure 5.6: RX-Path Transfer Function with 20dB attenuation

## 5.3 Output Power

For all output power measurements an input power of XYdBm (xyW) is used. Furthermore, at the input of the Vector Network Analyzer a 23dBm attenuator is mounted to handle the power. So, the final output Power consists of the measured power and the 23dBm.
For exmaple, if the spectrum analyser shows an output power of XYdBm the output power equals $XY + 23dBm = dBm(XYW)$.

### 5.3.1 20W Output Power

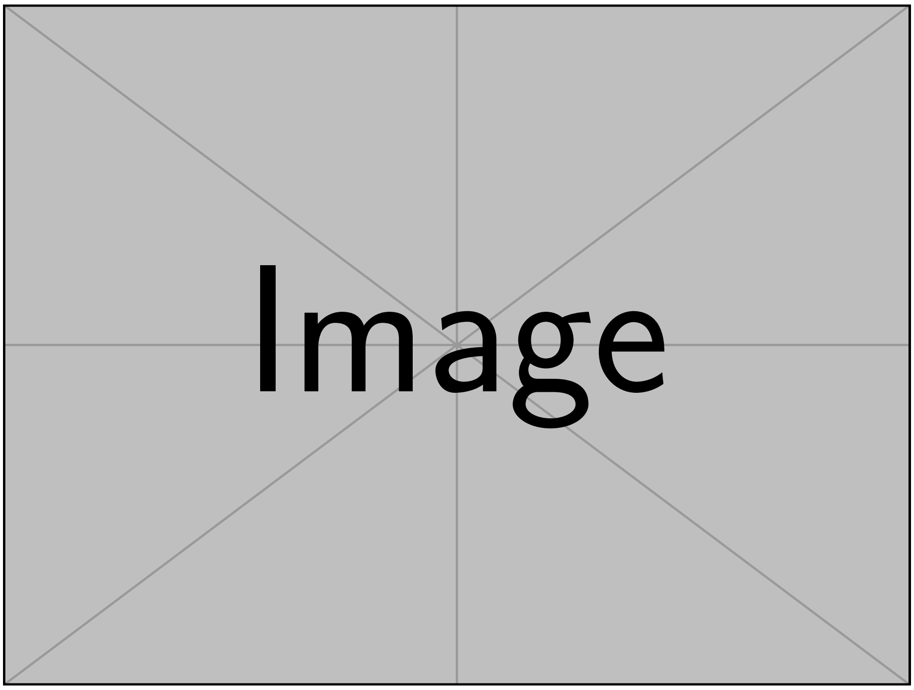



Figure 5.7: 20W Output Power

### 5.3.2 40W Output Power

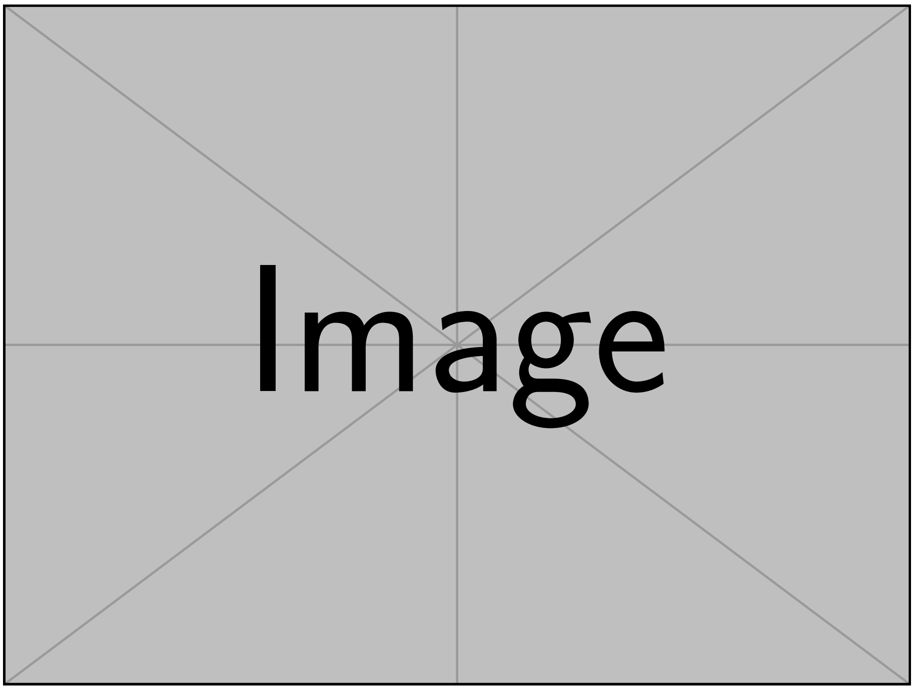



Figure 5.8: 40W Output Power

### 5.3.3 60W Output Power

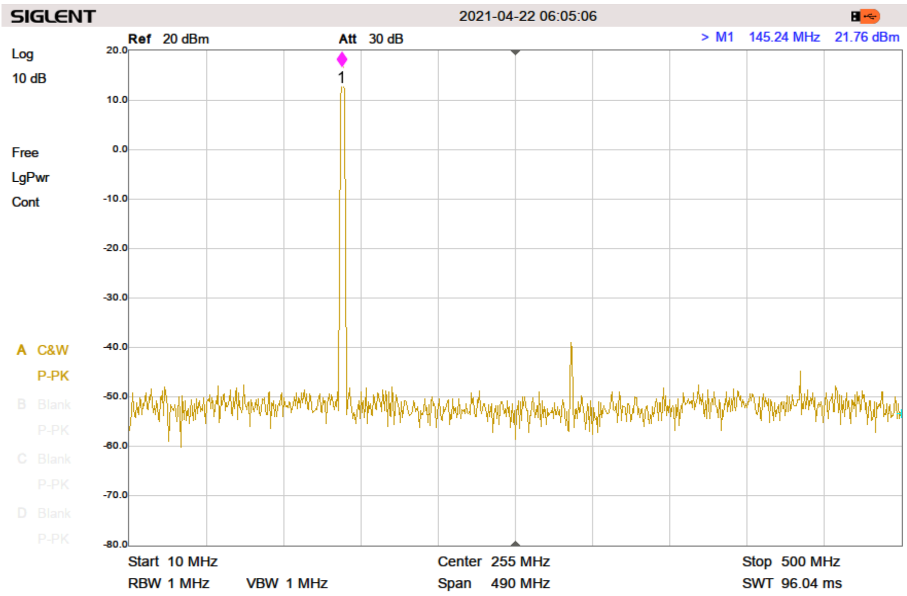



Figure 5.9: 60W Output Power

# Chapter 6

# Microcontroller Code

In this chapter, the MCU Code written with Arduino IDE is implemented. The internal hardware of the used MCU (ATmega328) is programmed on register level. To control the OLED display, the external DAC and some sensors, external libraries are used.

```
//Arduino Code for DRA818V FM TRX + 60W RA60H1317M1A
//This code is written by OE3SDE, Simon Dorrer!

//Define CPU-Clock-Speed (16MHz internal clock)
//#define F_CPU 16000000UL

//Define Baudrate for UART
//#define BAUD 9600UL

//Include Libraries
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>

//I2C
#include <Wire.h>

//MCP4728
#include <Adafruit_MCP4728.h>

//DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>

//Libs. for SD1306 OLED-Display
#include <Adafruit_GFX.h>              // Include core graphics
   library for the display
#include <Adafruit_SSD1306.h>          // Include Adafruit_SSD1306
   library to drive the display
#include <Fonts/FreeMono9pt7b.h>       // Add a custom font
/*List of different fonts :
```

```
FreeMono9pt7b.h
FreeMonoBold9pt7b.h
FreeMonoBoldOblique9pt7b.h
FreeMonoOblique9pt7b.h
*/
// --------------------------------------------------------------------------------

//Definitions
//I/O Declaration
//1W FM TRX Board (DRA818V)
//#define TX         0     //PD0 / D0
//#define RX         1     //PD1 / D1
#define SW1          2     //PD2 / D2 - INT0
#define PTT_IN       3     //PD3 / D3 - INT1
#define PTT_OUT      7     //PD7 / D7
#define PD           0     //PB0 / D8
#define H_L          1     //PB1 / D9
#define HC05_TX      2     //PB2 / D10
#define HC05_RX      3     //PB3 / D11
#define E1           0     //PC0 / A0
#define E2           1     //PC1 / A1

//60W FM TRX Board (RA60H1317M1A)
#define PGA_SHDN     5     //PB5 / D13 (LOW --> PGA OFF, HIGH --> PGA ON) + RX Switch
#define TX_SW        4     //PB4 / D12 (HIGH TX Switch)
//#define T1_SW      7     //PD7 / D7  (switches T1 for TRX Switch - LOW for TX, HIGH for RX) --> see PTT_OUT D7
#define DS18B20_OW   6     //PD6 / D6  (One wire input of DS18B20 Temp. Sensor)
#define SWR_REV      7     //PC7 / A7  (ADC samples reverse voltage)
#define SWR_FWD      6     //PC6 / A6  (ADC samples forward voltage)
#define ADC_VDD      3     //PC3 / A3  (ADC samples VDD voltage)

//I2C: OLED Display, MCP4728

//Makros
//RX or TX ATTENUATOR
#define RX_ATTENUATOR_CHANNEL  MCP4728_CHANNEL_A
#define TX_ATTENUATOR_CHANNEL  MCP4728_CHANNEL_B

//PA max. ratings
#define VDD_LOWER_LIMIT 10.7  //in V
#define VDD_UPPER_LIMIT 14.2  //in V
#define SWR_LOWER_LIMIT 0.5
#define SWR_UPPER_LIMIT 2
```

```
#define HEAT_UPPER_LIMIT 45   //in degrees
// --------------------------------------------------------------------------

//Global variables / Objects
uint8_t TX = 0;  // 0... RX; 1... TX
uint8_t switched = 0;

//OLED
Adafruit_SSD1306 display(128, 64);    // Create display
volatile uint16_t timer1Count = 0;

//DRA818 Variables
typedef struct{
  float txFreq;         //TX-Frequency in MHz (134.0000 - 174.0000)
  float rxFreq;         //RX-Frequency in MHz (134.0000 - 174.0000)
  String txCTCSS;       //CTCSS frequency (0000 - 0038); 0000 = "no CTCSS"
  String rxCTCSS;       //CTCSS frequency (0000 - 0038); 0000 = "no CTCSS"
  uint8_t bw;           //Bandwith in KHz (0= 12.5KHz or 1= 25KHz)
  uint8_t squ;          //Squelch level  (0 - 8); 0 = "open"
  uint8_t vol;
  uint8_t prf;
  uint8_t hpf;
  uint8_t lpf;
}DRA818;

DRA818 dra818 = {145.5000, 145.5000, "0000", "0000", 1, 4, 8, 0, 0, 0};

//DS18B20
OneWire oneWire(DS18B20_OW);
DallasTemperature ds18b20(&oneWire);

//MCP4728
Adafruit_MCP4728 mcp4728;

//Rotary Encoder Variables
int counter = 0;
int aState;
int aLastState;

//RA60H1317M1A Variables
float txGain = 10;
float txAtt = 0;
float rxAtt = 0;
```

```
uint8_t monitorPA = 0;   //monitorPA = 1... PA is OK!
//
   ----------------------------------------------------------------------------

//Subprograms
//Hardware Init Methods
void IO_Init()                          //initialize IO
{
  //Define INPUTs
  DDRD &= ~(1 << SW1);                  //set PD2 (SW1) as Input
  PORTD |= (1 << SW1);                  //activate Pull-Up-R at PD2 (SW1)

  DDRD &= ~(1 << PTT_IN);               //set PD3 (PTT-IN) as Input
  PORTD |= (1 << PTT_IN);               //activate Pull-Up-R at PD3 (PTT_IN)

  DDRC &= ~(1 << E1);                   //set PC0 (E1) as Input
  DDRC &= ~(1 << E2);                   //set PC1 (E2) as Input

  DDRC &= ~(1 << ADC_VDD);              //set PC3 (ADC_VDD) as Input
  //PORTC |= (1 << ADC_VDD);              //activate Pull-Up-R at PC3 (
     ADC_VDD)

  DDRC &= ~(1 << SWR_FWD);              //set PC6 (SWR_FWD) as Input
  PORTC |= (1 << SWR_FWD);              //activate Pull-Up-R at PC6 (SWR_FWD)

  DDRC &= ~(1 << SWR_REV);              //set PC7 (SWR_REV) as Input
  PORTC |= (1 << SWR_REV);              //activate Pull-Up-R at PC7 (SWR_REV)

  //Define OUTPUTs
  DDRD |= (1 << PTT_OUT);               //set PD7 (PTT_OUT) as Output
  DDRB |= (1 << PD);                    //set PB0 (PD) as Output
  DDRB |= (1 << H_L);                   //set PB1 (H_L) as Output

  DDRB |= (1 << TX_SW);                 //set PB4 (TX_Switch) as Output
  DDRB |= (1 << PGA_SHDN);              //set PB5 (PGA_SHDN) as Output
}

void Timer1_COMPA_Init()
{
  //Control Registers
  TCCR1A = 0; TCCR1B = 0; TCCR1C = 0; //Reset

  // set CTC mode
  //TCCR1A |= (1 << WGM10);
  //TCCR1A |= (1 << WGM11);
  TCCR1B |= (1 << WGM12);

```

```
  // Set Prescaler 1024 (TCCR1B = (1 << WGM12) | (0x5 << CS10);)
  TCCR1B |= (1 << CS12);
  TCCR1B &= ~(1 << CS11);
  TCCR1B |= (1 << CS10);

  // initialize compare value
  OCR1A = 16;

  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);
}

void Timer2_COMPA_Init()
{
  //Timer2 settings
  TCCR2A = 0x00; TCCR2B = 0x00; //Reset

  //Enable Timer2 CTC Mode
  //TCCR2A |= (1 << WGM20);
  TCCR2A |= (1 << WGM21);
  //TCCR2B |= (1 << WGM22);

  //Prescaler
  // 1024 prescaling for Timer2 (TCCR2B = (0x7 << CS20);)
  TCCR2B |= (1 << CS20);
  TCCR2B |= (1 << CS21);
  TCCR2B |= (1 << CS22);

  //Initialize compare value
  OCR2A = 0;

  //initialize TIMER0-Counter
  //TCNT2 = 0; // set counter value FORMEL: x = maximaler Zaehlwert -
      ((CPUtakt/PRESCALER)/ gesuchte Frequenz)

  //disable Timer compare interrupt
  TIMSK2 &= ~(1 << OCIE2A);
}

void INT0_Init()
{
  //enable Interrupt
  EIMSK |= (1 << INT0); //Ext. Int0 ein

  //Set falling Edge Interrupt (EICRA |= (0x2 << ISC00);)
  EICRA &= ~(1 << ISC00);
  EICRA |= (1 << ISC01);
}
```

```
void INT1_Init()
{
  //enable Interrupt
  EIMSK |= (1 << INT1); //Ext. Int1 ein

  //Set rising & falling Edge Interrupt (EICRA |= (0x2 << ISC10);)
  EICRA |= (1 << ISC10);
  EICRA &= ~(1 << ISC11);
}

void ADC_Init()
{
  //ADC Setup
  //Set Reference Voltage (VCC = VREF)
  ADMUX &= ~(1 << REFS1);
  ADMUX |= (1 << REFS0);

  //Prescaler --> 128 (50kHz - 200kHz)
  ADCSRA |= (1 << ADPS0);
  ADCSRA |= (1 << ADPS1);
  ADCSRA |= (1 << ADPS2);

  ADCSRA |= (1 << ADEN); //Enable ADC

  //Dummy Readout to warm up the ADC
  ADCSRA |= (1<<ADSC);          //Start ADC conclusion
  while (ADCSRA & (1<<ADSC)){}  //wait until ADC is ready
  (void) ADC;
}
uint16_t ADC_ReadValue(uint8_t channel)
{
  ADMUX = (ADMUX & ~(0x1F)) | (channel & 0x1F); //which ADCx? Bit
     Mask to secure ADMUX settings in Init() function

  ADCSRA |= (1<<ADSC);          //Start ADC conclusion
  while (ADCSRA & (1<<ADSC)){}  //wait until ADC is ready

  return ADC;                   //return the ADC value
}

//DRA818V Methods
void DRA818V_setGroup()
{
  Serial.print("AT+DMOSETGROUP=");          // begin message
  Serial.print(dra818.bw);
  Serial.print(",");
  Serial.print(dra818.txFreq, 4);
  Serial.print(",");
```

```
  Serial.print(dra818.rxFreq, 4);
  Serial.print(",");
  Serial.print(dra818.txCTCSS);
  Serial.print(",");
  Serial.print(dra818.squ);
  Serial.print(",");
  Serial.println(dra818.rxCTCSS);
}
void DRA818V_setVolume()
{
  Serial.print("AT+DMOSETVOLUME=");
  Serial.println(dra818.vol);
}
void DRA818V_setFilter()
{
  Serial.print("AT+SETFILTER=");
  Serial.print(dra818.prf);
  Serial.print(",");
  Serial.print(dra818.hpf);
  Serial.print(",");
  Serial.println(dra818.lpf);
}
void DRA818V_Init()                    // initialize DRA818V
{
  PORTB &= ~(1 << PD);
  delay(2000);

  //I/O
  PORTD |= (1 << PTT_OUT);       // set PD7 (PTT_OUT) HIGH at the beginning (RX-Mode)
  PORTB |= (1 << PD);            // set PB0 (PD) HIGH at the beginning (Normal-Mode)
  PORTB &= ~(1 << H_L);          // set PB1 (H/L) LOW at the beginning (LOW-Power = 0.5W)

  //UART
  Serial.begin(9600);
  delay(10);
  DRA818V_setGroup();
  delay(500);
  DRA818V_setVolume();
  delay(500);
  DRA818V_setFilter();
  delay(500);
}

//RA60H1317M1A Methods
float getVDD()       //Get VDD Voltage (10.8VDC to 13.6VDC)
```

```
{
  float vdd = 0;
  // #PJN: You may spend a LOT of time in this loop => make ADC reading asynchronous

  vdd = ADC_ReadValue(ADC_VDD) * 5.0 / 1024.0;

  return((vdd * ((4700 + 18000) / 4700)) + 2.2);  //returns supply voltage without voltage division (magic numbers equals the voltage divider resistor values)
}
float getSWR()      //Get SWR of output load (should be between 0.5 and 2)
{
  float rev = 0;
  float fwd = 0;
  // #PJN: You may spend just more time in this loop => make ADC reading asynchronous
  for(int i = 0; i < 10; i++)   //10 iterations for a more precise result
  {
    rev = rev + ADC_ReadValue(SWR_REV);
    fwd = fwd + ADC_ReadValue(SWR_FWD);
  }
  rev = rev / 10 / 1024 * 5;
  fwd = fwd / 10 / 1024 * 5;

  //SWR Calculation
  return((fwd + rev) / (fwd - rev));
}
float getPWR()
{
  float fwd = 0;
  for(int i = 0; i < 10; i++)   //10 iterations for a more precise result
  {
    fwd = fwd + ADC_ReadValue(SWR_FWD);
  }
  fwd = fwd / 10 / 1024 * 5;

  //PWR Calculation
  return((fwd * fwd) / 50);
}
float getHeat()     //Get heat of RA60H1317M1A measured by DS18B20, should be smaller than 50
{
  ds18b20.requestTemperatures();
  return(ds18b20.getTempCByIndex(0));
```

```
}
void setTXGain(float gain)  //Set Gain of RA60H1317M1A via MCP4728 (VOUTD)
{
  if(gain == 0)
  {
    mcp4728.setChannelValue(MCP4728_CHANNEL_D, 0);
  }
  else
  {
    //ToDo Formula
    mcp4728.setChannelValue(MCP4728_CHANNEL_D, 4095);
  }
}
void setAttenuation(MCP4728_channel_t channel, float att) //Attenuation of U11 (RX) or U12 (TX) (F2255NLGK)
{
  //Note that the PGA-103 has a constant gain of 25dB for RX path! RX_ATTENUATOR_CHANNEL MCP4728_CHANNEL_A
  //Note that an 6dB Attenuator is assembled on board for TX path! TX_ATTENUATOR_CHANNEL MCP4728_CHANNEL_B

  if(att > 35)
  {
    mcp4728.setChannelValue(channel, 2048);
  }
  else if(att < 2.5)
  {
    mcp4728.setChannelValue(channel, 0);
  }
  else
  {
    att = att * (-1);
    mcp4728.setChannelValue(channel, (uint16_t)(4096 * (-0.03692 * att + 0.60769) / 5));  //0.7V (-2.5dB) to 1.9V (-35dB), magic numbers are calculated according to the attenuation curve of the datasheet!
  }
}
void switchPAtoTX(uint8_t TX_Att, uint8_t TX_Gain)
{
  if(monitorPA == 1)
  {
    //turn OFF RX:
    mcp4728.setChannelValue(MCP4728_CHANNEL_C, 0);    //turn RX Path in TRX Switch OFF (VOUTC to LOW)
    PORTB &= ~(1 << PGA_SHDN);                        //turn OFF PGA RX-Amp. and RX Switch (D13 to LOW)
```

```
    setAttenuation(RX_ATTENUATOR_CHANNEL, 36);          //set full attenuation of U11 (F2255NLGK)

    delay(100);     //100ms settling time

    //turn ON TX:
    PORTD &= ~(1 << PTT_OUT);                           //turn TX Path in TRX Switch ON (D7 LOW)
    PORTB |= (1 << TX_SW);                              //turn on TX Switch (D12 to HIGH)
    setAttenuation(TX_ATTENUATOR_CHANNEL, TX_Att);      //set attenuation of U12 (F2255NLGK)
    setTXGain(TX_Gain);                                 //Set TX gain (VGG (Gate) voltage) of RA60H1317M1A
    delay(100);                                         //100ms settling time
  }
  else
  {
    switchPAtoRX(rxAtt);
  }
}
void switchPAtoRX(float RX_Att)
{
  //turn OFF TX:
  setTXGain(0);                                         //Set gain of RA60H1317M1A to 0
  setAttenuation(TX_ATTENUATOR_CHANNEL, 36);            //set full attenuation of U12 (F2255NLGK)
  PORTB &= ~(1 << TX_SW);                               //turn OFF TX Switch (D12 to LOW)
  PORTD |= (1 << PTT_OUT);                              //turn TX Path in TRX Switch OFF (D7 HIGH)
  delay(100);                                           //100ms settling time

  //turn ON RX
  mcp4728.setChannelValue(MCP4728_CHANNEL_C, 4095);     //turn RX Path in TRX Switch ON (VOUTC to HIGH)
  PORTB |= (1 << PGA_SHDN);                             //turn ON PGA RX-Amp. and RX Switch (D13 to HIGH)
  setAttenuation(RX_ATTENUATOR_CHANNEL, RX_Att);        //set attenuation of U11 (F2255NLGK)
  delay(100);                                           //100ms settling time
}
void verifyPA()   //Shutdown RA60H1317M1A according to SWR, VDD and Heat
```

```
{
  float vdd = getVDD();
  float swr = getSWR();
  float heat = getHeat();
  Serial.print("VDD: ");
  Serial.println(vdd);
  Serial.print("SWR: ");
  Serial.println(swr);
  Serial.print("HEAT: ");
  Serial.println(heat);

  //if((vdd >= VDD_LOWER_LIMIT && vdd <= VDD_UPPER_LIMIT) && (swr >= SWR_LOWER_LIMIT && swr <= SWR_UPPER_LIMIT) && heat <= HEAT_UPPER_LIMIT)
  if((vdd >= VDD_LOWER_LIMIT && vdd <= VDD_UPPER_LIMIT) && heat <= HEAT_UPPER_LIMIT)
  {
    monitorPA = 1;  //PA is OK!
  }
  else
  {
    monitorPA = 0;
    if(TX == 1)
    {
      switchPAtoRX(rxAtt);
    }
  }
}

//Display Methods
void displayInit()
{
  delay(100);                                // This delay is needed to let the display to initialize
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){ // Address 0x3C for 128x32
    Serial.println(F("SSD1306 allocation failed"));
    while (1){
      delay(10);
    }
  }
  display.clearDisplay();                    // Clear the buffer
  display.setTextColor(WHITE);               // Set color of the text
  //display.setRotation(2);                   // Set orientation. Goes from 0, 1, 2 or 3
  display.setTextWrap(false);                // By default, long lines of text are set to automatically "wrap" back to the
```

```
    leftmost column.
  // To override this behavior (so text will run off the right side of the display - useful for scrolling marquee effects), use setTextWrap(false).
  display.dim(0);                              //Set brightness (0 is maximun and 1 is a little dim)
  display.setFont(&FreeMono9pt7b);             // Set a custom font
  display.setTextSize(0);                      // Set text size. We are using a custom font so you should always use the text size of 0
  display.display();                           // Print everything we set previously
}
void displayStartScreen()
{
  display.setCursor(10, 15);                   // (x,y)
  display.println("60W FM TRX");               // Text or value to print
  display.setCursor(10, 35);                   // (x,y)
  display.println("by OE3SDE");                // Text or value to print
  display.setCursor(10, 55);                   // (x,y)
  display.println("S. Dorrer");                // Text or value to print
  display.display();                           // Print everything we set previously
}
void refreshDisplay()
{
  display.clearDisplay();

  // TX / RX
  if(TX == 1)
  {
    display.setCursor(10, 15);                 // (x,y)
    display.println("TX");
  }
  else if(TX == 0)
  {
    display.setCursor(10, 15);                 // (x,y)
    display.println("RX");
  }

  // Power
  display.setCursor(50, 15);                   // (x,y)
  display.println("PWR:.5W");

  // Frequency
```

```
  display.setCursor(30, 35);                          // (x,y)
  display.println(dra818.txFreq);

  //Values
  display.setCursor(2, 55);                        // (x,y)
  display.print(getVDD());                          // Text or value to
     print
  display.print("␣");
  display.print(getHeat());                        // Text or value to
     print
  display.display();
}
//
   ---------------------------------------------------------------------------

//Setup (Initialize hardware)
void setup()
{
  //HW Init
  IO_Init();
  INT0_Init();
  INT1_Init();
  Timer1_COMPA_Init();
  Timer2_COMPA_Init();
  ADC_Init();

  // OLED Init
  displayInit();
  displayStartScreen();

  //DRA818V Init
  DRA818V_Init();

  //DS18B20 Init
  ds18b20.begin();

  //MCP4728 Init
  if(!mcp4728.begin()){
    Serial.println("Failed␣to␣find␣MCP4728␣chip!");
    while (1){
      delay(10);
    }
  }

  //RA60H1317M1A
  verifyPA();
  switchPAtoRX(rxAtt); //standard attenuation of 10dB
```

```

  sei(); //enable global interrupts

  //Reads the initial state of the outputA (E2)
  aLastState = digitalRead(E2);   // #PJN: wondering why you're now using the digitalRead() because output handling is done via direct register access - it's however OK anyways

  display.clearDisplay();
  display.display();
}
// ---------------------------------------------------------------------------

//Loop
void loop()
{
  if(timer1Count >= 500)
  {
    timer1Count = 0;
    verifyPA();
    refreshDisplay();
  }

  if(TX == 1 && switched == 1)
  {
    switchPAtoTX(txAtt, txGain);     //PA PCB is transmitting now!
    //PORTD &= ~(1 << PTT_OUT);      //DRA818V TX (done in switchPAtoTX)
    switched = 0;
  }
  else if (TX == 0 && switched == 1)
  {
    //PORTD |= (1 << PTT_OUT);       //DRA818V RX (done in switchPAtoRX)
    switchPAtoRX(rxAtt);             //PA PCB is receiving now!
    switched = 0;
  }

  // #PJN: Because of the above code you'll get a lot of jitter when reading the encoder below
  aState = digitalRead(E2); // Reads the "current" state of the outputA
  // If the previous and the current state of the outputA are different, that means a Pulse has occured
  if (aState != aLastState)
  {
```

```
    // If the outputB state is different to the outputA state, that means the encoder is rotating clockwise
    if (digitalRead(E1) != aState)
    {
      counter++;
    }
    else
    {
      counter--;
    }
  }
  aLastState = aState; // Updates the previous state of the outputA with the current state
}
// ---------------------------------------------------------------------------

//ISP (Interrupt Service Routine)
ISR(TIMER1_COMPA_vect)  // every 1ms
{
  timer1Count++;
}

ISR(TIMER2_COMPA_vect) // execute by TIMER2 CTC
{
  static uint16_t timer2Count = 0; //only at first ISR call

  timer2Count++; //inkrementieren

  if(timer2Count == 4000) //256ms sperren
  {
    //enable INT0
    EIMSK |= (1 << INT0); //Ext. Int0 on

    //enable INT1
    EIMSK |= (1 << INT1); //Ext. Int0 on

    //clear INTF0
    EIFR |= (1 << INTF0);

    //clear INTF1
    EIFR |= (1 << INTF1);

    //disable TIMER2 compare interrupt
    TIMSK2 &= ~(1 << OCIE2A);

    //reset of counter value
```

```
    timer2Count = 0;
  }
}

ISR(INT0_vect)  //Rotary Encoder Switch
{
  static uint8_t encoderPressedCounter = 1; //only at first ISR call

  uint8_t temp = SREG; //Store SREG

  encoderPressedCounter++;

  //DEBOUNCE
  //disable INT0
  EIMSK &= ~(1 << INT0); //Ext. INT0 off
  //enable TIMER2 compare interrupt
  TIMSK2 |= (1 << OCIE2A);

  switch (encoderPressedCounter)
  {
    case 1:
    // #PJN: Off topic: channel spacing on 2m is 12.5 kHz ... maybe
       just increase/decrease frequency values instead of fixed ones?
    dra818.txFreq = 145.5000;   //TX-Frequency in MHz
    dra818.rxFreq = 145.5000;   //RX-Frequency in MHz
    DRA818V_setGroup();
    break;
    case 2:
    dra818.txFreq = 144.2000;   //TX-Frequency in MHz
    dra818.rxFreq = 144.2000;   //RX-Frequency in MHz
    DRA818V_setGroup();
    break;
    case 3:
    dra818.txFreq = 144.5000;   //TX-Frequency in MHz
    dra818.rxFreq = 144.5000;   //RX-Frequency in MHz
    DRA818V_setGroup();
    break;
    case 4:
    dra818.txFreq = 145.2000;   //TX-Frequency in MHz
    dra818.rxFreq = 145.2000;   //RX-Frequency in MHz
    DRA818V_setGroup();
    break;
    case 5:
    dra818.txFreq = 145.0000;   //TX-Frequency in MHz
    dra818.rxFreq = 145.0000;   //RX-Frequency in MHz
    DRA818V_setGroup();
    encoderPressedCounter = 0;
    break;
```

```
    default:
    encoderPressedCounter = 0;
    break;
  }

  SREG = temp;
}

ISR(INT1_vect) //PTT-IN (active LOW)
{
  //DEBOUNCE
  //disable INT1
  //EIMSK &= ~(1 << INT1); //Ext. INT1 off
  //enable TIMER2 compare interrupt
  //TIMSK2 |= (1 << OCIE2A);

  switched = 1;

  if(!(PIND & (1 << PTT_IN)))         //PTT_IN changed to LOW
  {
    //switchPAtoTX(txAtt, txGain);    //PA PCB is transmitting now!
    //PORTD &= ~(1 << PTT_OUT);       //DRA818V TX (done in switchPAtoTX)
    TX = 1;
  }
  else if (PIND & (1 << PTT_IN))      //PTT_IN changed to HIGH
  {
    //PORTD |= (1 << PTT_OUT);        //DRA818V RX (done in switchPAtoRX)
    //switchPAtoRX(rxAtt);            //PA PCB is receiving now!
    TX = 0;
  }
}
//-------------------------------------------------------------------------
//End of Code!
```

# Chapter 7

# Enclosure

Unfortunately, the 3D enclosure is not finished yet. In the end, the enclosure will be 3D printed.

# Chapter 8

# Result Pictures

## 8.1 Assembled PCB

ToDo Text

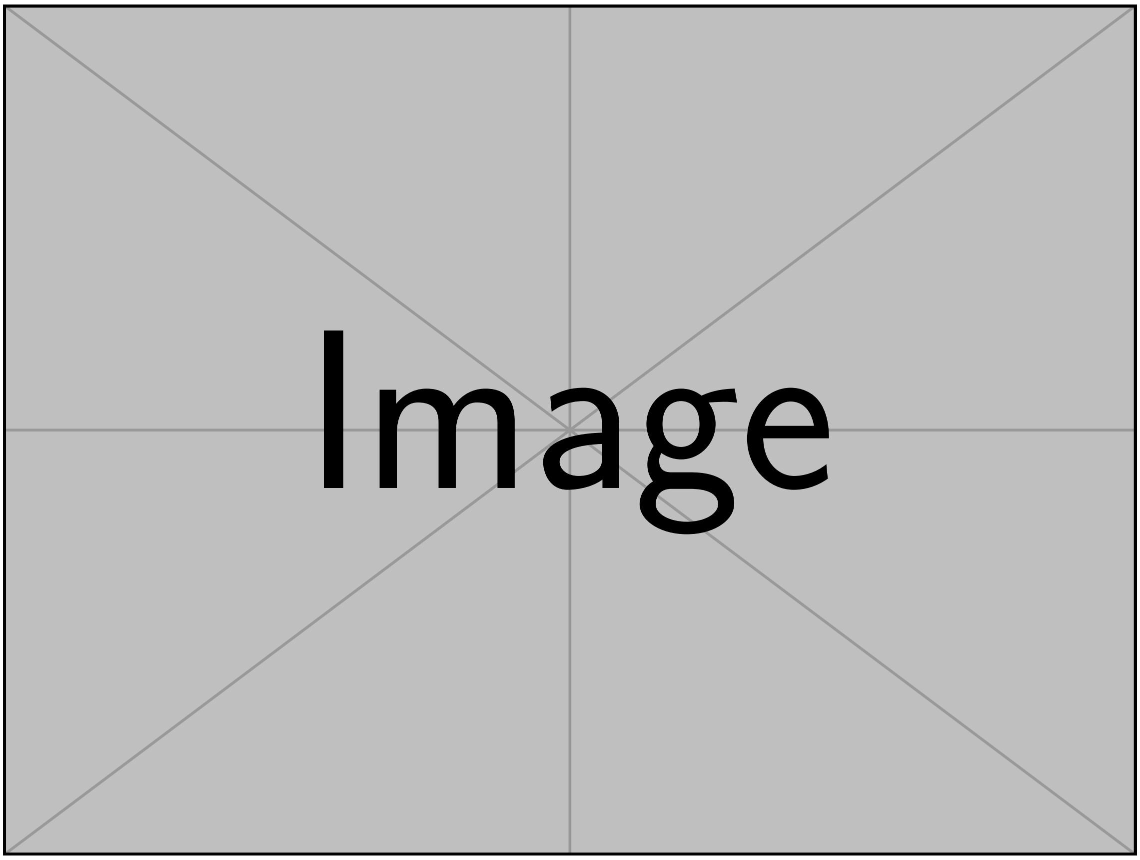



Figure 8.1: Assembled PCB

## 8.2 Transceiver with Power Amplifier

ToDo Text

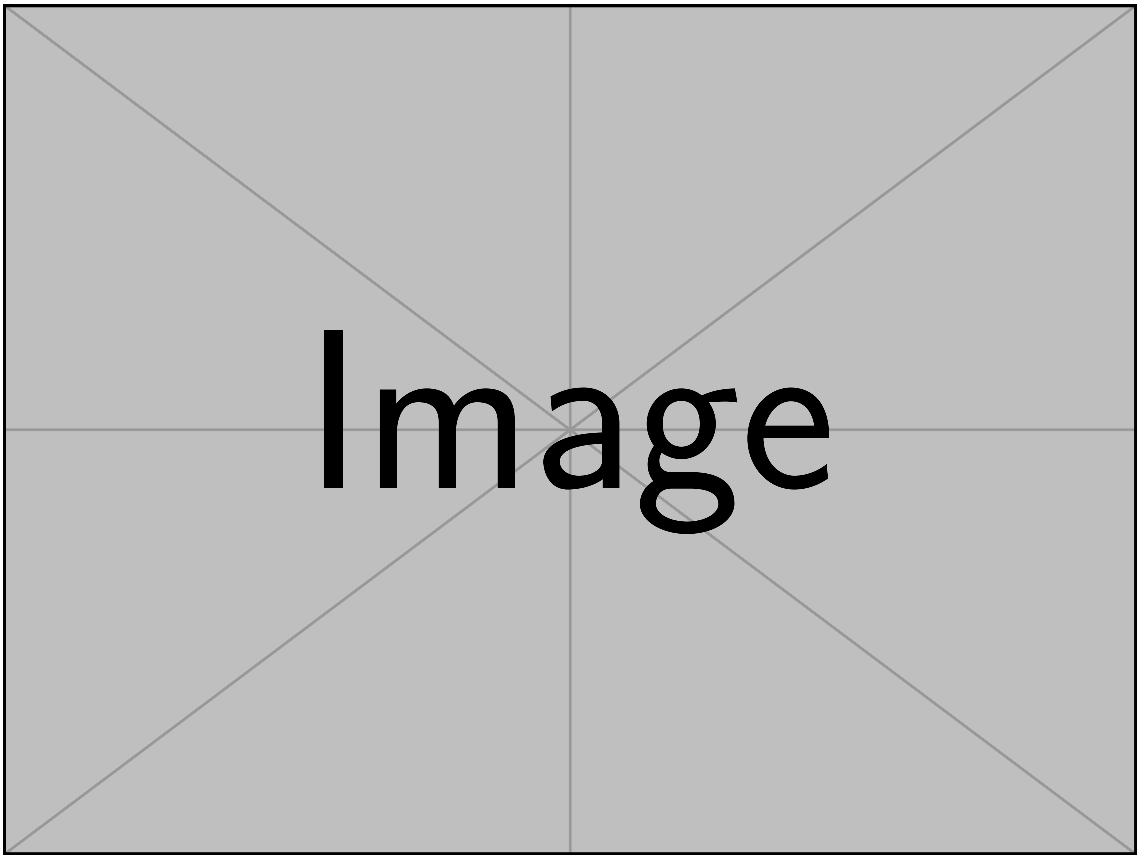



Figure 8.2: DRA818V Transceiver with Power Amplifier

## 8.3 Implemented in Enclosure

Unfortunately, the 3D enclosure is not finished yet.

# List of Abbreviations

| | |
|---|---|
| **ADC** | analog to digital converter |
| **CMOS** | complementary metal oxide semiconductor |
| **CW** | continuous wave |
| **DAC** | digital to analog converter |
| **MCU** | microcontroller unit |
| **IC** | integrated circuit |
| **LNA** | low noise amplifier |
| **LPF** | lowpass filter |
| **HPF** | highpass filter |
| **BPF** | bandpass filter |
| **LUT** | lookup table |
| **PA** | power amplifier |
| **PCB** | printed circuit board |
| **RF** | radio frequency |
| **RX** | receive |
| **SNR** | signal to noise ratio |
| **SSB** | single sideband |
| **TX** | transmit |
| **TRX** | transceiver |
| **FM** | frequency modulation |

# List of Figures

# List of Tables

# Bibliography

[1] M0UKD. Online air cored inductor calculator. [Online]. Available: https://m0ukd.com/calculators/air-cored-inductor-calculator/

[2] C. ELECTRONICS. Online coplanar waveguide grounded impedance calculator. [Online]. Available: https://chemandy.com/calculators/coplanar-waveguide-with-ground-calculator.htm

[3] B. C. Wadell, *Transmission Line Design Handbook.* Artech House, 1991.

[4] R. E. Corporation. Datasheet f2255nlgk - digital controllable rf attenuator. [Online]. Available: https://www.mouser.at/datasheet/2/698/IDT_F2255_Datasheet_DST_20180209-1997542.pdf