BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
DEPARTMENT OF BROADBAND INFOCOMMUNICATIONS AND ELECTROMAGNETIC THEORY

Thesis

of

Eszter Szabó

BSc electrical engineer candidate, whose topic is

Analog signal suppressor for DVB-T band passive radar

- Study the background of the principles of passive radars based on the cited references below
- Get familiar with the operating principle and the buildup of the DVB-T based passive radar developed at the Radar Research Group
- Design an analog signal suppressor to minimize the reference signal in the surveillance channels with the following requirements:
  - The signal suppressor shall support up to 7 surveillance channels and 1 reference channel
  - The signal suppressor shall operate between 460 MHz and 710 MHz
  - The amplitude and phase angle of the received reference signal shall be modified and superposed with the surveillance signal
  - The reference signal shall be suppressed by at least 10 dB
  - The board shall be controlled by a PC
  - The board shall be powered from a 5V power source, preferably USB
- Develop an algorithm that suppresses the reference signal in the surveillance channels in an analogous way
- Using open source CAD software (preferably KiCAD):
  - Design the schematic of the device
  - Design the PCB layout of the device
- Populate the PCB and carry out bench tests with it
- Carry out test measurements in the laboratory and out on the field to verify the operation of the circuit and document the results

References:

- *Seller, Rudolf; Pető, Tamás; Dudás, Levente; Kovács, Levente*: Passzív radar. II. rész; HADITECHNIKA 54 : 1 pp. 43-47. , 4 p. (2020)
- *Seller, Rudolf; Pető, Tamás; Dudás, Levente; Kovács, Levente*: Passzív radar. I. rész; HADITECHNIKA 53 : 6 pp. 51-55. , 5 p. (2019)
- *Pető, Tamás; Dudás, Levente; Seller, Rudolf*:Analog Direct Path Interference Suppression for FM Based Passive Radars; In: IEEE (szerk.) 28th International Conference Radioelektronika (RADIOELEKTRONIKA); New York (NY), Amerikai Egyesült Államok : IEEE (2018) 4 p. Paper: 8376381 , 6 p.

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Broadband Infocommunications and Electromagnetic Theory

Analog signal suppressor for DVB-T band passive radar

Bachelor's Thesis

Author
Szabó Eszter

Advisor
Herman Tibor

2022

HALLGATÓI NYILATKOZAT

Contents

Kivonat

Napjainkban egyre nagyobb figyelmet kapnak a passzív radaros technológiák, hiszen számos előnnyel rendelkeznek a klasszikus aktív radarokkal szemben. Azáltal, hogy nem sugároznak ki elektromágneses energiát, hanem csak vevő irányban működnek, nem felderíthetőek.

A Mikrohullámú Távérzékelés Laboratórium fejleszt egy DVB-T sávban működő passzív radart. Feladatom volt, hogy ennek a radarnak a teljesítményét és hatótávolságát javítsam egy külső részegység beiktatásával. Dolgozatom ennek az áramkörnek a megtervezését, realizálását, kalibrálását és mérését részletezi.

A hatótávolság korlátozásában jelentős szerepe van a felderítő csatornák jelébe beszivárgó referencia jelnek. Ennek kiküszöbölése érdekében terveztem egy referencia jelkioltót, mely a felderítő csatornák jeléből kivonja a referencia jelet.

A jelkioltás IQ-moduláció segítségével történik, mely végén a két jelet ellenfázisban összegezve megkapjuk a referencia jel mentes felderítő csatorna jelét. Ennek megértése érdekében dolgozatom elején röviden összefoglalom a szükséges elméleti hátteret.

Dolgozatom további részében kifejtem a jelkioltás folyamatának alapgondolatát sokkal részletesebben is és a tervezés különböző lépéseit. Bemutatom a tervezés során végzett szimulációkat és méréseket, az egyes részegységek működését.

Részletezem a prototípus során felmerült problémákat és ezen problémák megoldását a későbbi áramkörben. Indoklom a nyomtatott huzalozású lemezek elkészítése során alkalmazott szempontokat és technikákat.

Írásom a jelkioltóval végzett mérésekkel zárul, ahol megemlítem a végzett mérések eredményét és az ezt segítő kalibrációs eljárás menetét, majd megjelenítem és értékelem a kapott adatokat.

Abstract

Today, passive radar technologies are attracting more and more attention, as they have many advantages over classical active radars. They are not detectable because they do not emit electromagnetic energy, but only operate in the receiver direction.

The Microwave Remote Sensing Laboratory is developing a passive radar operating in the DVB-T band. I was tasked to improve the performance and range of this radar by adding an external component. My thesis details the design, implementation, calibration and measurement of this circuit.

The reference signal that infiltrates the signals of the surveillance channels plays a significant role in limiting the range. To eliminate this, I have designed a reference signal suppressor to subtract the reference signal from the signals of the surveillance channels.

Signal suppression is done by IQ-modulation, at the end of which the two signals are summed in counter-phase to obtain the signal of the reference signal-free surveillance channel. To understand this, I will briefly summarize the necessary theoretical background at the beginning of my thesis.

In the rest of my thesis, I will explain the main idea of the signal suppression process in much more detail and the different steps of the design. I will describe the simulations and measurements carried out during the design and the operation of the different components.

I will detail the problems encountered during the prototype and how these problems will be solved in the final circuit. I will justify the considerations and techniques used in the design of the printed circuit boards.

I will conclude with the measurements made with the signal suppressor, where I will mention the results of the measurements and the calibration procedure that assisted them, then present and evaluate the data obtained.

Chapter 1

Analog signal suppressor

In the case of passive radars, the reference signal that infiltrates the surveillance channels plays a major role in limiting the range. The reason is that the surveillance channels also receive the direct signal, which has a much higher power density than the reflected signal at the given location.

The impact of this infiltration on detection during signal processing can be reduced by various methods, but not completely eliminated. The role of the analog reference signal suppressor is to reduce the amount of signal infiltration entering the surveillance channels in an analogous way, before the signal processing chain, thus improving the sensitivity of the radar.

The Microwave Remote Sensing Laboratory has already produced a signal suppressor. It had fewer channels and operated in the FM band.[1] My task was to correct the known errors and develop a better performing version, increasing, for example, the range, and to generate a calibration procedure for the signal substraction, all this in another frequency range. The aim was to achieve a signal suppression capability of at least 10 dB.

For my thesis topic I had to design the reference signal suppressor for an 8-element antenna system of a DVB-T based passive radar, where the radar consists of 1 reference and 7 surveillance channels. The frequency range of DVB-T is 460-710 MHz.

Chapter 2

Scientific background

2.1 Passive radar

RADAR is an acronym (RAdio Detection And Ranging), which stands for both a measurement procedure and the device that performs it. The way radar works is that it emits electromagnetic energy and then detects the reflected signal from various reflecting objects. From changes in the reflected signal, we can obtain various target parameters. Radar can directly measure the target angle, radial distance and radial angle.

In classical active radars, the electromagnetic pulse is generated by the radar itself, using the transmitter device. If the transmitter and receiver antennas are identical or only close to each other, it is called a monostatic arrangement. If the transmitter and receiver are far apart, it is called bistatic. If several transmitters and several receivers are placed at a large distance from each other, a multistatic arrangement is obtained.

Compared to active radar, passive radar differs in that it does not emit an illuminating signal, but uses existing sources in the environment. These typically come from a broadcasting or telecommunications networks. In my case, this is DVB-T, or terrestrial digital video broadcasting.

It comes from the operation of the passive radar that only a bistatic or multistatic arrangement is possible here, since the device is used only in the receiver direction.

Radar has many advantages over visual observation. It works both day and night, no matter how dark or bright it is, and it does so at a long range. It can be used in any weather conditions, rain or fog, even through walls and thick snow.[2]

2.2 IQ modulation

In the device I made, the essence of the analog reference signal suppressior, the signal suppression itself, is done by IQ modulation.

In the IQ modulator, 'I' stands for in-phase signal and 'Q' for quadrature signal.

The diagram below 2.1 shows the structure of the IQ modulator.

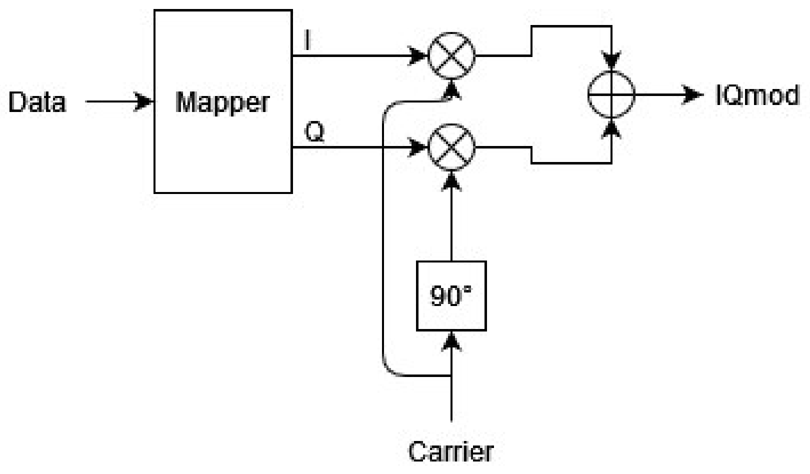


Figure 2.1: IQ modulator

The I signal path goes directly to the mixer, while Q is 90° out of phase. In a vector diagram, the phase component is plotted on the horizontal (real) axis and the quadrature component on the vertical (imaginary) axis, i.e. they are orthogonal to each other. A mapper is connected ahead of the IQ modulator which is fed with the data stream to be transmitted. The output signals of the mapper are the modulation signals for the I- and Q-mixers. These signals are no longer data signals but signed voltages.

The modulated I and Q signals are combined in an adder, so the output of the modulator is the sum of the output signals of the I mixer and the Q mixer.

The output signals of the I and Q paths are sine and cosine signals, differ only in amplitude and have the same frequency, so the output signal of the modulator summed by them will also be a sinusoidal signal, varying in phase and amplitude. On this basis, the phase and amplitude of the resulting output signal can be varied depending on the control signals. [3]

Chapter 3

Description of the operation

In analog reference signal suppression, the aim is to subtract the signal of the reference channel from the signals of the seven surveillance channels. This substraction is in fact performed with the same amplitude, using counter-phase summation. This requires a complex mixer on each channel, which changes the reference signal to the appropriate amplitude and phase.

The principle of the operation is shown in Figure 3.1.

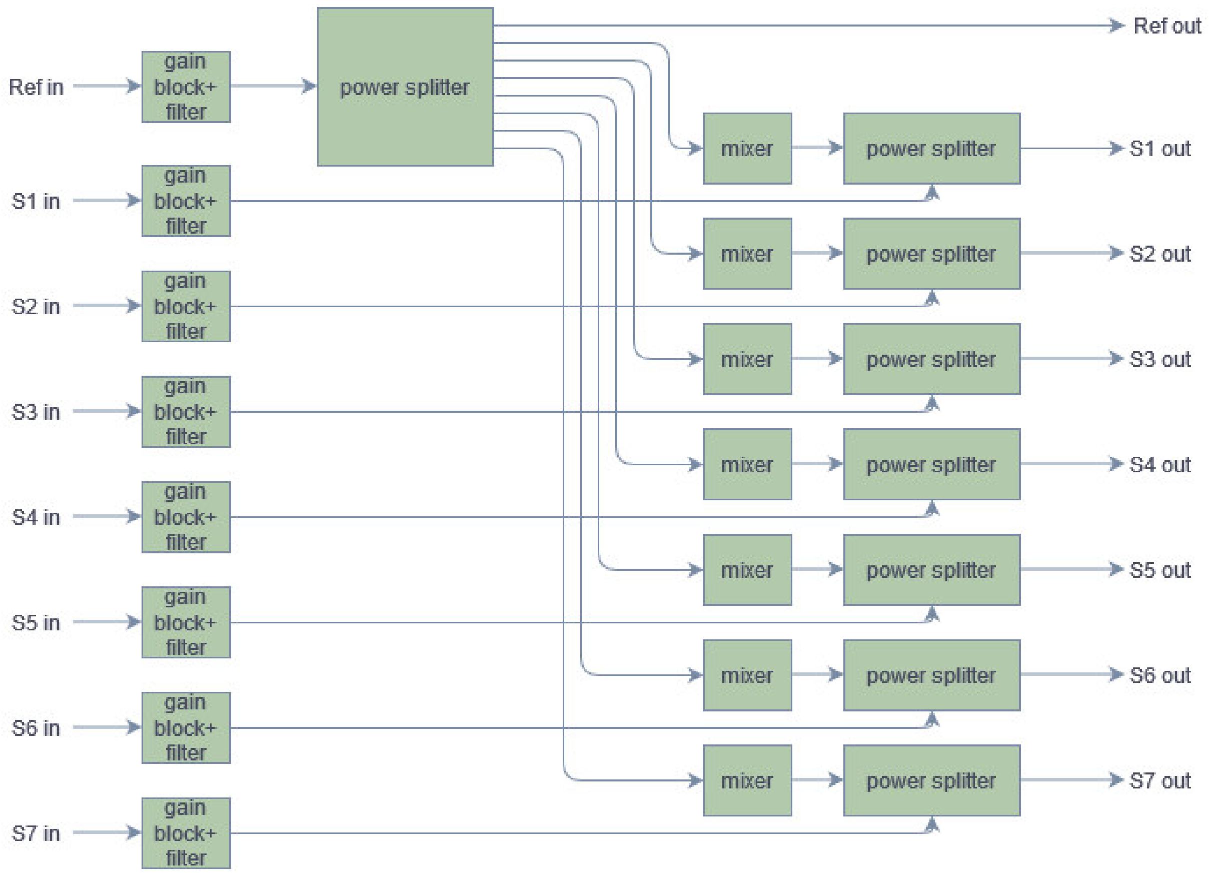


Figure 3.1: The block diagram of the signal suppressor

The passive radar has one reference and seven surveillance channels for inputs. Each channel has an input stage with RF amplification and band-pass filtering. The signal from the reference channel is then divided into eight parts by a power divider, the outputs of which are fed into the mixer block. Here, complex multiplication is performed using a controllable mixer. The system is controlled by a microcontroller. The output power dividers are implemented resistively so that they also act as power combiners.

The outputs of the signal suppressor are the reference channel that is passed on after filtering, and the seven surveillance channels, which are now reference signal-free.

The diagram in Figure 3.1 does not include everything that will be needed to make the device work, but only illustrates the principle of implementation. The parts that are left out, as well as the different blocks, will be explained later.

Chapter 4

The structure of the schematic

The schematic was created by using Eeschema, a subprogram of the KiCad software.

The gain and filtering block shown in the block diagram in Figure 4.1 are implemented by a hierarchical sheet called the input stage, while the multiplication and subsequent summation is implemented by a hierarchical sheet called the IQ mixer.

The diagram also shows the distribution of the reference channel and the connection of the control unit and the voltage-inverting switching regulator.

The channels are accessible via SMA connectors.

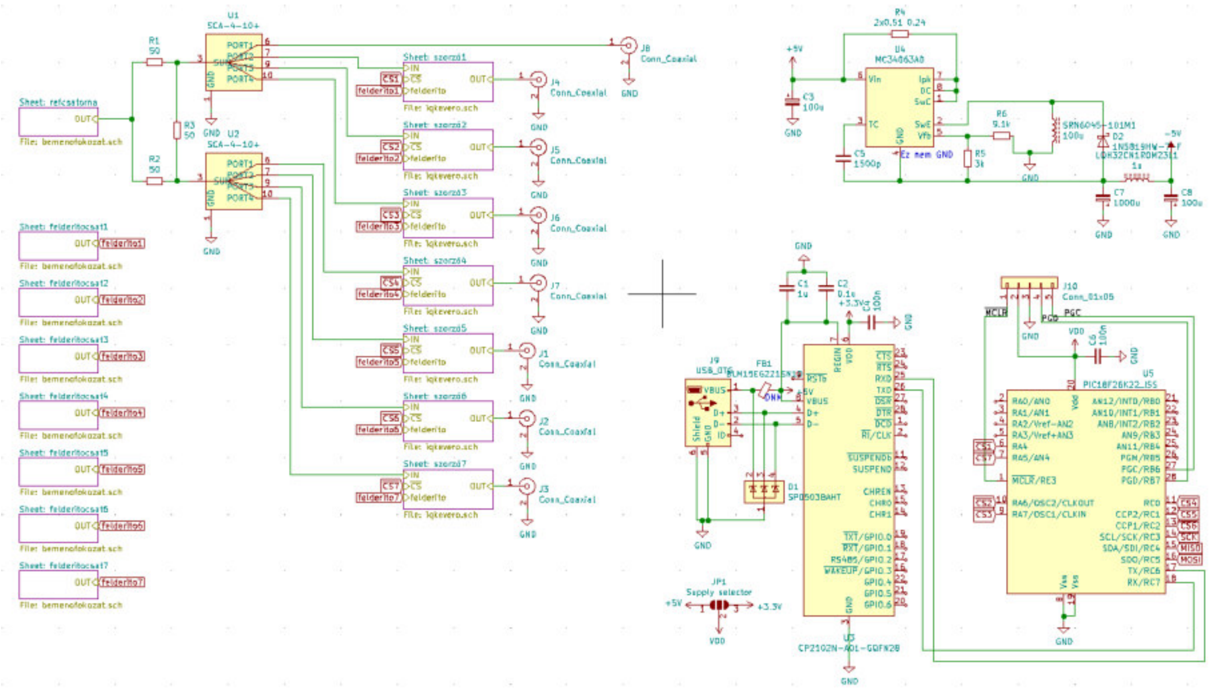


Figure 4.1: The schematic of the signal suppressor

4.1 Input stage

There is an identical input stage, in Figure 4.2, at the beginning of each channel . This includes a gain block and a band-pass filter.

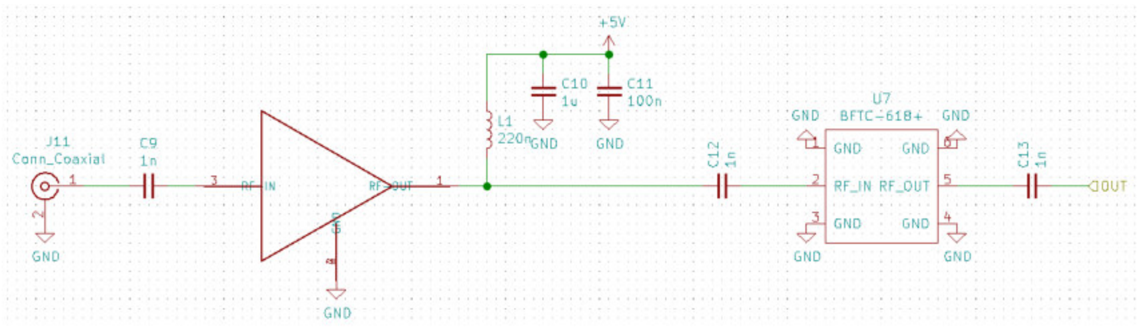


Figure 4.2: The schematic of the input stage

4.1.1 RF gain block

The RF signal suffers attenuation in several places from the different components, so it will need to be amplified. For this purpose, I have chosen the PSA4-5043+ low noise amplifier, which is capable of 20 dB gain with an ultra low noise figure of 0.75 dB and implemented the circuitry suggested by the datasheet [4]. The low noise amplifier amplifies the input RF signals without significantly worsening their signal-to-noise ratio. This type operates in a wide range, from 50 MHz to 4 GHz, allowing to apply it anywhere in the RF paths. The gain blocks shown in the signal paths are the same at all locations.

4.1.2 Band-pass filter

The signals from the antennas should be filtered to the frequency band we use, at the beginning of the signal path.

To achieve this, I created a simulation for a band-pass filter using a program called Elsie. I did this by successively creating a schematic for a high-pass filter with 460 MHz cutoff frequency and a low-pass filter with 710 MHz, as I had to design for the DVB-T frequency range, i.e. the 460-710 MHz band.

Here, I will only demonstrate the high-pass filter.

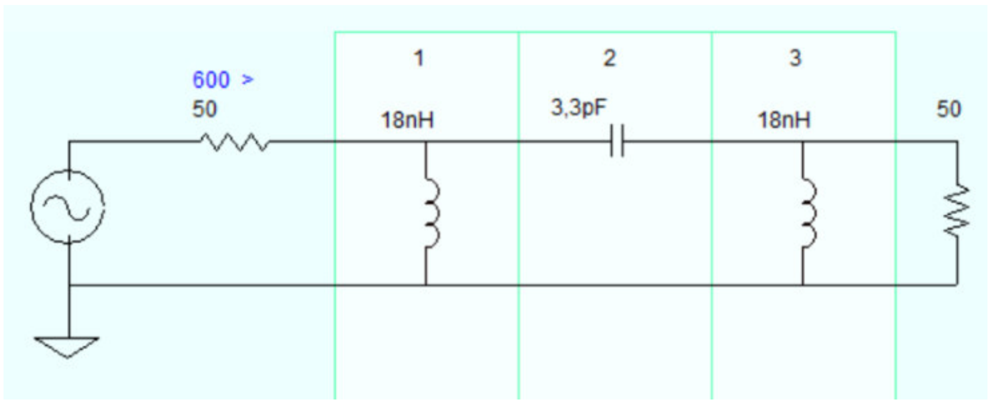
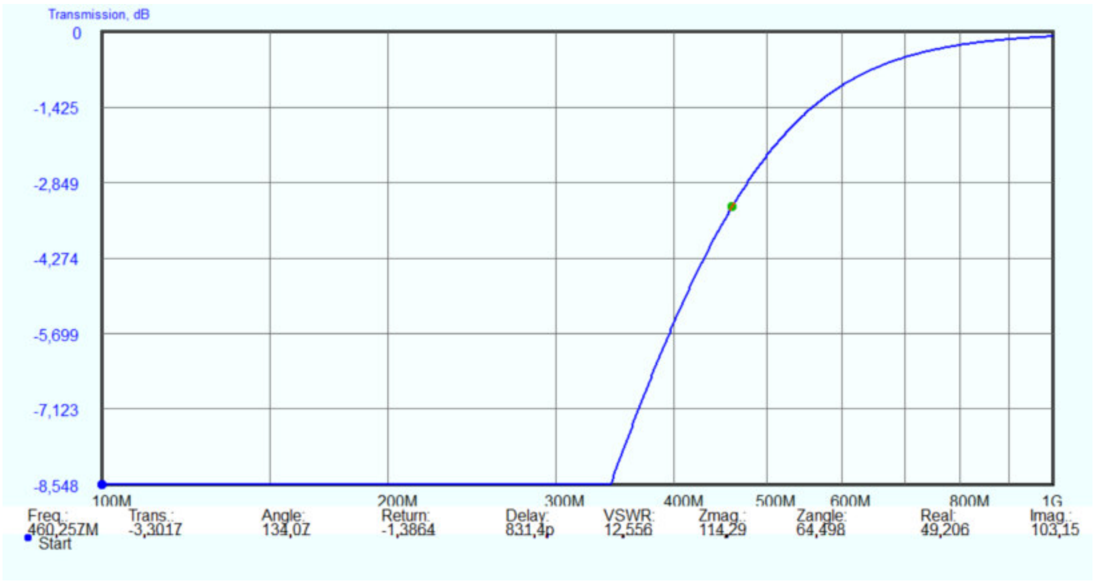


Figure 4.3: The Elsie model for the high-pass filter

Instead of the values obtained by the program for the capacitors and inductances, I ended up choosing values that were closest to them but were available. In Figure 4.3 you can see the values that were set afterwards.



```
Freq.:     Trans.:   Angle:   Return:   Delay:   VSWR:   Zmag.:   Zangle:   Real:    Imag.:
460,257M   -3,3017   134,07   -1,3864   831,4p   12,556  114,29   64,498    49,206   103,15
```

Figure 4.4: The result of the simulation

It can be seen in Figure 4.4 that the values chosen for capacitances and inductances are suitable, because the cut-off frequency is at about -3.3 dB. The transmission band attenuates approximately 8.5 dB.

The filter built in this model was not needed in the end, because I used the BFTC-618+ band-pass filter [5] in the final circuit. This type filters the 460-776 MHz band. Due to its tiny packaging it saves space in the PCB layout and minimizes parasitic effects. It was not a significant aspect in my case, but the filter also shows a reliable performance in tough environments such as high humidity and extreme temperature.

4.2 Division of the reference channel

The signal from the reference channel is first split into two by a resistive divider [8], and then further divided into 4-4 parts by two non-resistive SCA-4-10+ power splitters. One output of the SCA-4-10+ power splitter continues as the reference channel output, and the remaining 7 are sent to the mixer blocks. The resistive power divider is made out of three resistors with a value of 50 Ω, taking into account the 50 Ω input impedance.

The SCA-4-10+ is a 4-way power splitter with 1 sum port and 4 output ports. It operates in the 5 - 1000 MHz frequency band. [6]

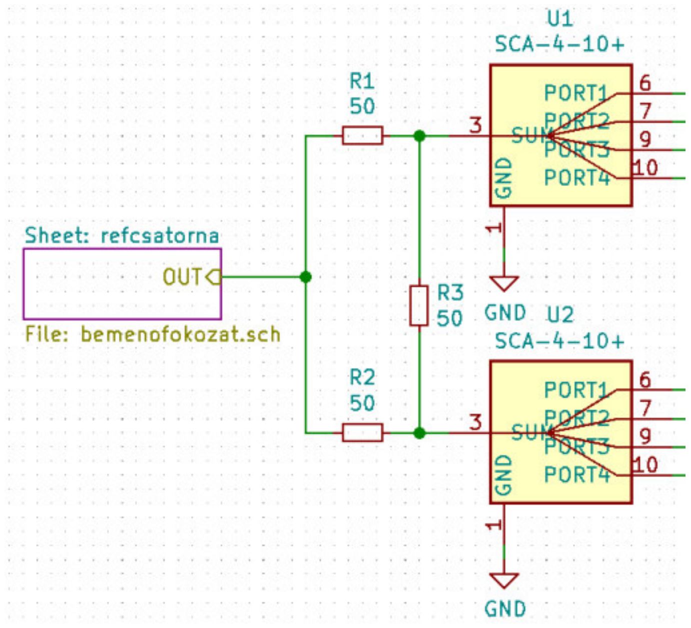


Figure 4.5: The schematic of the reference channel's division

4.3 Structure of the complex mixer

The complex multiplication is implemented as follows.

One input of the complex mixer block is the reference signal. After amplification, it is passed to a polyphase filter which maintains a 90° phase shift between its two outputs. Then the analog multiplication is performed and the I and Q signals are summed at the end.

With the DC values from the controlled DC source, we can adjust our signal to any amplitude and phase by the help of the mixers.

The other input of the block is the signal from one of the surveillance channels, from which the reference signal is extracted at the end of the block by a counter-phase summation.

Thus the output of the block is the signal of the reference signal-free surveillance channel.

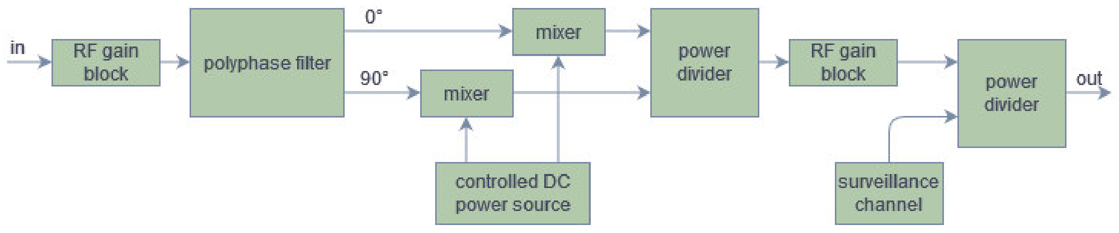


Figure 4.6: Block diagram of the complex multiplier

4.3.1 Polyphase filter

Using the program called LTspice, I created a simulation for an RC polyphase filter.

Since, from the aspect of the components, I did not need a very large bandwidth for the DVB-T frequency range, I just had to design with two stages. I used a method of implementation where the values of the capacitors are the same all along and only the values of the resistors differ.

I calculated with the lower and upper limits of the frequency band, i.e. 460 and 710 MHz. Using these, the required capacitor and resistor values can be obtained from the RC multiplications in the time constant. This way, R was chosen to be 15 Ω and C to be 13 pF.

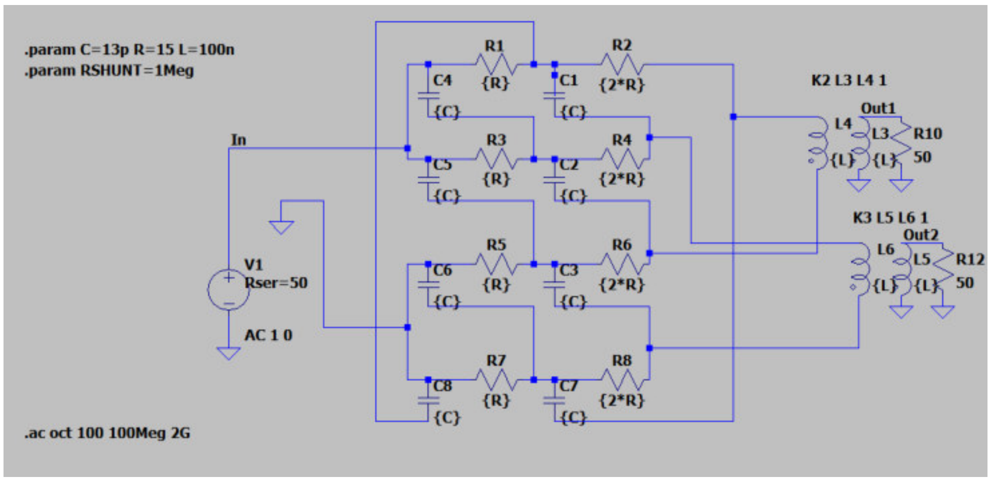


Figure 4.7: LTspice model for the polyphase filter

The aim was to achieve a 90° phase difference between the two outputs and moderate attenuation. This is shown in Figure 4.8, which illustrates the result of the simulation.

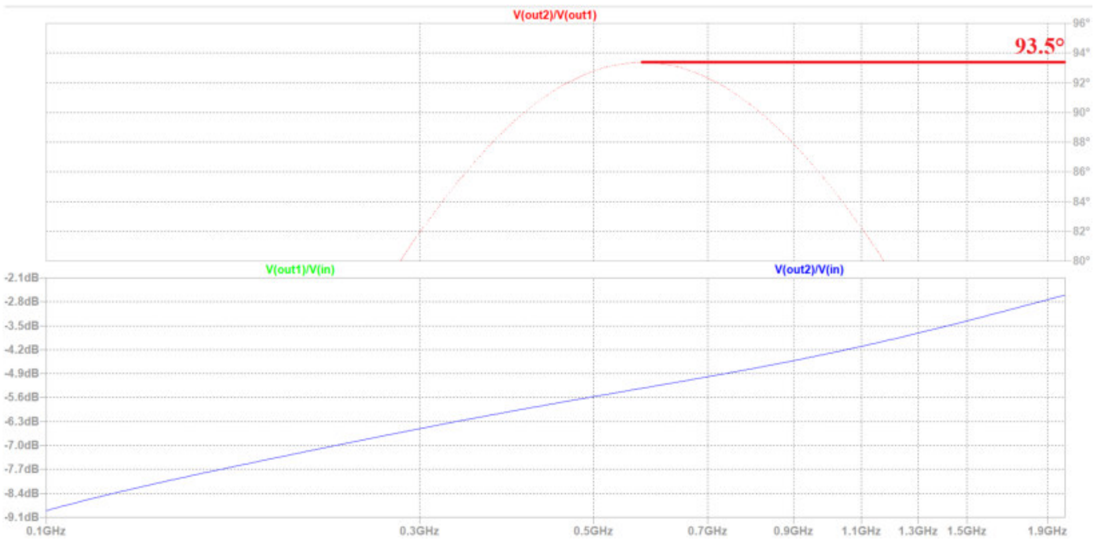


Figure 4.8: The result of the simulation

It was also not necessary to build a polyphase filter, I used the QCN-8+ type [7], available as an integrated circuit. I also added a gain block in front of the filter, which receives the distributed reference signal.

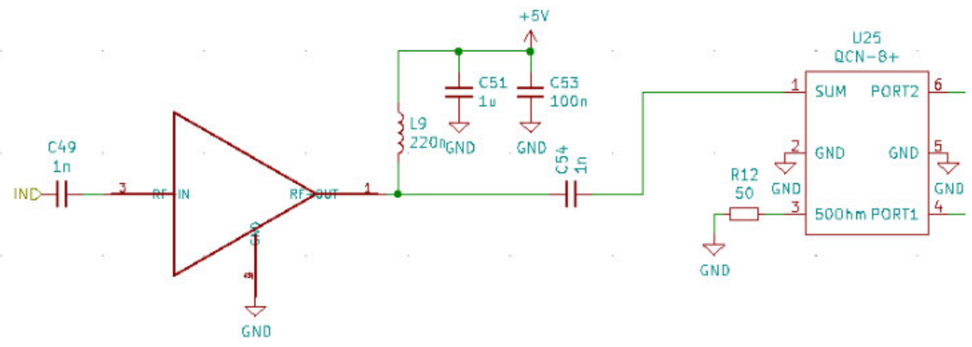


Figure 4.9: The schematic of the polyphase filter with the gain block

4.3.2 The mixer and the counter-phase summation

The multiplication itself is carried out by SYM-12+ mixers with diode ring construction. [9] An important criterion for the selection of this mixer was that it should have a port which supports DC current as input, as this is what we need to implement the multiplication.

The I and Q signals at the output of the mixers are fed into a resistive splitter circuit. An RF amplifier is then required, which is identical to the circuitry used previously.

At the end of the signal path, a resistive power combiner is also used to perform the counter-phase summation, so that the reference signal is already extracted from the output of the unit.

Figure 4.10 shows the end of the complex mixer block.

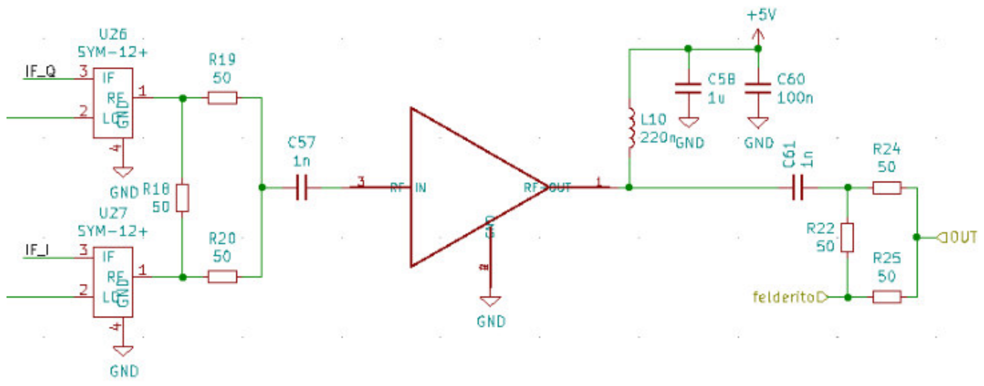


Figure 4.10: The schematic of the mixers and the summation

4.3.3 The voltage-controlled current-source

In the signal suppressor previously built in the lab, the control signals for the multipliers were generated by a voltage-driven voltage generator. This had the disadvantage of reducing the dynamic range. Diodes are current-controlled devices, so it is better to control the current rather than the voltage for greater dynamics. Since the multipliers are diode-ring mixers, when voltage is applied, they open abruptly, reducing the dynamic range.

Figure 4.11 illustrates a diode characteristic for a better understanding. [10]

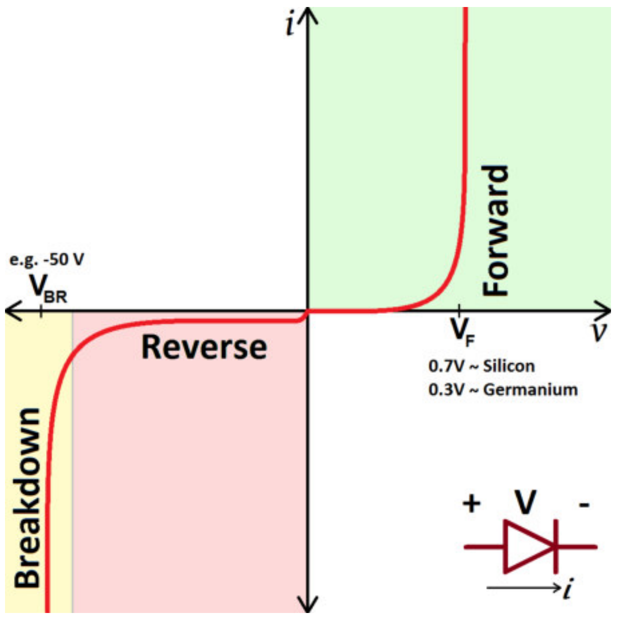


Figure 4.11: Diode characteristic

The structure of the mixer can be seen in Figure 4.12, which is driven by the current-source.

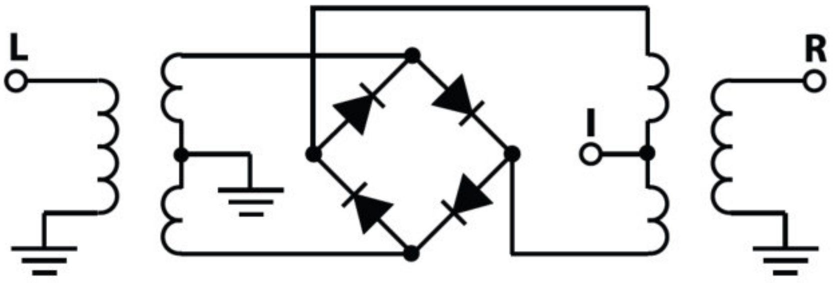


Figure 4.12: Structure of the mixer

This motivated to switch to a current generator, for which I created a simulation in LTspice software. This is implemented by a bidirectional operational amplifier circuit with differential input and current feedback [11]. The role of bidirectionality is that negative current values are required to control the multipliers. In the schematic below, the element driven by the current generator is an antiparallel-connected diode pair, since this is what is used in mixers.

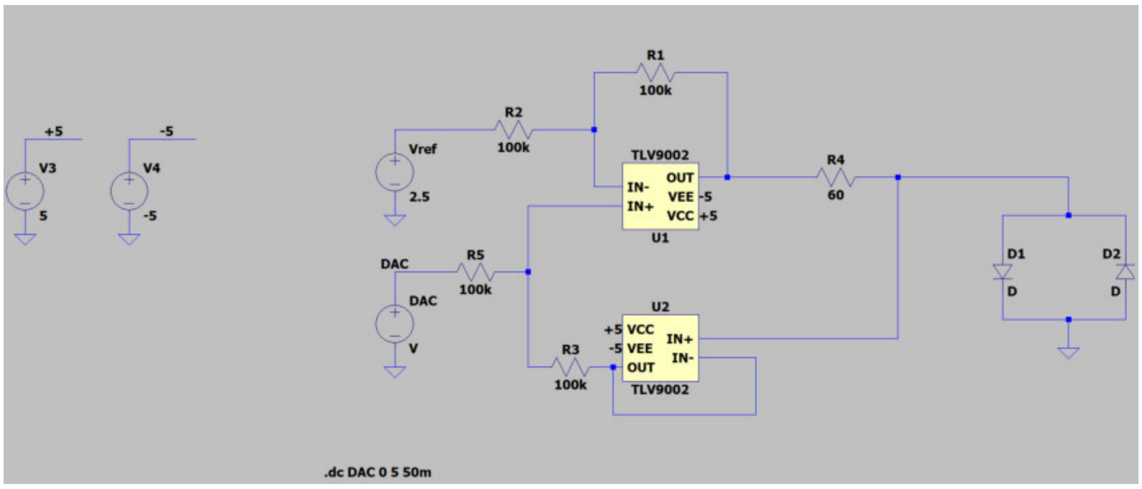


Figure 4.13: LTspice model for the voltage-controlled current-source

For the simulation, I varied the input voltage linearly between 0-5 V. This will be the signal produced by the DAC. The goal was a linear input voltage and output current characteristic, which was achieved based on the simulation results. The circuit was sized for the output current range +-40 mA. The jump seen in the characteristics at 2.5 V is due to the output of the digital-to-analog converter, since the offset voltage at the input of the operational amplifier is set here, which is the transition 0 at the output of the amplifier.

The output resistor, R4, shown in Figure 4.13, was eventually changed to 600 Ω. This reduced the range to +-4 mA.

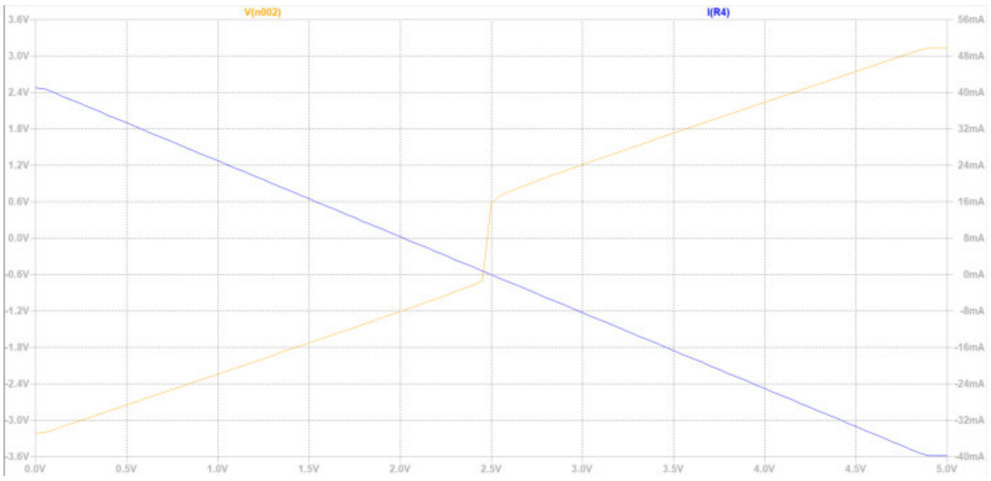


Figure 4.14: The result of the simulation

The circuit is implemented with TLV272CDG4 operational amplifiers, which require dual supply voltage and can operate rail-to-rail, utilizing the full voltage range, which is 10 V.

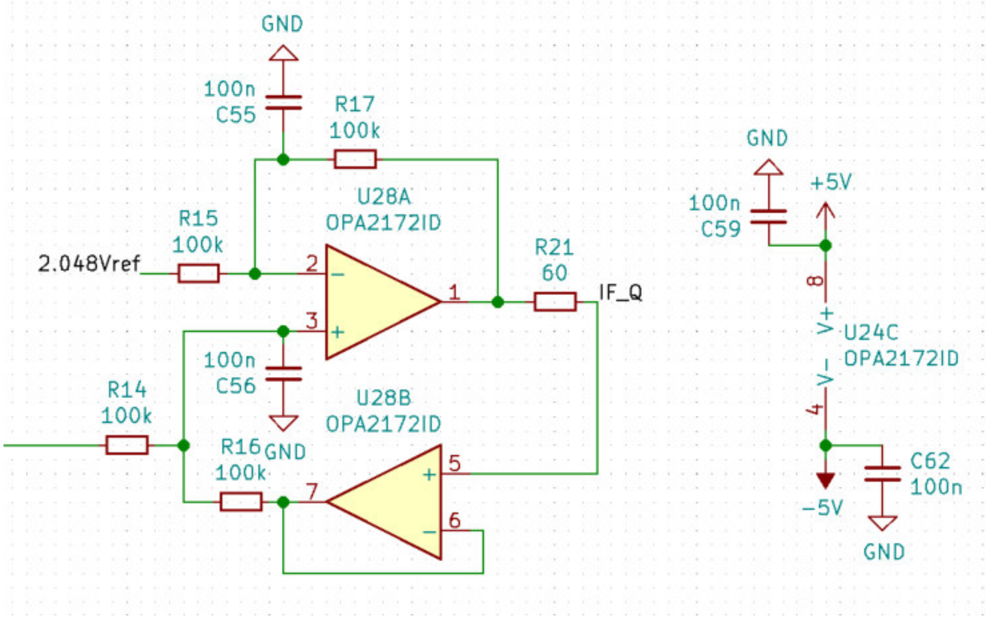


Figure 4.15: The schematic of the voltage-controlled current-source

4.3.4 Digital-to-analog converters

I chose a 12-bit DAC of type MCP4822-E/MS as the digital-to-analog converter controlled by the microcontroller. It has an internal reference voltage of 2.048 V and can be set to 1 or 2 times gain, i.e. it can output a voltage range of 0-2.048 V or 0-4.096 V. The device will need the double gain to use up the range provided by the supply voltage.

Each surveillance channel has a separate DAC and is connected to the microcontroller via an SPI interface.

The inputs of the DAC provided by the microcontroller are the followings. SCK is the clock signal, MOSI is the data input and CS is the chip select signal of the SPI interface. The analog outputs of the converters are fed to the inputs of the current generators, and from here the value to be set is fed to the mixers.

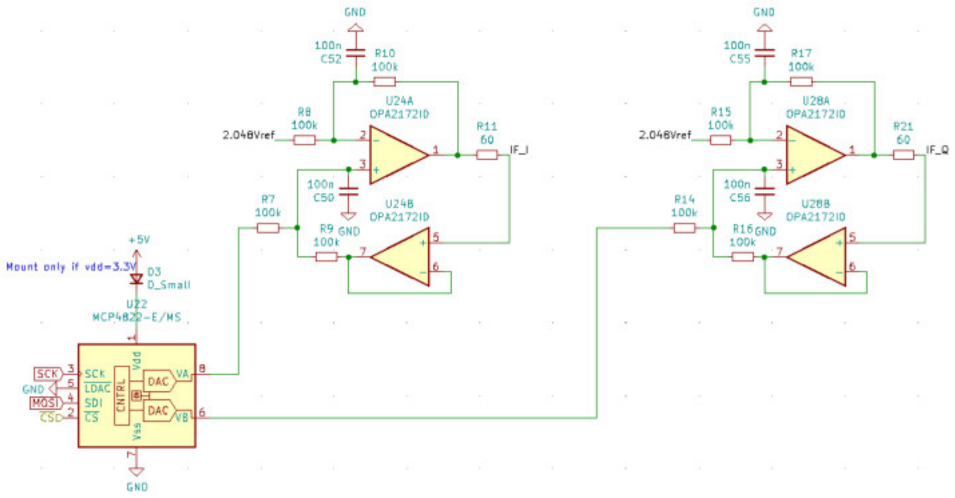


Figure 4.16: The schematic of the DAC

4.3.5 Voltage reference for the digital-to-analog converters

The output of the DAC is fed to the input of an operational amplifier scheme with differential input. We also know that the output of the converter is in the range of 0-4096 mV. Thus, if the reference voltage is chosen to be at the half of this range, which is 2048 mV, then symmetrically, negative and positive values can be applied to the mixers around this value, which is what we need for the multiplication.

The reference voltage is generated by the reference voltage IC type LM4040CIM3-2.0/NOPB. I designed the schematic according to the recommendation in the datasheet [13].

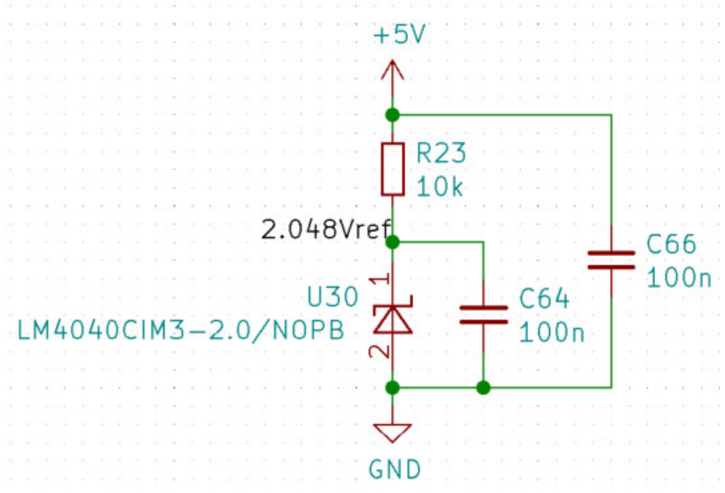


Figure 4.17: Generating the reference voltage

4.4 The voltage-inverting switching regulator

We also need a circuit that provides -5 V, as operational amplifiers need to operate from dual supply voltage to produce negative currents. The device receives the 5 V supply voltage via USB and from this the circuit needs to generate -5 V.

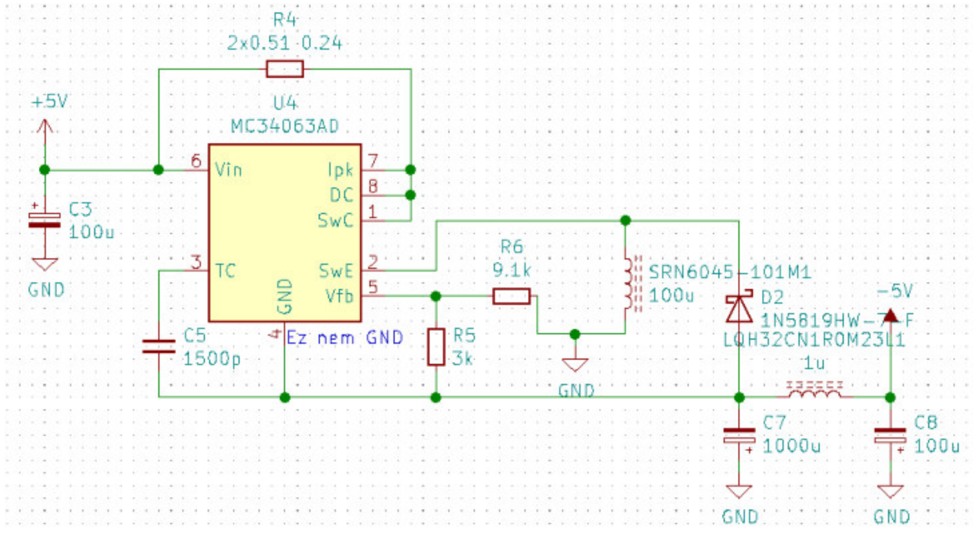


Figure 4.18: Schematic of the voltage-inverting switching regulator

For the implementation of the switching regulator circuit I used the MC34063AD type IC. I produced the schematic according to the datasheet [14].

The IC contains all the primary circuitry needed for building the DC-DC converters. In the earlier stages of the project I designed the circuit for 40 mA on each output of the current-sources, which would mean 80 mA per one surveillance channel and 560 mA for all of the 7. Although, later the output current was reduced by a tenth, the up to 1.5 A provided by the regulator would have been enough.

I adjusted the -5V voltage at the output using resistors R5 and R6 as follows:

$$U_{out} = -1.25(1 + \frac{R6}{R5}) = -5.04V \tag{4.1}$$

4.5 Control unit

The control is implemented by a PIC18F26K22 microcontroller. It controls the seven digital-to-analog converters through SPI interface. I connected the programming pins by the help of to the datasheet [15]. I chose this type because of its low energy consumption, and it is easily accessible and programmable.

A CP2102 [16] type serial interface makes contact between the USB input and the microcontroller.

The circuit also includes a jumper to select the supply of the serial port and the microcontroller from 5 or 3.3 V. In the latter case, as the rest of the device still operates from 5 V, a diode must be connected to each of the digital-to-analog converters to ensure the correct logic levels.

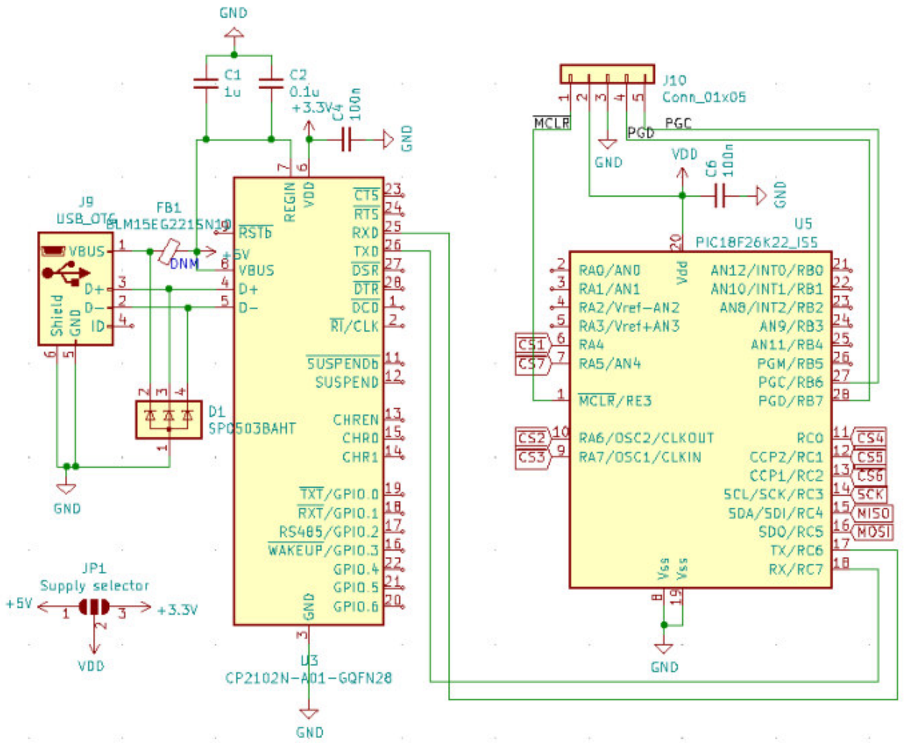


Figure 4.19: The schematic of the USB connector, the USB-UART bridge and the microcontroller

Chapter 5

The prototype

5.1 The PCB layout of the prototype

The design of the prototype's printed circuit board was created using the Pcbnew subroutine of KiCad.

This consists of the signal path of the reference channel and one surveillance channel. It also houses the control unit, the switching regulator and the part generating the control signals.

I designed the circuit for an FR4 carrier. I used an electrical parameter calculator subroutine called PCB Calculator to determine the parameters of the RF signal path wiring. I chose coplanar waveguide with ground plane, and the goal was to achieve a wave impedance of 50 Ω. Thus, I obtained a width of 0.8 mm for the RF wires and 0.25 mm clearance. During the wiring process, I tried to minimize the sharp turns of the wires.

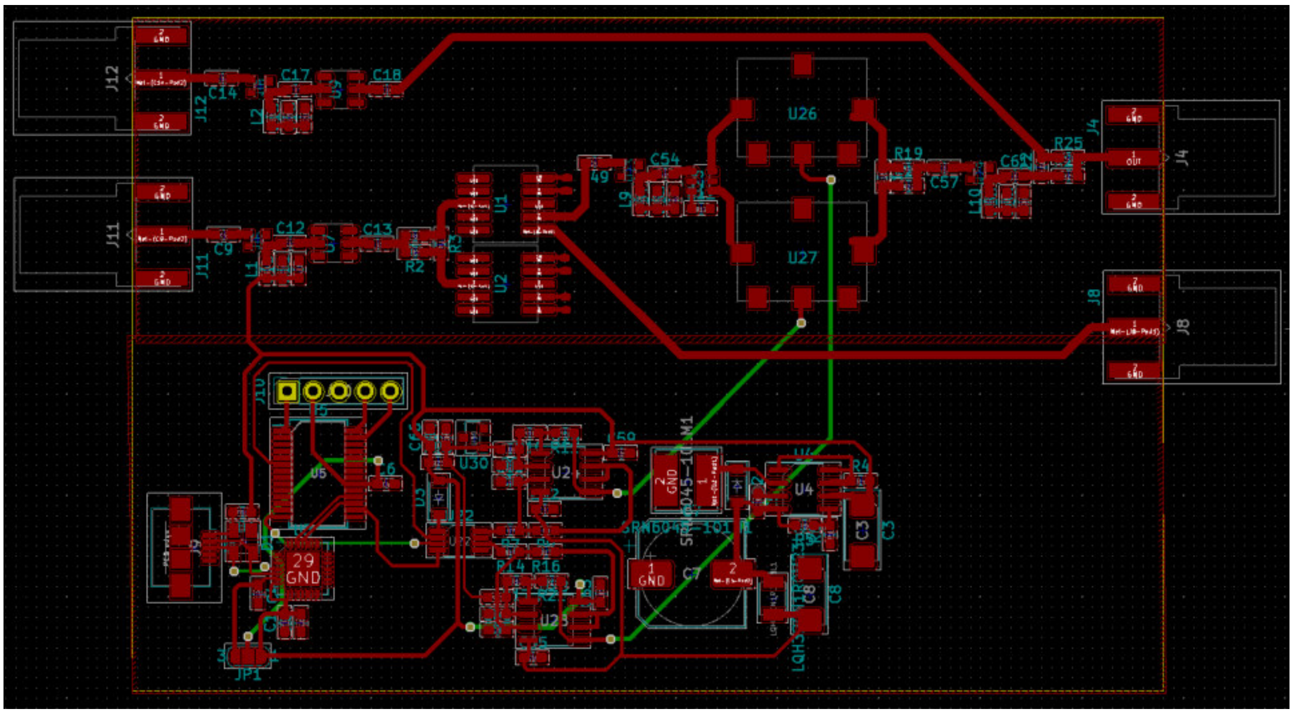


Figure 5.1: The PCB layout of the prototype

5.2 The prototype PCB containing the RF signal paths

The prototype's printed circuit board containing the RF signal paths was prepared in the laboratory. This was sufficient for testing the circuit and the components, but I will not be producing the final version myself. The circuit after toner transfer and pcb etching is shown in Figure 5.2.

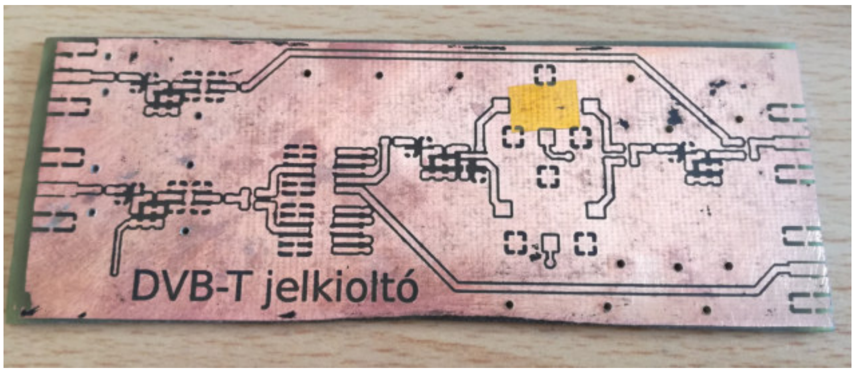


Figure 5.2: Prototype - PCB containing the RF signal paths

5.2.1 Soldering the components

After the printed wiring board was ready, the components were fitted, which I also did in the laboratory. On the PCB shown in the figure 5.3, all the existing pieces are already in place, but measurements were taken in the meantime to test the different parts.

The power splitters to distribute the reference channel had not yet arrived, so I simply rewired the single surveillance channel on the prototype to the appropriate location, while the reference channel was routed out with a piece of coaxial cable.

To adjust the DC inputs of the mixers the other part of the prototype will be needed, so I put two wires on the two IF input pins to make it easy to connect the circuits later.

The final version is a 2-layer PCB, but for ease of implementation, the prototypes are single-sided boards. This meant that I had to relocate the wiring on the bottom layer to the top layer, so I also provided the power line for the amplifiers by rewiring.

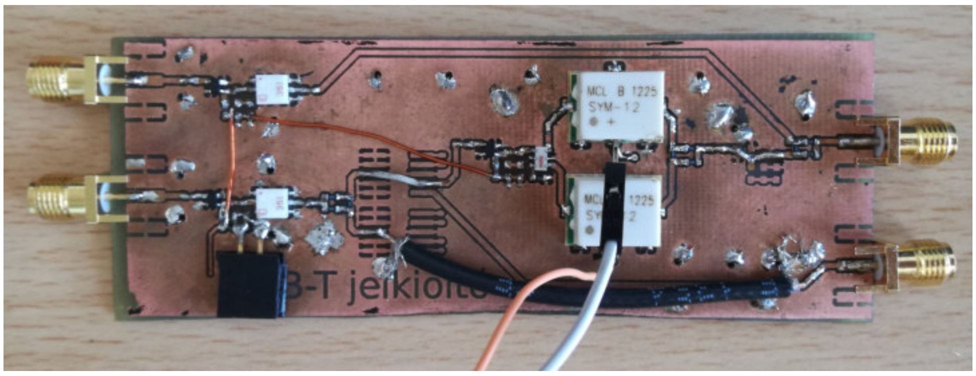


Figure 5.3: Prototype - RF PCB

5.2.2 Measurements

I took measurements to test the components and sub-assemblies, some of which I will show further on. The measurements were taken while the various components were still being installed, so these pictures do not show the complete prototype.

The first element I measured was the mixer. Here I connected a software radio between the input and output of the mixer, then I performed a current-generator measurement on the DC input of the mixer, as this is how the final version will work.

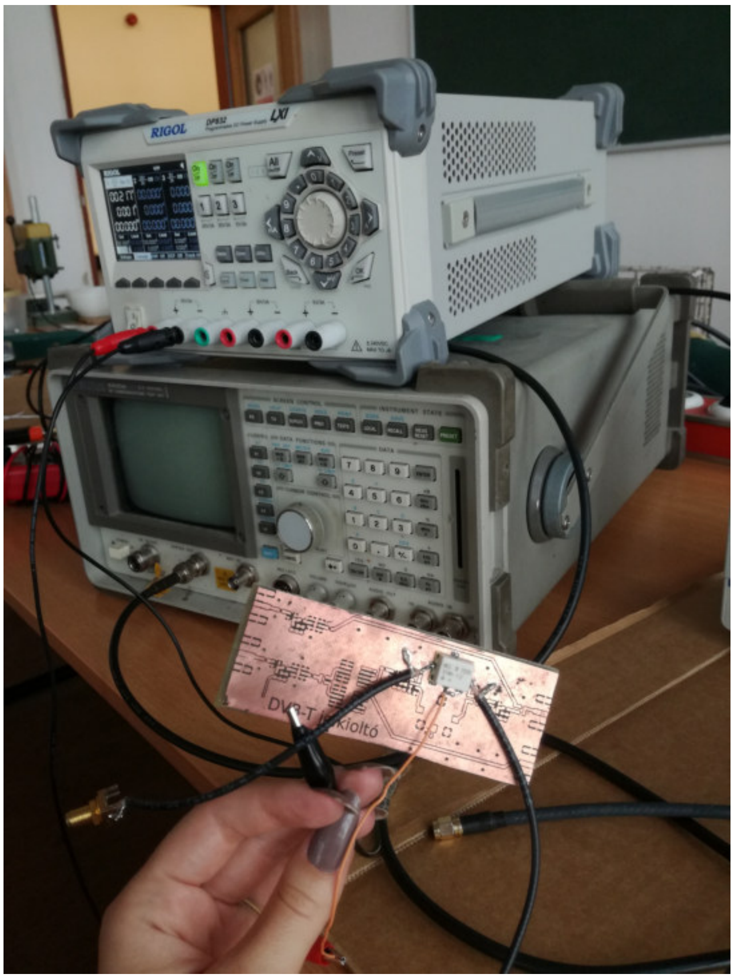


Figure 5.4: Measurement of the mixer

Using the software radio, I examined the result on a constellation diagram and plotted the amplitude versus current on a logarithmic scale.

The result shows that a scalable 25dB dynamic range has been achieved.

It turned out that the mixer goes into saturation around 3-4 mA, so it is necessary to set the current limit to this value instead of the previously simulated 40 mA. If the current limit is too high, the DAC will not be used to its maximum and the resolution will be worse, and if it is too low, it will lose dynamics.

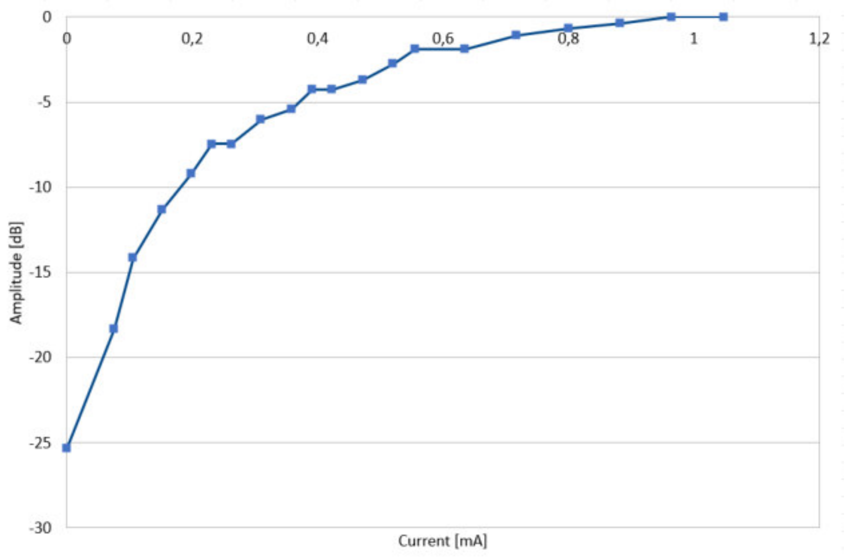


Figure 5.5: The result of the mixer's measurement

The next component I tested was the band-pass filter. For the measurement I used a vector network analyser. The measurement setup is shown in the picture below.

The gain block has already been installed in front of the filter, which is powered by a powerbank via pinsocket. I soldered a piece of coaxial cable to the output of the filter and connected it to the instrument through an SMA connector.

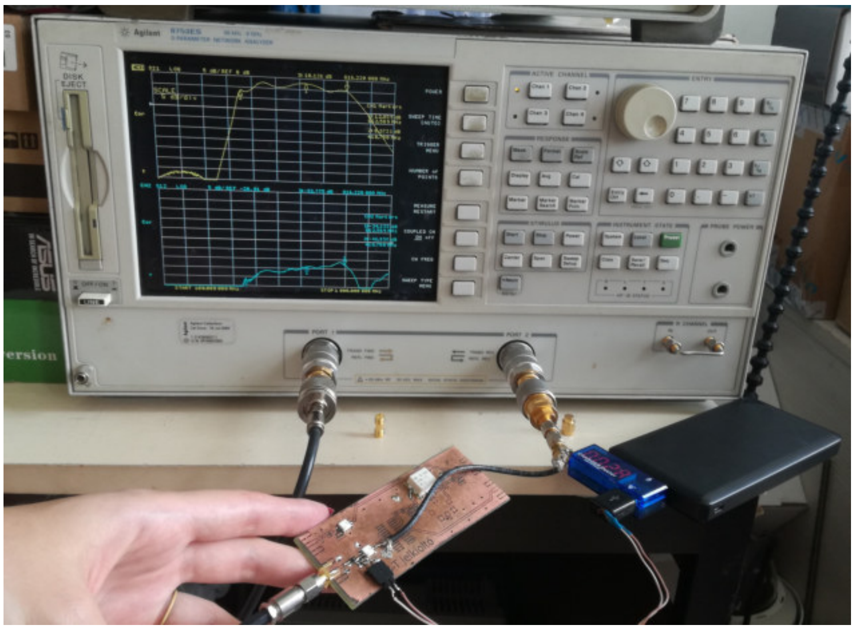


Figure 5.6: Measurement setup for testing the filter

It can be seen in the image of the VNA display, in Figure 5.7, that I have visualised S21 and S12 out of the S parameters. Using the markers on the figure we can see that the filter passes roughly between 420 and 820 MHz as expected, which is sufficient for DVB-T band operation. It can also be read that, since the filter attenuates by about 7 dB after the LNA has been amplified by 20 dB, the value of nearly 13 dB indicated by each marker is also correct.

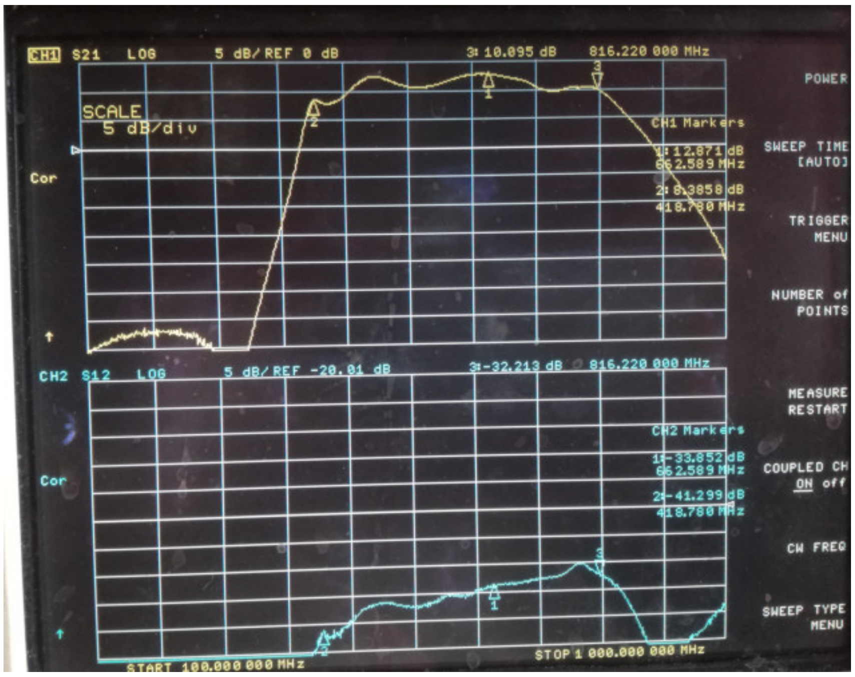


Figure 5.7: S parameters

For comparison, Figure 5.8 illustrates the transmission of the band-pass filter according to the datasheet [5]. The filter operates as expected.

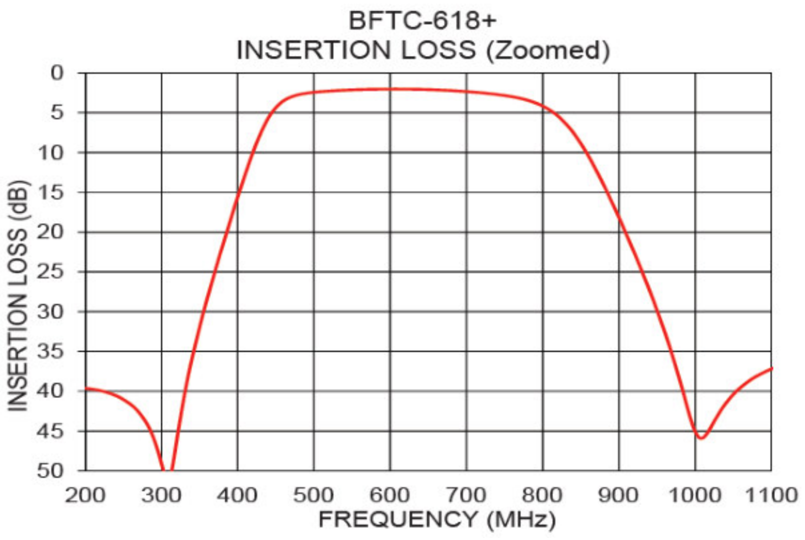


Figure 5.8: Transmission of the filter

After making sure that the different RF components were working correctly, I fitted the remaining components to test their behaviour together. I made the necessary jumper connections to the amplifiers and connected a wire to the DC inputs of the mixers. I connected the latter to a DC power supply and then monitored the changes by switching the power on and off.

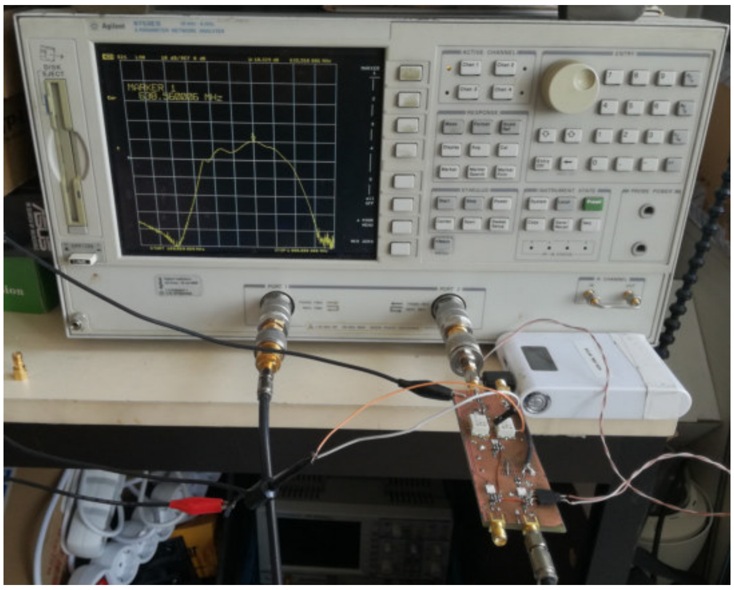


Figure 5.9: Transmission measurement with and without mixers

The next two pictures, Figure 5.10 and Figure 5.10, show the change of about 25 dB, when the power supply of the mixers' DC inputs is switched on and off.

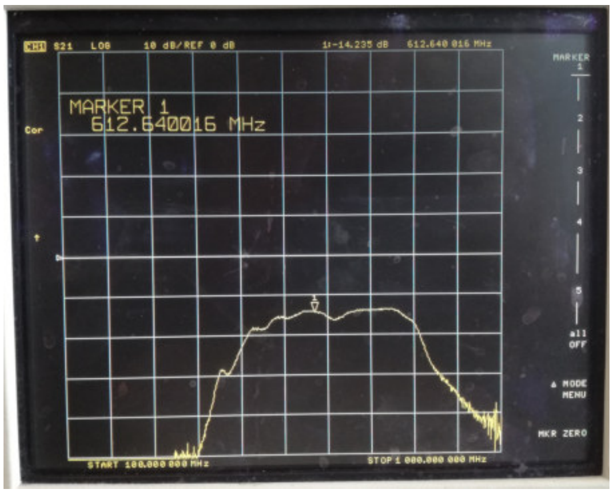


Figure 5.10: Measurement with mixers

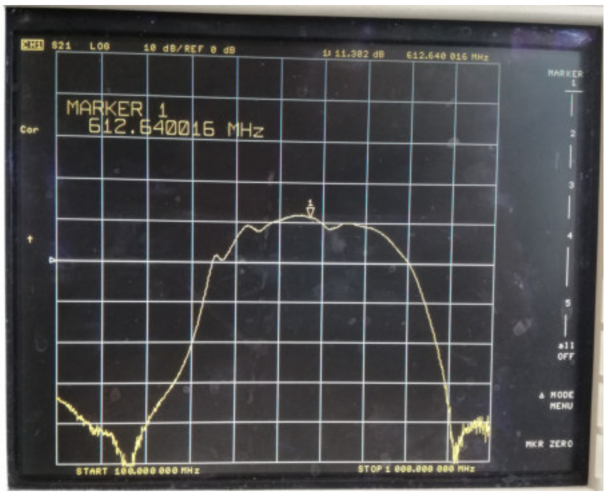


Figure 5.11: Measurement without mixers

5.3 The prototype PCB containing the DC parts

The prototype DC circuit has the following parts.

The PIC18F26K22 type microcontroller, which controls the digital-to-analog converter via SPI interface.

The digital-to-analog converter, whose output is fed to the current generator and provides the DC input signal of the mixers.

The circuit is powered from a USB input and a CP2102 type USB-UART serial interface is placed between the USB and the microcontroller.

I also made the printed circuit board of the prototype, containing the control unit, the DAC and the voltage-inverting switching regulator in the laboratory.

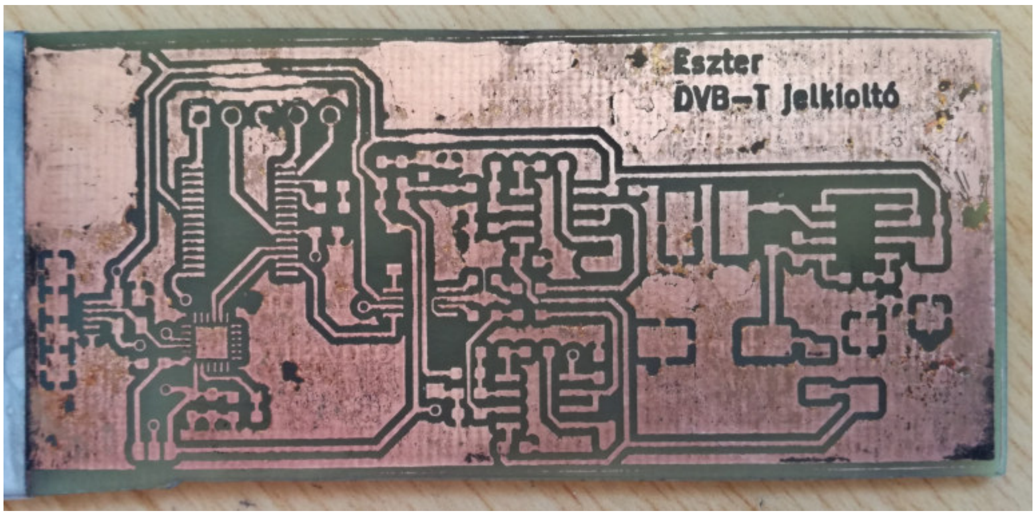


Figure 5.12: Prototype - printed circuit board of the DC parts

5.3.1 Soldering the components

This time the parts were populated in the laboratory as well.

The components used for the prototype were chosen partly from previously ordered elements and partly from already existing ones in the lab.

The figure below shows the state where the mistakes have already been corrected. One of these errors was due to the ground plane, as some ground areas were isolated from each other. I solved this problem by connecting the different parts with wires.

Another error was that I had not soldered one of the ground pads of the USB-UART bridge, so it did not work properly. The solution to this problem was to connect the tx and rx pins of the microcontroller to a pin header and use another serial interface to adjust the I and Q values on the channel, while still supplying power from the USB input.

To program the microcontroller, I led the programming pins out through a pin header. The operation of the program will be explained in detail later. During the design process, a LED was added to one of the pins, so I only checked the operation with the first code, which turned out to be correct.

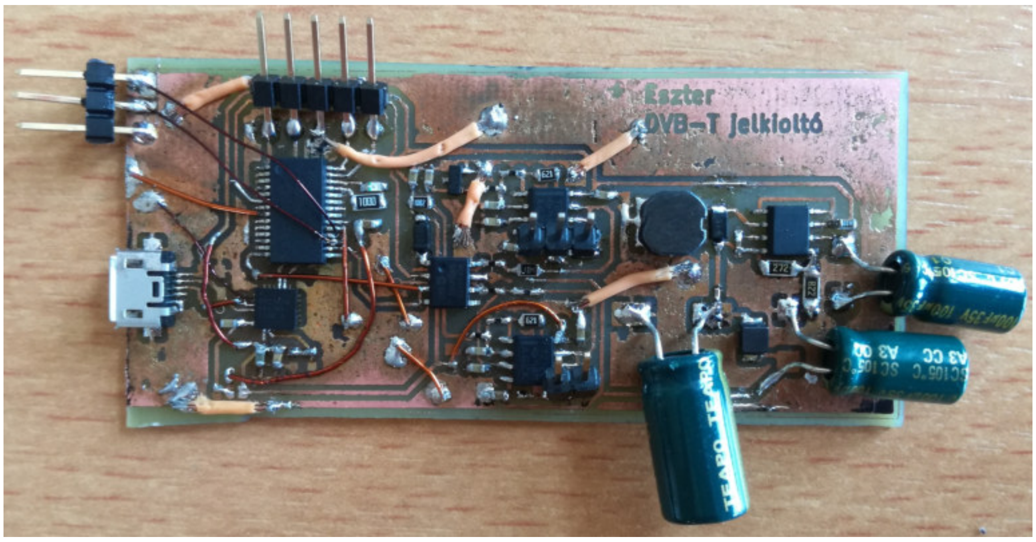


Figure 5.13: Prototype - PCB of the DC parts

5.3.2 Measurements

Once the components were in place, I wrote a code for the operation of the microcontroller, which I will explain later. Then I examined what current values appear at the outputs of the current generators, with different data provided by the digital-to-analog converter. The figure below even shows the illuminating test LED.

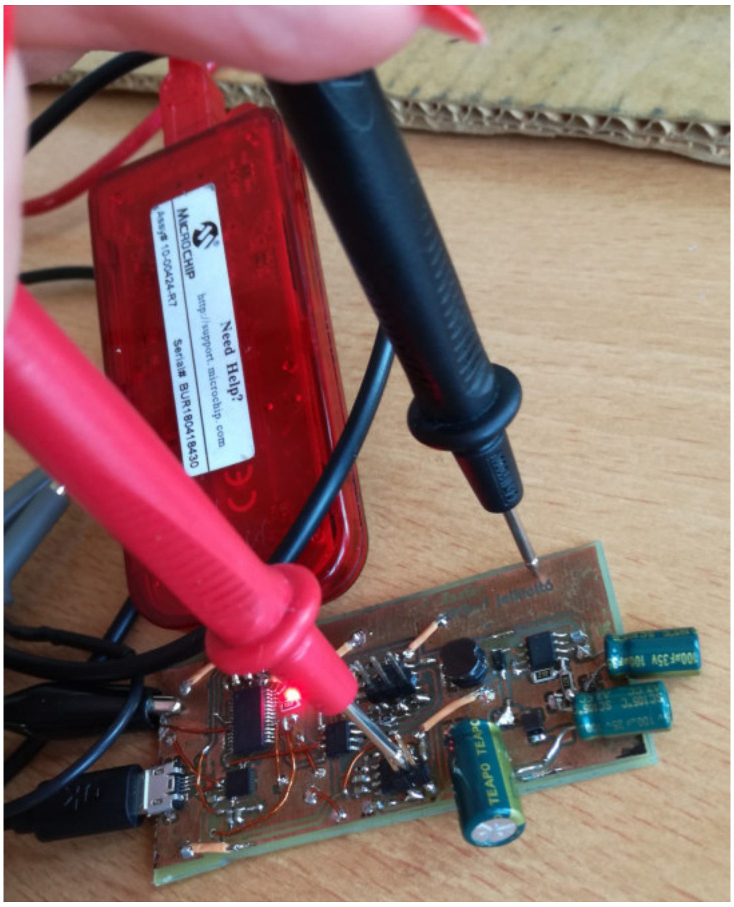


Figure 5.14: Measurement of the current generator's output

The measurement was done using a multimeter. The following two figures show two different current values. The current generators worked as expected, producing both positive and negative values. They were able to step through the range provided by the DAC with a step size of about 1.5 uA.

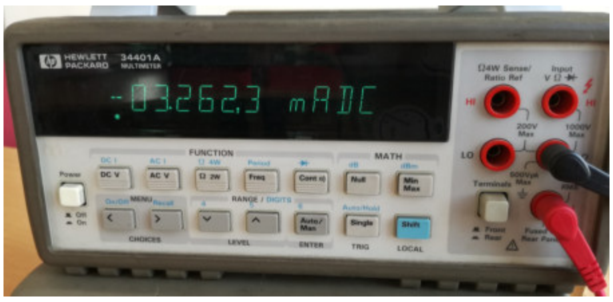


Figure 5.15: Measurement of the current generator - lower limit

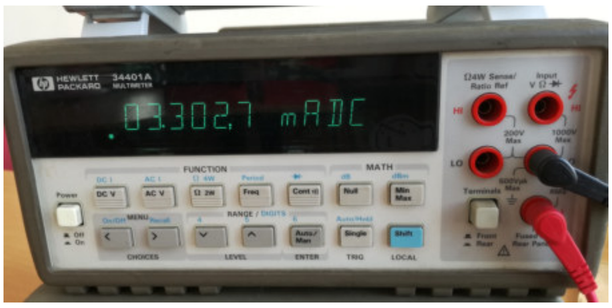


Figure 5.16: Measurement of the current generator - upper limit

I have also plotted the result of the measurement, showing the current values for the different DAC outputs, which vary between 3.28 and -3.28 mA.

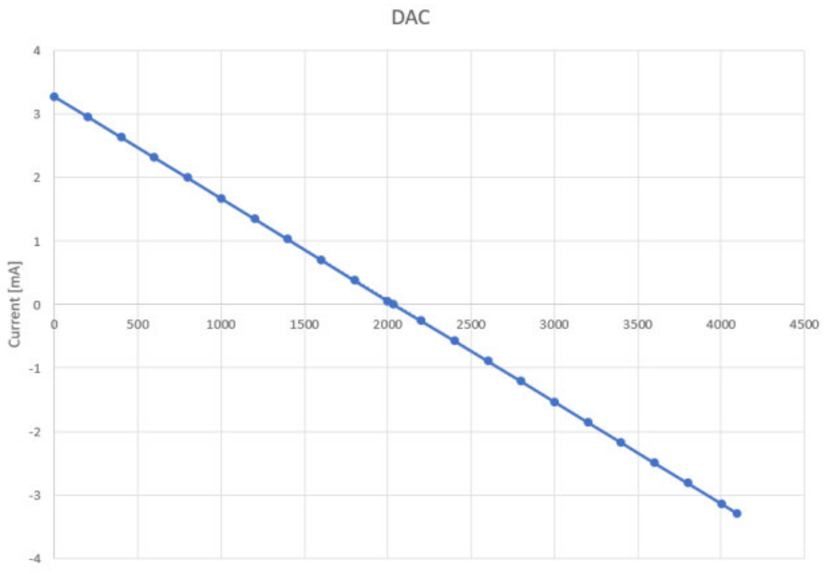


Figure 5.17: DAC - current characteristic

5.4 Joining the two PCBs together

I connected the two PCBs via a pin header that was placed for the programmer. I also soldered a pin header to the output of the generators so I could connect them to the DC inputs of the mixers. The tx and rx pins of the microcontroller were connected to a USB-UART bridge to receive the data to be set. I also had the help of a software radio, which I could use to examine the result of changing the I and Q values on a constellation diagram. Unfortunately no picture was taken, but the operation proved to be correct.

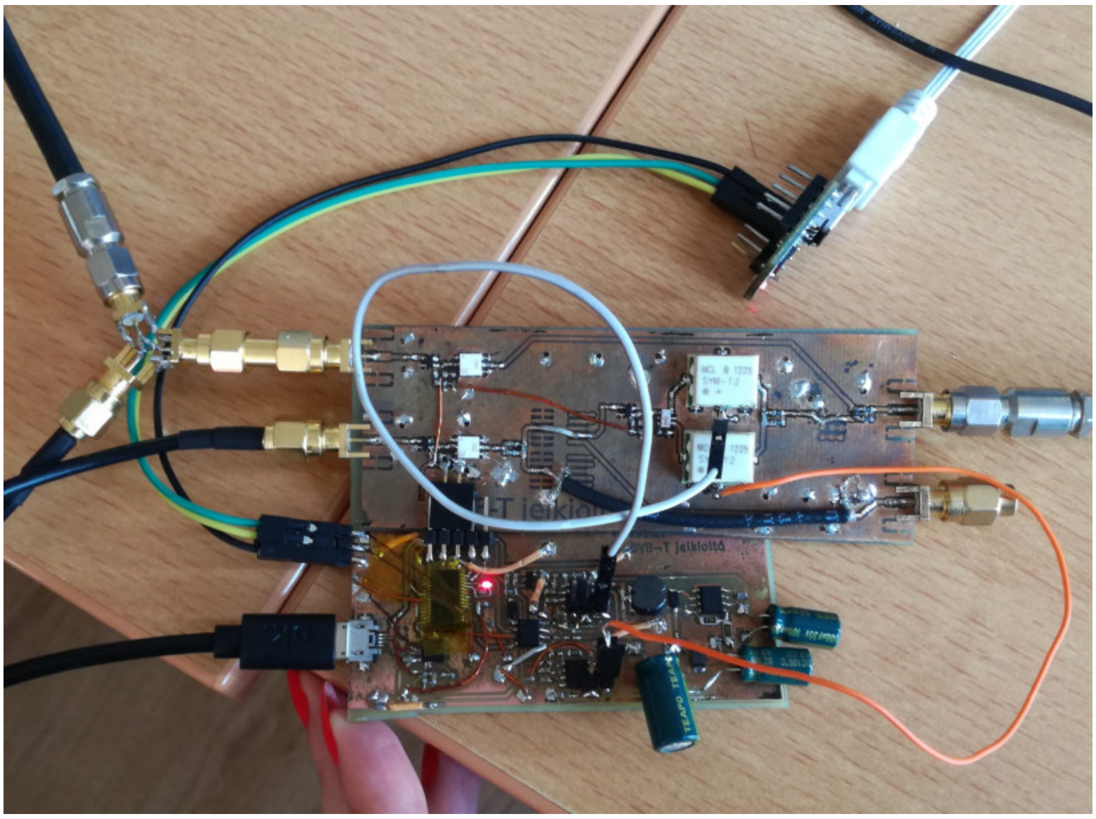


Figure 5.18: The two PCBs

Chapter 6

The final version

As with the prototype, the final signal suppressior is made up of several parts. The different channels are placed on a motherboard, which carries the microcontroller's control signals and power lines. The reference channel is distributed here. All the wiring necessary for the operation of the channels is fed to the circuits via connectors.

6.1 Designing the printed circuit board

The final version of the printed circuit boards was created with the help of literature [17].

As before, I designed it for an FR4 carrier, using the RF wire parameters calculated for the prototype. I designed 2-layered boards with ground plane.

I avoided having to run the RF line through vias to the backside and having another signal or power line running alongside it. The former to create reflections, radiation and losses, the latter to avoid causing undesirable effects by electrical and/or magnetic coupling in critical signals.

I placed vias next to the ground pins of the components in the signal path, reducing return current impedance. To avoid further high frequency problems, vias were placed along the RF lines. The high-frequency lines are the same width everywhere and the width of the component pads is identical, and these two widths are similar in most places, preventing parasitic effects. There are also scattered vias in unused PCB areas to keep the GND impedance low and reduce EMC issues.

The PCB layouts of the motherboard and one of the channels are shown below.

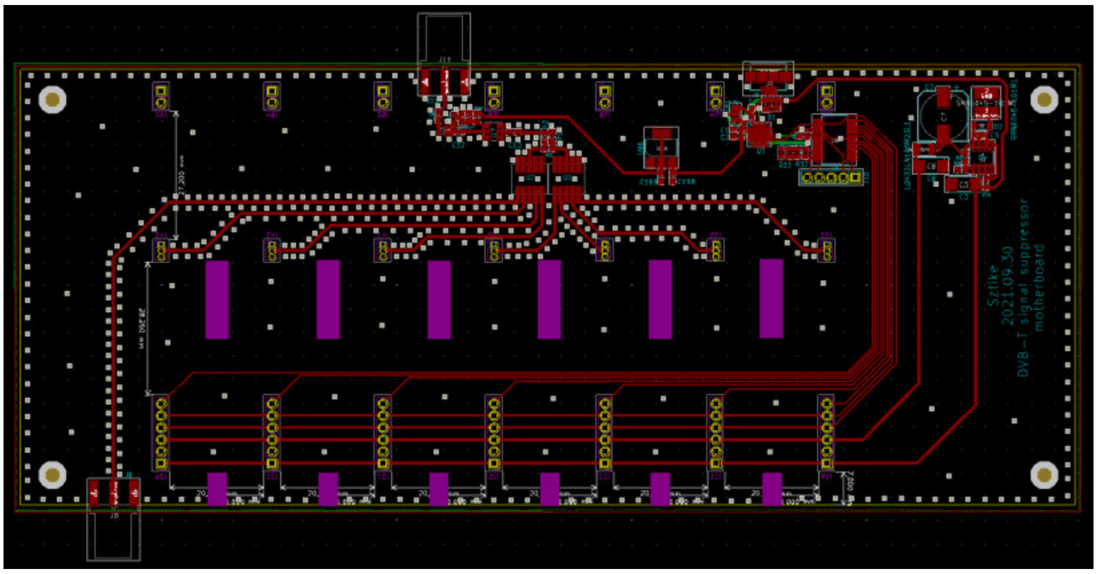


Figure 6.1: The PCB layout of the motherboard

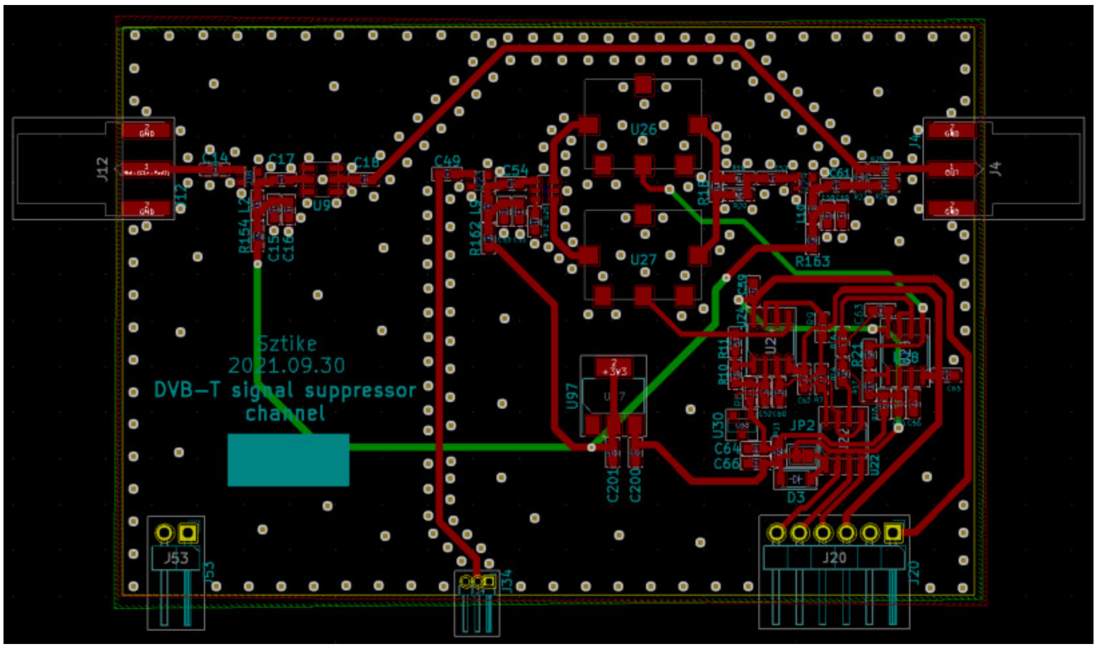


Figure 6.2: The PCB layout of the channel

6.2 Motherboard

On the motherboard, the reference channel is divided, which is passed on to the channels via a pin header. The voltage-inverting switching regulator, the microcontroller and the serial interface are also located here.

The cut-outs between the pin headers of the channels will later allow the possibility of adding shielding between the circuits. Around these, a contact area has been left out of the solder mask to ensure more stable insertion of any plates between the channels by soldering.

The holes in the corners of the motherboard will be used to later insert the circuit board into a box.

The implementation of the motherboard is shown in Figure 6.3.

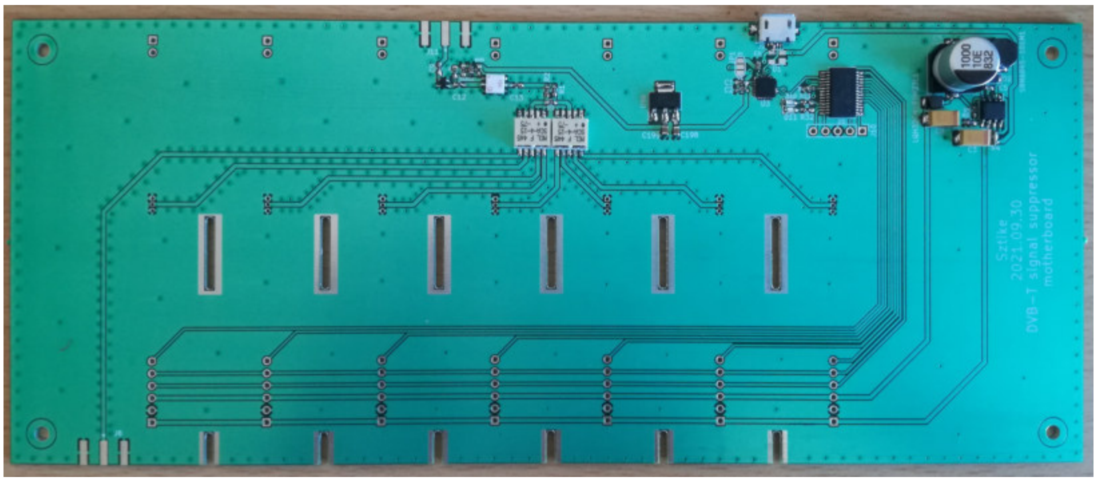


Figure 6.3: The motherboard

6.3 Channel

Each channel is located on a separate PCB and is connected to the motherboard, orthogonal to it, by the help of pin headers.

This circuit includes the surveillance channel signal path, the complex multiplication unit and the DAC together with the current generators.

During the semester, one channel circuit was completed, as shown in Figure 6.4.

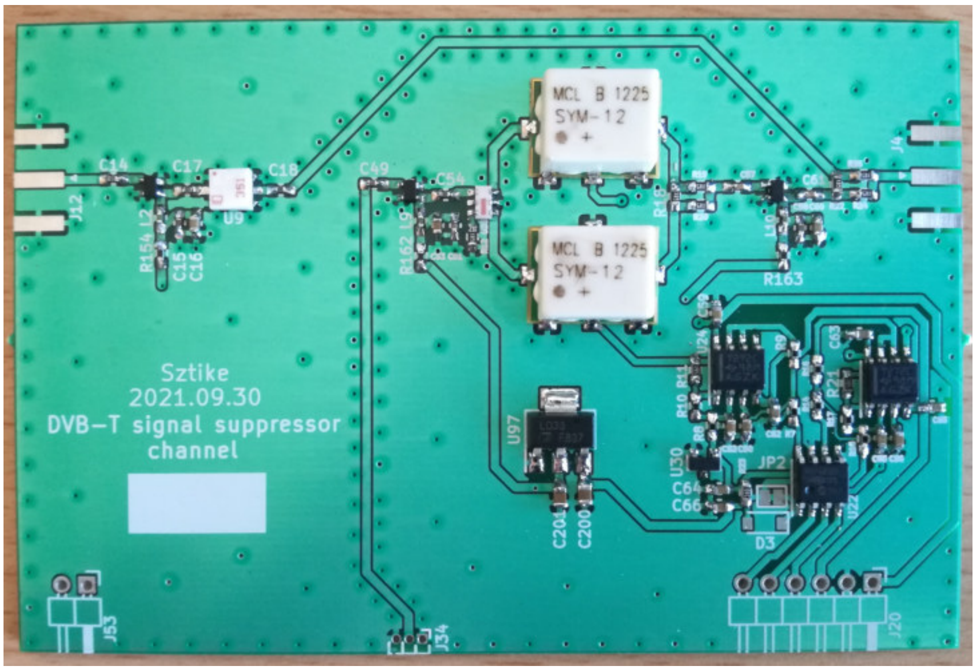


Figure 6.4: The channel

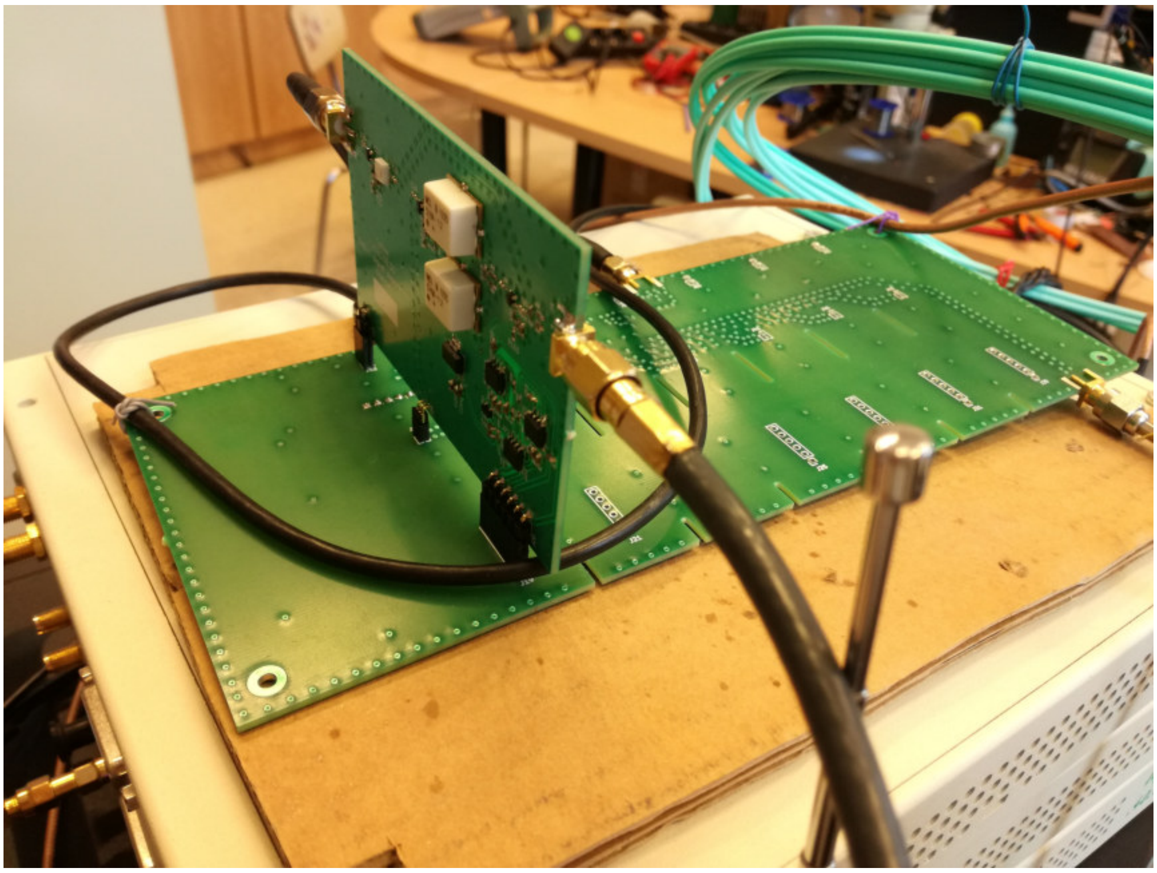


Figure 6.5: The connected PCBs

Chapter 7

The microcontroller's program code

I programmed the microcontroller through a pin header using Pickit3 programmer and communicated with it via USB. I used the software MPLAB to write the program code. To configure the pins I used the MPLAB Code Configurator subroutine.

The goal was to write a code that would implement the ability to set the I and Q values for the different channels from a PC. To do this, the code expects a string in which it is possible to specify which channel, I or Q signal and which value is to be set. Since the signal suppressor will have 7 channels and they are numbered from 0, a number between 0-6 can be chosen as the first character. Then the I or Q is set, followed by a number between 0 and 4095, since the DAC operates in this range.

Example of a possible value: 2q1234

If any of the parameters is set to an incorrect value, the code throws an error message and the entered string will not be adjusted.

DACs can be controlled by 16-bit code words, according to their datasheet [12] . The first four bits set which of their channels should be active and whether to implement single or double amplification. The two channels (A and B) of a DAC will control the I and Q for one of the surveillance channels. DACs sample the data at their input to the rising edge of the clock signal and output it when the enable signal is set to 1. The code snippet below shows the defining of the control signals. These are bit-shifted to the appropriate bit position.

```
#define DAC_A (0<<15)
#define DAC_B (1<<15)
#define GA_1X (1<<13)
#define GA_2X (0<<13)
#define SHDN_ACTIVE (1<<12)
```

The DACSet function prepares the 16-bit code words and sends them to the appropriate location.

It receives the channel and number to be set and sets the control and data bits using the predefined values.

```
void DAC_Set(uint8_t ch, uint16_t value)
{
    uint16_t reg = (ch & 0x01)<<15  | GA_2X | SHDN_ACTIVE | (value & 0x0FFF
    );
    uint8_t dac_h = (uint8_t)((reg & 0xFF00)>>8);
    uint8_t dac_l = (uint8_t)((reg & 0x00FF));
    uint8_t cs = (ch & 0xFE)>>1;
```

It selects the appropriate channel, sets the enable signal low, sends out the 16-bit code word, and then sets the chip select back high. The 16 bits must be sent in two parts due to the SPIExchangeByte function.

```
    switch(cs)
    {
        case 0:
            CS0_SetLow();
            break;
        case 1:
            CS1_SetLow();
            break;
        default:
            printf("Set low error\n");
    }
    SPI1_ExchangeByte(dac_h);
    SPI1_ExchangeByte(dac_l);
    switch(cs)
    {
        case 0:
            CS0_SetHigh();
            break;
        case 1:
            CS1_SetHigh();
            break;
        default:
            printf("Set high error\n");
    }
}
```

At start, the microcontroller sets both I and Q to 2048 on each channel. This is half way up the range of the DAC, so a current of around 0 mA is applied to the IF input of the mixers. It also triggers the flashing of two test LEDs.

The next few lines ensure that appropriate values can be set with a serial port.

The first step is to scan the data. This continues until \n characters are received or until the string reaches the expected maximum length of 6 characters.

```
    len = 0;
    do
    {
            rxData = EUSART1_Read();
            EUSART1_Write(rxData);
            value[len++]=rxData;
    }
    while(rxData!='\n' && len<6);
```

Then the next loop checks whether the first character is a channel number and if so, stores it in a variable. Otherwise, it throws an error message. The subtraction shown when the channel variable is specified is necessary, because by subtracting the character '0' from the incoming character, the difference of the two characters in the ASCII table will give the value of the desired channel.

```
    if(value[0]>='0' && value[0]<='6')
     channel = value[0]-'0';
    else{
        printf("Ilyen csatorna nincs: %c.\n",value[0]);
        break;
      }
```

After deciding whether the given channel exists, we can specify which one to set out of 'i' and 'q'. If 'i' has been chosen, 'iq' is set to 0. If a 'q' has been received, 'iq' is set to 1. These will later be the 'A' and 'B' channels of the DAC. Here again, an error message will be indicated in the case of incorrect values.

```
      if(value[1]=='i')
          iq=0;
      else if(value[1]=='q')
          iq=1;
      else{
          printf("IQ helytelen: %c.\n",value[1]);
          len = 0;
          break;
      }
```

The variables 'ch' and 'val' will be obtained by the DACSet function. The former contains the channel and IQ value, the latter the number to set.

```
      ch= channel<<1 | iq;
      val = atoi(value+2);
```

Finally, it should be decided whether the given number is definitely within the range provided by the DAC, i.e. not greater than 4095. If this is also correct, the variables are forwarded to the DACSET function and the desired parameters are set.

```
      if(val<=4095){
          DAC_Set(ch, val);
          printf("OK\n");
          break;
      }
      else{
      printf("DAC ertek tul nagy: %d\r\n",val);
      len = 0;
      break;
      }
```

The appendix contains the full programme code.

Chapter 8

The calibration algorithm

The measurements are started by a bash script with the character 1 as the trigger from a fifo. Once the measurement is done and the result is saved to a file, the Python code for the calibration is executed. The bash script gets the 1s from the Python code. The flowchart below shows the process.

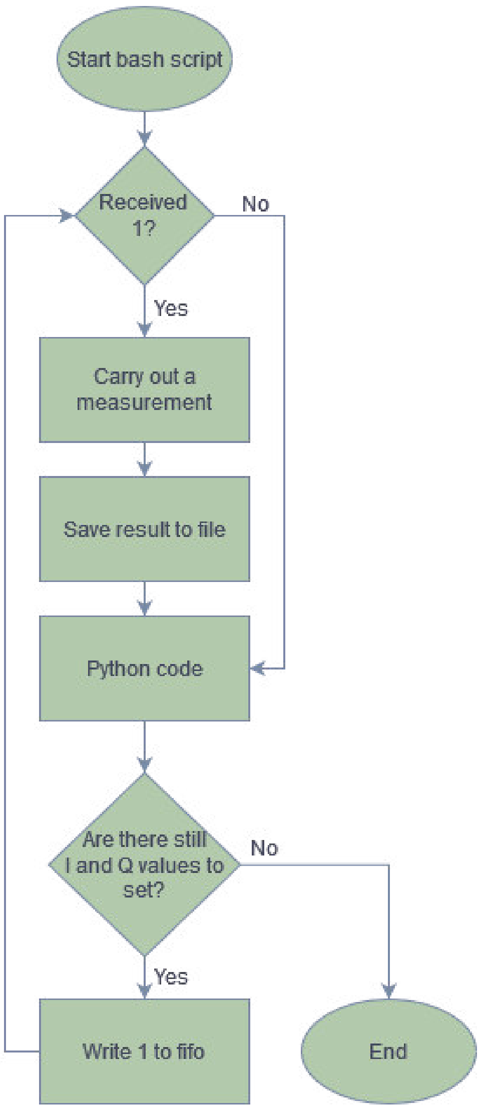


Figure 8.1: The flowchart of the calibration

8.1 The code implementing the calibration

I wrote the code for the calibration in Python.

The goal was to develop a procedure that would measure the average power values of the output data on the reference and surveillance channels with arbitrary scanning, and then find the minimum from the survelliance channel's measurements. In the next step, it takes measurements around the previously found minimum location with better resolution and then looks for a minimum here as well. The code I have written performs three iterations in succession.

The scanning itself is done by a function. It expects, as an input, the start and stop values of I and Q and a number to set the size of the step. The values of I and Q used during an iteration are stored in arrays together with the result of the measurement.

```python
def sweep(starti,startq,stopi,stopq,scanning):

  arri = []
  arrq = []
  arrp = []
```

There may be a case where a minimum location found in a previous iteration falls on the edge of the scanned area. In such a case, if the next observation area is defined in a given distance around the minimum point, it would be possible to get values less than 0 or greater than 4095, which stands outside of the full range of the measurement. For this reason, the next few lines deal with these extreme cases. If a negative number tries to be used as a starting point, 0 will be set, and if the number is too large, 4095.

```python
  if starti < 0:
    starti = 0
  if startq < 0:
    startq = 0
  if stopi > 4095:
    stopi = 4095
  if stopq > 4095:
    stopq = 4095
```

Dividing the difference between the stop and start values by the number received as a parameter gives the number of steps to be taken.

```python
  stepi = int((stopi - starti) / scanning)
  stepq = int((stopq - startq) / scanning)
```

The next loop opens the 'log.txt' file, where the measurement data is saved by the bash script. Next, it defines the serial port to use and sets the parameters required for communication. It starts by incrementing the values of I and then Q and appends both to an array. Once a pair of I and Q has been created, the program writes a 1 to fifo, which starts the code that performs the measurements. After a measurement is taken, the result is stored in the 'arrp' array. This is only the value for the surveillance channel.

```python
  with open('log.txt','r') as log:
    while log.readline() != '':
      pass
    with serial.Serial("/dev/ttyUSB0",9600,timeout = 1) as s:
```

```
    for i in range(starti,stopi,stepi):
      s.write(("0i{0}\n".format(i))[:6].encode())
      for j in range(startq,stopq,stepq):
        arri.append(i)
        arrq.append(j)
        s.write(("0q{0}\n".format(j))[:6].encode())
        with open('output.txt','a') as fo:
          p = []
          while len(p) < 2:
            with open('fifo','w') as fifo:
              fifo.write('1')
              sleep(0.1)
            p = log.readline().split()
          arrp.append(float(p[1]))
          print(f'i = {i}, q = {j}, p = {p}')
          fo.write(f'{i},{j},{p[1]}\n')
          print('1')
        sleep(0.1)
```

The next few lines are used to find the minimum average performance value. The program searches for the minimum value and its index, and then uses the index to select the required elements from both the I and Q arrays. The I, Q and power arrays are sorted into an array and are written to an output file. In order to allow multiple measurements in a well separated sequence, the file name is determined by the start and end values of I and the step size.

```
minind = np.argmin(arrp)
minp = min(arrp)
mini = arri[minind]
minq = arrq[minind]

arr=np.stack((arri,arrq,arrp),axis=-1)

with open(f'{starti}_{stopi}_{stepi}.txt','w') as f:
  for item in arr:
    f.write(f'{item[0]},{item[1]},{item[2]}\n')
```

Before visualization, the function converts the measured values into decibels, and then the plotting is also done in the sweep function, saving the generated image.

```

arrp =np.reshape(arrp,(int(math.sqrt(len(arrp))),int(math.sqrt(len(arrp))
  )))
arrp = np.transpose(arrp)

fig, ax = plt.subplots()

plt.pcolormesh(arrp)
plt.colorbar()

ax.set_xticks(np.arange(len(arrp))+0.5)
ax.set_yticks(np.arange(len(arrp))+0.5)

ax.set_xticklabels([i for i in range(starti,stopi,stepi)])
ax.set_yticklabels([i for i in range(startq,stopq,stepq)])

```

```
    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    plt.xlabel('I values')
    plt.ylabel('Q values')

    plt.savefig(f'{starti}_{stopi}_{stepi}.png', bbox_inches="tight")
```

The function returns with the parameters of the minimum value.

```
    logminp = 10*(math.log(minp))
    return mini, minq, logminp
```

In the first iteration, both the start and stop values are given, since the whole range is still being considered here, so only the step size can be changed.

```
starti1 = 0
startq1 = 0
stopi1 = 4095
stopq1 = 4095
scanning1 = 8
mini1,minq1,minp1= sweep(starti1,startq1,stopi1,stopq1,scanning1)
```

During the second and third cycles of the calibration, the length of the range to be tested can be varied around the minimum value obtained before.

```

len2 = 1000
starti2 = int(mini1-(len2/2))
startq2 = int(minq1-(len2/2))
stopi2 = int(mini1+(len2/2))
stopq2 = int(minq1+(len2/2))
scanning2 = 8
mini2,minq2,minp2 = sweep(starti2,startq2,stopi2,stopq2,scanning2)


len3 = 160
starti3 = int(mini2-(len3/2))
startq3 = int(minq2-(len3/2))
stopi3 = int(mini2+(len3/2))
stopq3 = int(minq2+(len3/2))
scanning3 = 8
mini3,minq3,minp3 = sweep(starti3,startq3,stopi3,stopq3,scanning3)
```

When the whole program has finished, it prints the results of the three calibration steps.

```
print(mini1,minq1,minp1)
print(mini2,minq2,minp2)
print(mini3,minq3,minp3)
```

The appendix contains the complete code created for the calibration.

Chapter 9

Test measurements in the laboratory

To test the operation of the circuit, I first took measurements in the laboratory.

For this, I connected the signal suppressor to a software defined radio with two channels that can also be used in the DVB-T band. Using a given code I was able to measure the average power received by the different channels.

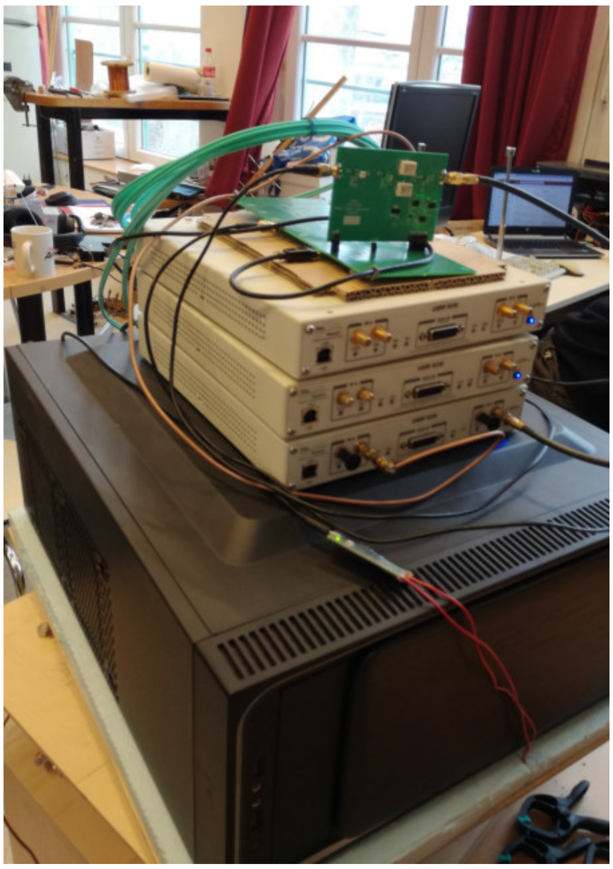


Figure 9.1: Measurement setup in the laboratory

After running the calibration, the result is also displayed in the Python code. The next three pictures, Figure 9.2, 9.3 and 9.4, show the three iterations per channel. The step size was in order 511, 125 and 20. The calibration ran for about two minutes and finally settled at (2044, 1879) as the coordinates of the minimum value, which was 24.6 dB. Compared to the 47.5 dB maximum, the signal suppression capability is the difference between the minimum and maximum values. As it follows, I was able to achieve approximately 22.9 dB signal suppression.

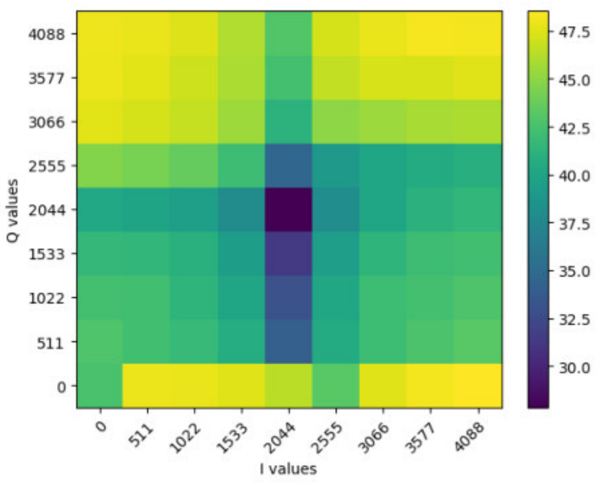


Figure 9.2: Calibration result

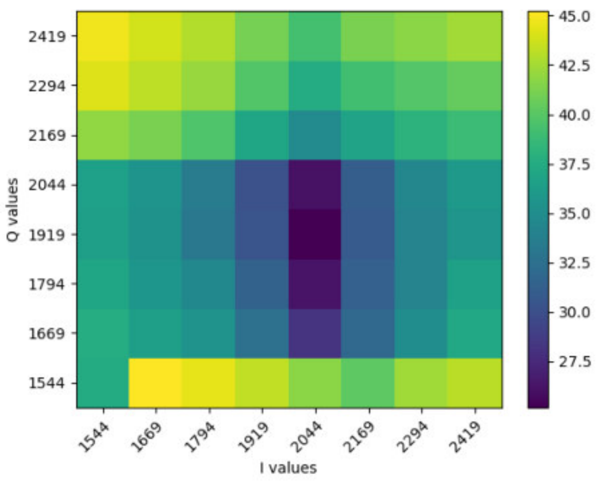


Figure 9.3: Calibration result

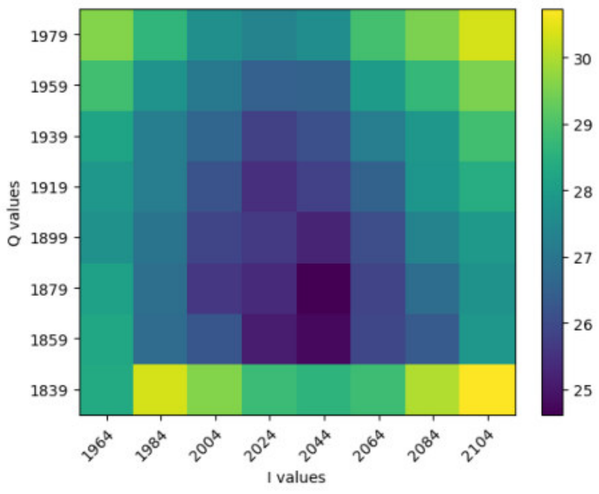


Figure 9.4: Calibration result

I repeated the measurement, but with different step sizes, starting from bigger steps and ending with smaller ones. It can be seen in Figure 9.5, 9.5 and 9.5. The end of the calibration settled at the (2026, 1881) point with a minimum value of 24.57 dB. Considering 48 dB as a reference, this time the device was able to perform a 23.43 dB suppression, which is even better than the result of the first measurement.

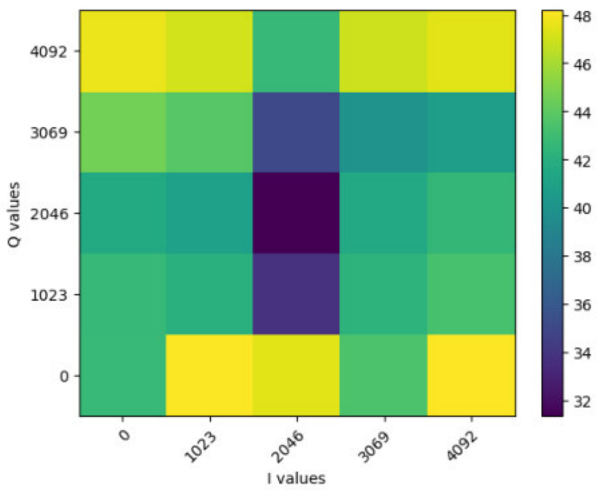


Figure 9.5: Calibration result

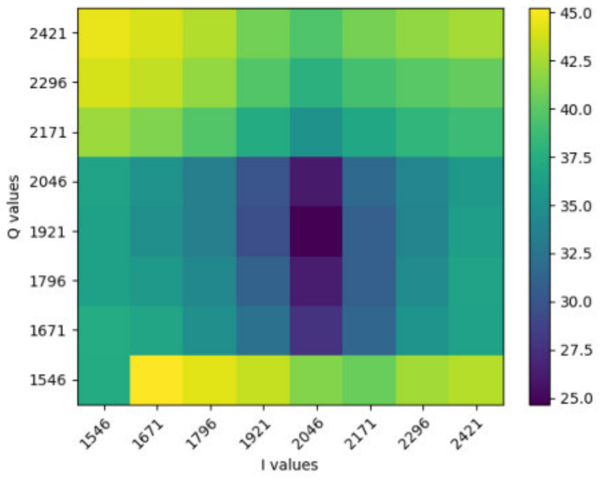


Figure 9.6: Calibration result

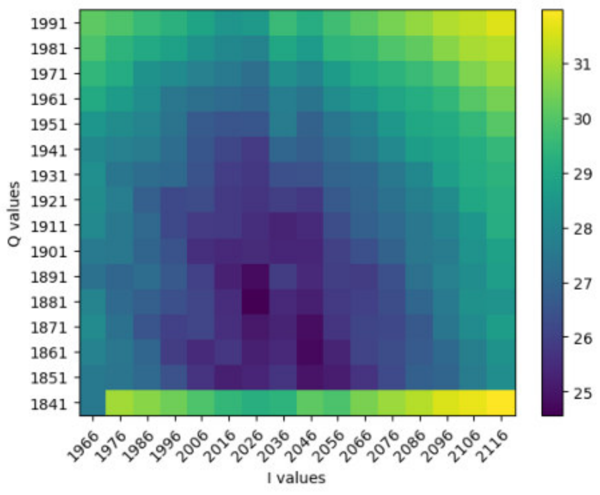


Figure 9.7: Calibration result

Chapter 10

Measurements out on the field

After testing the device in the laboratory and finding the operation sufficient, we placed an 8-element antenna system outside, which can be used in the DVB-T band.

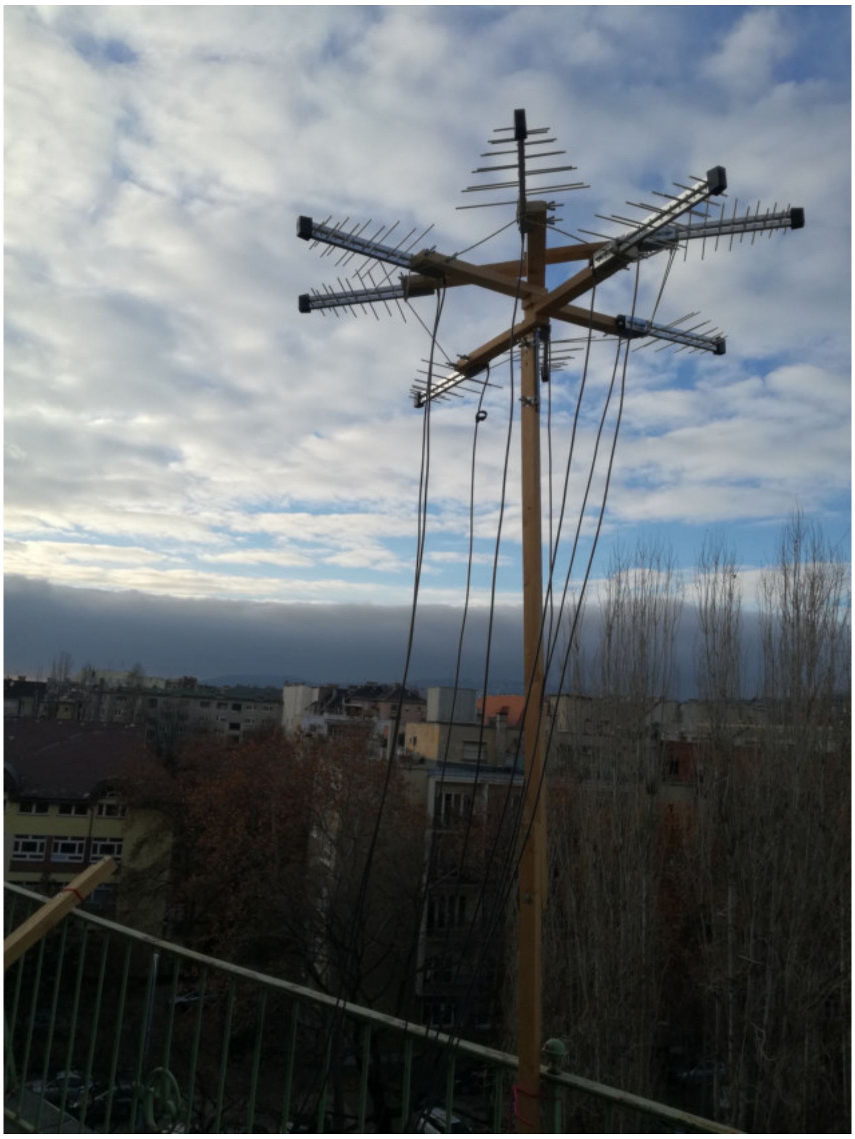


Figure 10.1: Measurement setup out on the field

The reference signal was given by the broadcast station on Széchenyi Hill and the surveillance channel pointed in another direction. I carried out a third measurement here, which gave the result of the (2046, 1860) minimum point with 24.58 dB. This means a 20.42 dB difference, the worst performance out of the three tests.

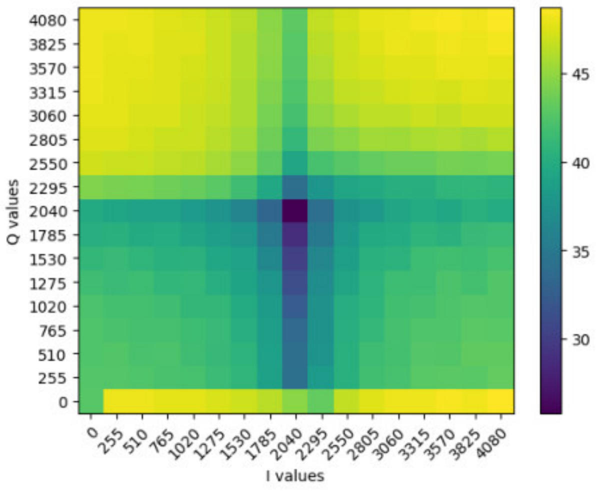


Figure 10.2: Calibration result

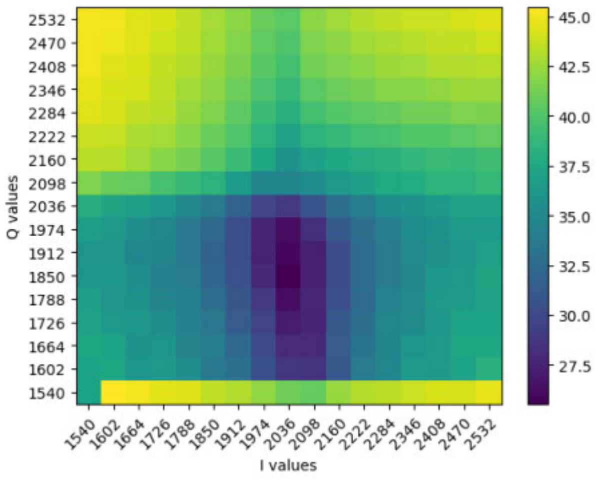


Figure 10.3: Calibration result

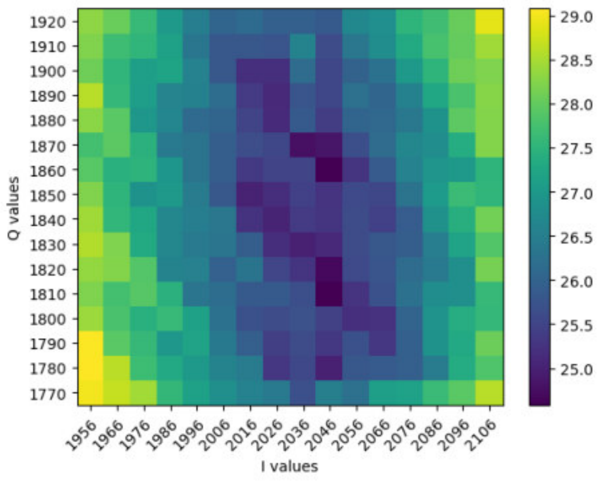


Figure 10.4: Calibration result

In conclusion, the device operates properly and it is ready to improve the performance of a DVB-T based passive radar.

Chapter 11

Summary

By the help of my thesis topic I got an overview of the process of designing, producing and testing an electrical device.

This semester, during my time in the Microwave Remote Sensing Laboratory at the Budapest University of Technology and Economics, I was introduced to high-frequency system techniques, how to design RF schematics and PCBs properly and the roles of different RF components. As working in a well-equipped laboratory, I was able to carry out RF measurements and got to know the usage of some instruments.

Besides of practical hardware skills, I could pick up some useful software knowledge as well. I learnt how to use different kind of programs for creating a schematic and PCB layout for circuits, and to simulate models for smaller parts. Moreover, I had the chance to write a code that brings the microcontroller to life and another one that gives the gist of the whole device through the calibration method.

Bibliography

[1] Szilassi András: Analóg referencia jelkioltó FM alapú passzív radarhoz, Szakdolgozat 2020

[2] Seller, Rudolf; Pető, Tamás; Dudás, Levente; Kovács, Levente: Passzív radar. I. rész; HADITECHNIKA 53 : 6 pp. 51-55. , 5 p. (2019)

[3] Fischer, W. (2010). Digital video and audio broadcasting technology. Springer. (p.267-273)

[4] PSA4-5043+ datasheet: `https://www.minicircuits.com/pdfs/PSA4-5043+.pdf`

[5] BFTC-618+ datasheet: `https://www.minicircuits.com/pdfs/BFTC-618+.pdf`

[6] SCA-4-10+ datasheet: `https://www.minicircuits.com/pdfs/SCA-4-10+.pdf`

[7] QCN-8+ datasheet: `https://www.minicircuits.com/pdfs/QCN-8+.pdf`

[8] Resistive Power Splitters: `https://www.microwaves101.com/encyclopedias/resistive-power-splitters`

[9] SYM-12+ datasheet: `https://www.minicircuits.com/pdfs/SYM-12.pdf`

[10] Diode characteristics: `https://learn.sparkfun.com/tutorials/diodes/real-diode-characteristics`

[11] How to design a precision current pump with op-amp `https://www.allaboutcircuits.com/technical-articles/how-to-design-a-precision-current-pump-with-op-amps/`

[12] MCP4822 datasheet: `https://www.farnell.com/datasheets/2125127.pdf`

[13] LM4040CIM3-2.0/NOPB datasheet: `https://www.ti.com/lit/ds/symlink/lm4040.pdf`

[14] MC34063AD datasheet: `https://www.ti.com/lit/ds/symlink/mc34063a.pdf`

[15] PIC18F26K22 datasheet: `https://4donline.ihs.com/images/VipMasterIC/IC/MCHP/MCHP-S-A0002358042/MCHP-S-A0002358042-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0`

[16] CP2102 datasheet: `https://www.farnell.com/datasheets/2245279.pdf`

[17] Optimized RF board layout for STM32WL Series

List of Figures

Appendix

Microcontroller's program code

```c
#include "mcc_generated_files/mcc.h"
#include <xc.h>
/*
                              Main application
 */
#define CH0_I 0x00
#define CH0_Q 0x01
#define CH1_I 0x10
#define CH1_Q 0x11

#define DAC_A (0<<15)
#define DAC_B (1<<15)
#define GA_1X (1<<13)
#define GA_2X (0<<13)
#define SHDN_ACTIVE (1<<12)




void DAC_Set(uint8_t ch, uint16_t value)
{
     uint16_t reg = (ch & 0x01)<<15  | GA_2X | SHDN_ACTIVE | (value & 0x0FFF
    );

     uint8_t dac_h = (uint8_t)((reg & 0xFF00)>>8);
     uint8_t dac_l = (uint8_t)((reg & 0x00FF));

     uint8_t cs = (ch & 0xFE)>>1;

     switch(cs)
     {
          case 0:
              CS0_SetLow();
              break;

          case 1:
              CS1_SetLow();
              break;

          default:
              printf("Set low error\n");
     }

     SPI1_ExchangeByte(dac_h);
     SPI1_ExchangeByte(dac_l);

```

```
        switch(cs)
    {
        case 0:
           CS0_SetHigh();
           break;

        case 1:
           CS1_SetHigh();
           break;

        default:
           printf("Set high error\n");
    }
}

void UART_Send(uint8_t *txdata, uint8_t size)
{
 int i;
    for (i = 0; i < size; i++) {
        EUSART1_Write(txdata[i]);
    }
}

void main(void)
{
    // Initialize the device
    SYSTEM_Initialize();

    // If using interrupts in PIC18 High/Low Priority Mode you need to
    enable the Global High and Low Interrupts
    // If using interrupts in PIC Mid-Range Compatibility Mode you need to
    enable the Global and Peripheral Interrupts
    // Use the following macros to:

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Enable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptEnable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    SPI1_Open(SPI1_DEFAULT);
    EUSART1_Initialize();

     __delay_ms(2000);
    printf("Program start\n");

    DAC_Set(CH0_I, 2048);
    DAC_Set(CH0_Q, 2048);

    uint16_t cnt=0;
    uint16_t cnt2=100;

    while (1)
    {
        if((cnt++)%10000 == 0)
            LED_Toggle();
        __delay_ms(200);

        if((cnt2++)%10000 == 0)
            LED2_Toggle();

        uint8_t rxData;
        uint8_t value[7];
        uint8_t i = 0;
        uint8_t len;
        uint16_t val;
        uint8_t c;
        uint8_t channel;
        uint8_t iq;
        uint8_t ch;
        uint8_t error = false;

        if(EUSART1_is_rx_ready())
        {
            do{
                len = 0;
                do
                {
                        rxData = EUSART1_Read();
                        EUSART1_Write(rxData);
                        value[len++]=rxData;
                }
                while(rxData!='\n' && len<6);

                if(value[0]>='0' && value[0]<='6')
                 channel = value[0]-'0';
                else{
                 printf("Ilyen csatorna nincs: %c.\n",value[0]);
                 break;
                }

                if(value[1]=='i')
                    iq=0;
                else if(value[1]=='q')
                    iq=1;
                else{
                    printf("IQ helytelen: %c.\n",value[1]);
                    len = 0;
                    break;
                }

                ch= channel<<1 | iq;
                val = atoi(value+2);

                if(val<=4095){
                    DAC_Set(ch, val);
                    printf("OK\n");
                    break;
                }
                else{
                printf("DAC ertek tul nagy: %d\r\n",val);
                len = 0;
                break;
            }
        }
        while(0);
    }
  }

}
```

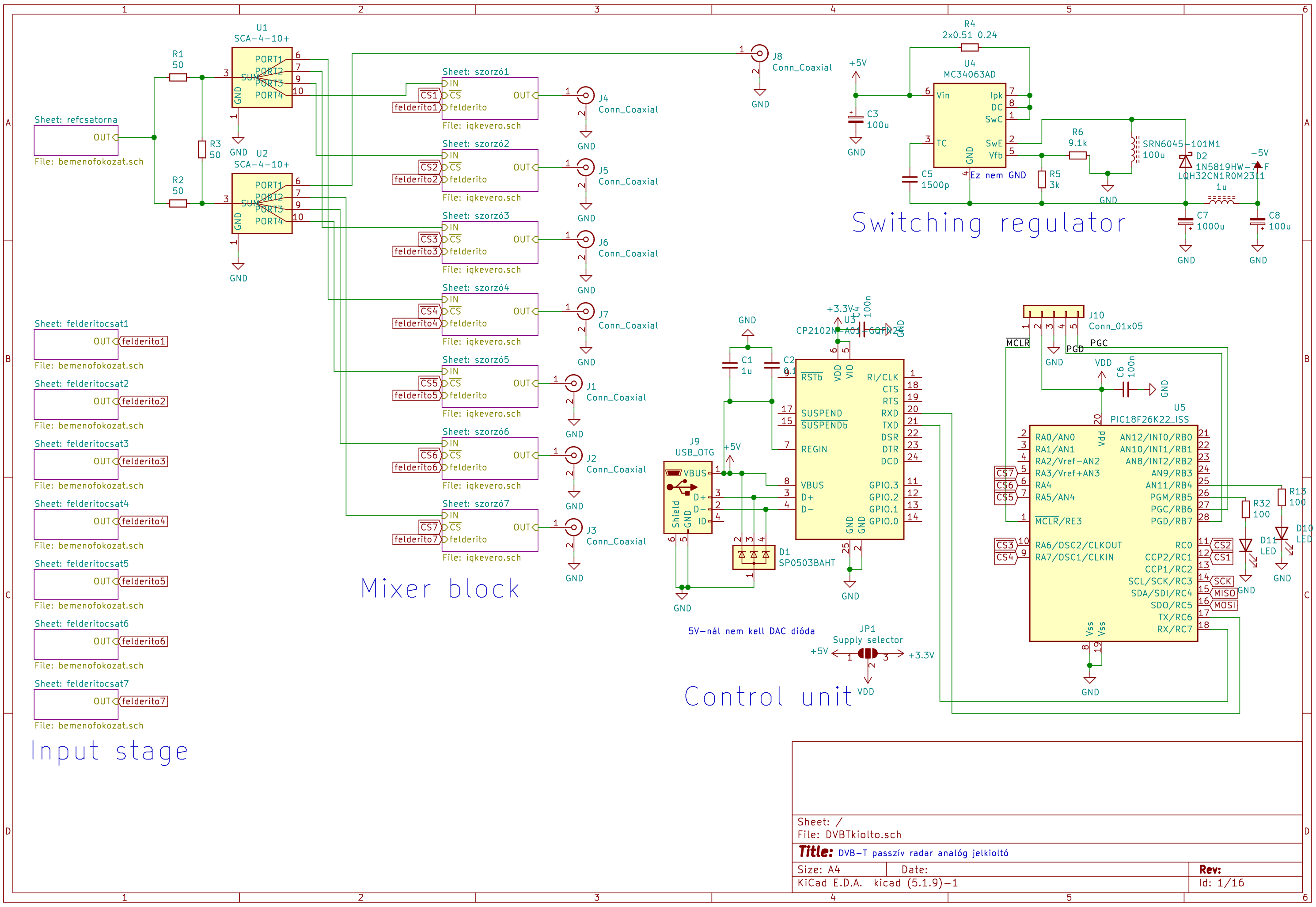
Bash script for the measurements

```
#!/bin/bash
rm fifo
pkill meas
pkill python3

mkfifo fifo
tail -f fifo | ./meas -f 510000000 -r 1000000 -g 20 -n 100000 | tee log.txt &
sleep 5
python3 kalibracio.py
```

Python code for the calibration

```
import serial
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from time import sleep
import math

def sweep(starti,startq,stopi,stopq,scanning):

  arri = []
  arrq = []
  arrp = []
  arrpnorm = []

  if starti < 0:
    starti = 0
  if startq < 0:
    startq = 0
  if stopi > 4095:
    stopi = 4095
  if stopq > 4095:
    stopq = 4095

  stepi = int((stopi - starti) / scanning)
  stepq = int((stopq - startq) / scanning)

  with open('log.txt','r') as log:
    while log.readline() != '':
      pass
    with serial.Serial("/dev/ttyUSB0",9600,timeout = 1) as s:
      for i in range(starti,stopi,stepi):
        s.write(("0i{0}\n".format(i))[:6].encode())
        for j in range(startq,stopq,stepq):
          arri.append(i)
          arrq.append(j)
          s.write(("0q{0}\n".format(j))[:6].encode())
          with open('output.txt','a') as fo:
            p = []
            while len(p) < 2:
              with open('fifo','w') as fifo:
                fifo.write('1')
                sleep(0.1)
              p = log.readline().split()
            arrp.append(float(p[1]))
            print(f'i = {i}, q = {j}, p = {p}')
            fo.write(f'{i},{j},{p[1]}\n')
            print('1')
          sleep(0.1)

  minind = np.argmin(arrp)
  minp = min(arrp)
  mini = arri[minind]
  minq = arrq[minind]

  for i in range(len(arrp)):
    arrp[i] = 10*(math.log(arrp[i]))

  arr = np.stack((arri,arrq,arrp),axis=-1)

  with open(f'{starti}_{stopi}_{stepi}.txt','w') as f:
    for item in arr:
      f.write(f'{item[0]},{item[1]},{item[2]}\n')

  # plotting

  arrp =np.reshape(arrp,(int(math.sqrt(len(arrp))),int(math.sqrt(len(arrp)))))
  arrp = np.transpose(arrp)

  fig, ax = plt.subplots()

  plt.pcolormesh(arrp)
  plt.colorbar()

  ax.set_xticks(np.arange(len(arrp))+0.5)
  ax.set_yticks(np.arange(len(arrp))+0.5)

  ax.set_xticklabels([i for i in range(starti,stopi,stepi)])
  ax.set_yticklabels([i for i in range(startq,stopq,stepq)])

  # Rotate the tick labels and set their alignment.
  plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

  plt.xlabel('I values')
  plt.ylabel('Q values')

  plt.savefig(f'{starti}_{stopi}_{stepi}.png', bbox_inches="tight")

  logminp = 10*(math.log(minp))
  return mini, minq, logminp

```

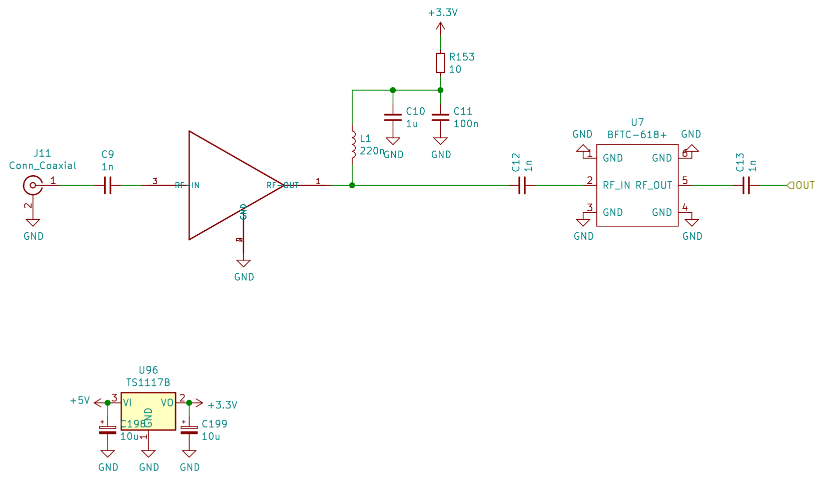
```
starti1 = 0
startq1 = 0
stopi1 = 4095
stopq1 = 4095
scanning1 = 32
mini1,minq1,minp1 = sweep(starti1,startq1,stopi1,stopq1,scanning1)


len2 = 1000
starti2 = int(mini1-(len2/2))
startq2 = int(minq1-(len2/2))
stopi2 = int(mini1+(len2/2))
stopq2 = int(minq1+(len2/2))
scanning2 = 32
mini2,minq2,minp2 = sweep(starti2,startq2,stopi2,stopq2,scanning2)


len3 = 160
starti3 = int(mini2-(len3/2))
startq3 = int(minq2-(len3/2))
stopi3 = int(mini2+(len3/2))
stopq3 = int(minq2+(len3/2))
scanning3 = 32
mini3,minq3,minp3 = sweep(starti3,startq3,stopi3,stopq3,scanning3)


print(mini1,minq1,minp1)
print(mini2,minq2,minp2)
print(mini3,minq3,minp3)
```



Sheet: /
File: DVBTkiolto.sch

Title: DVB-T passzív radar analóg jelkioltó

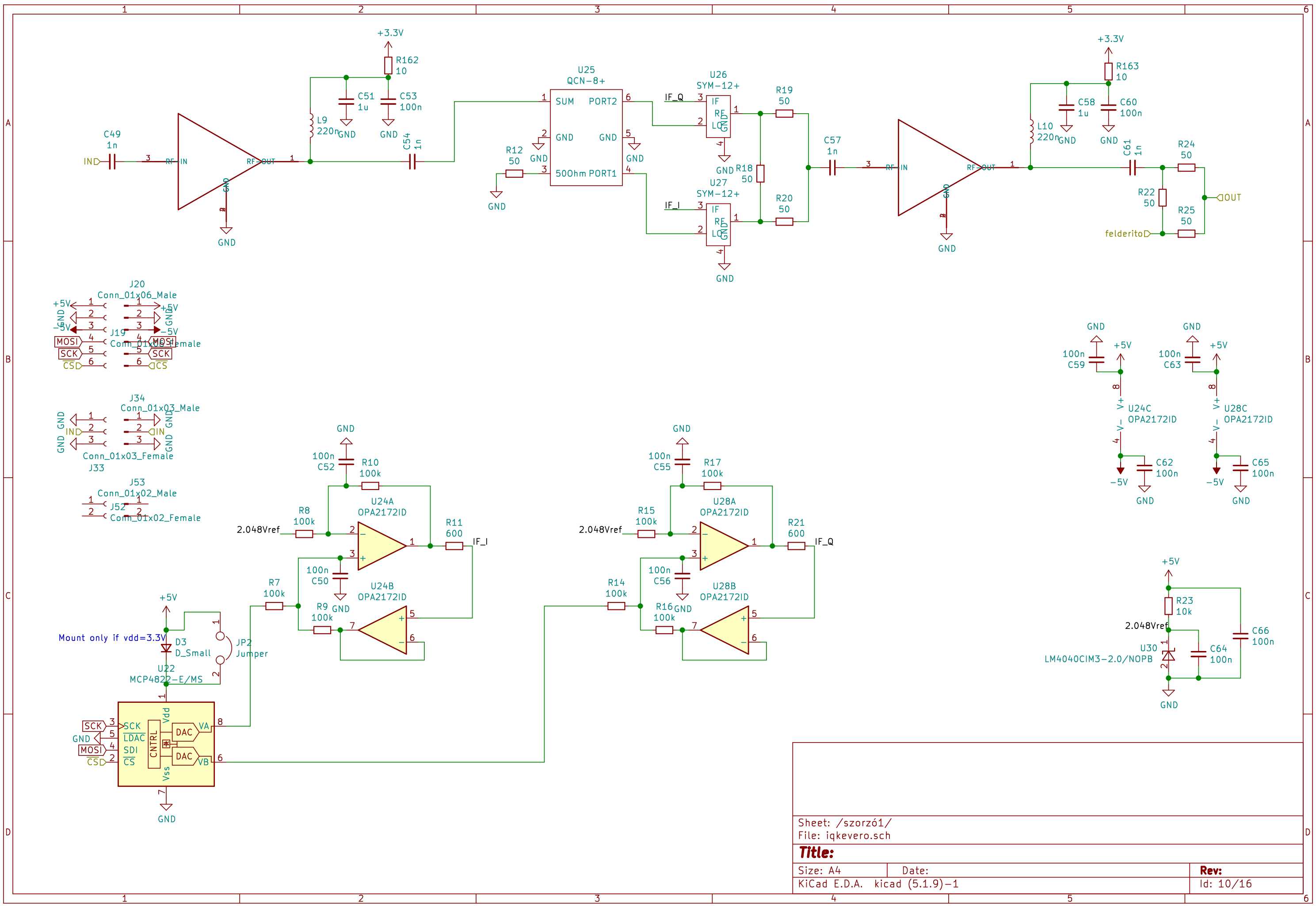
Size: A4 Date: Rev:
KiCad E.D.A. kicad (5.1.9)-1 Id: 1/16



Sheet: /refcsatorna/
File: bemenofokozat.sch

Title:

Size: A4 Date: **Rev:**
KiCad E.D.A. kicad (5.1.9)-1 Id: 2/16

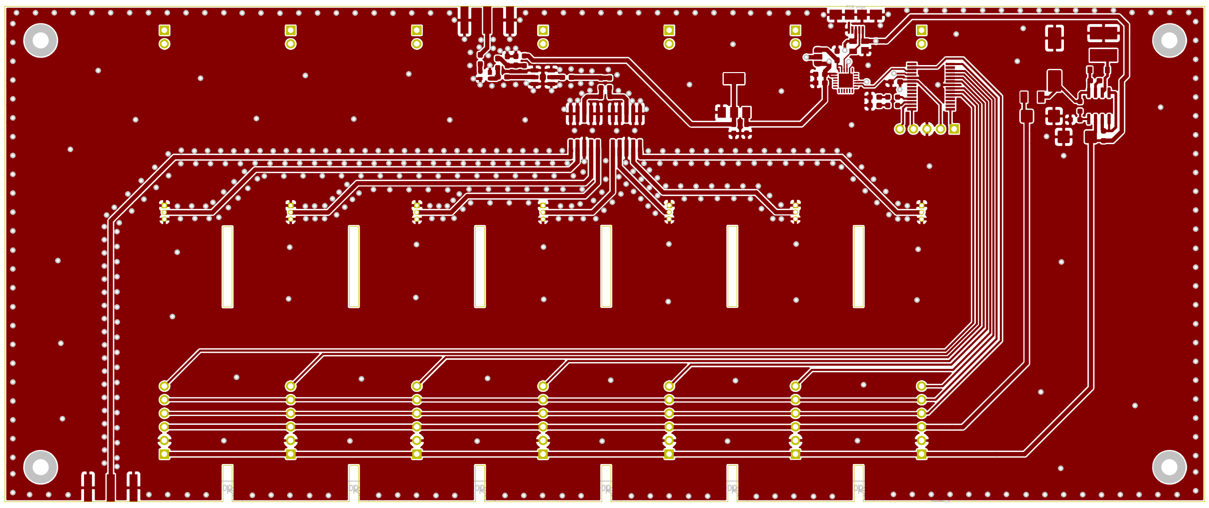


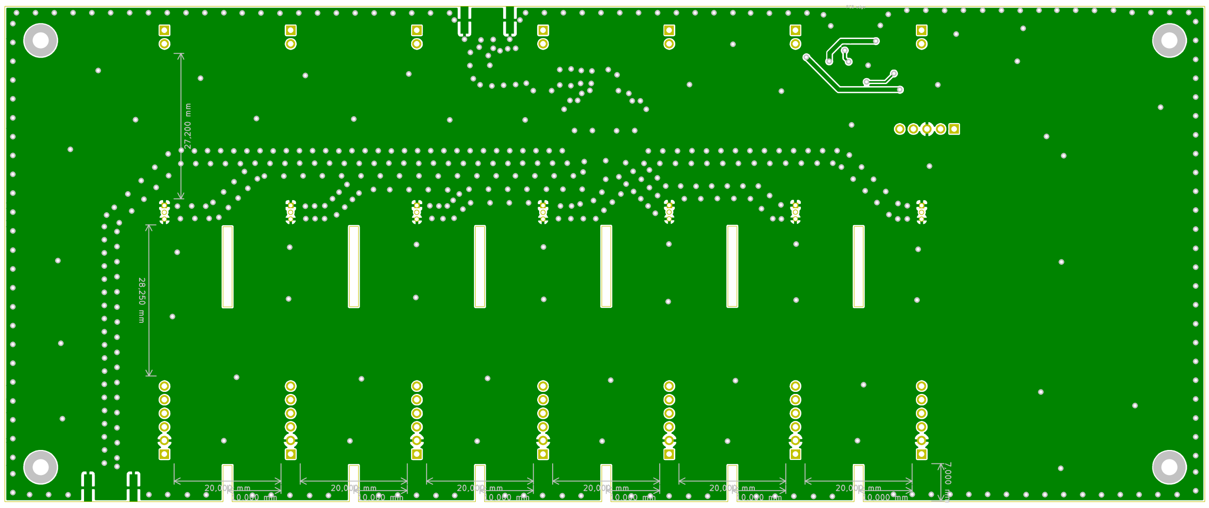
Sheet: /szorzó1/
File: iqkevero.sch

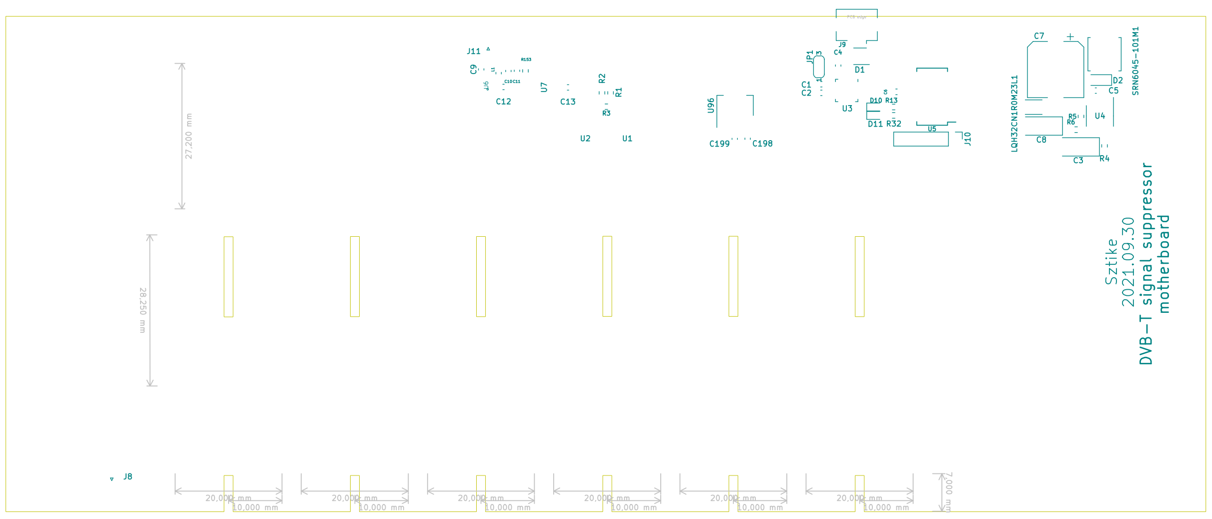
Title:

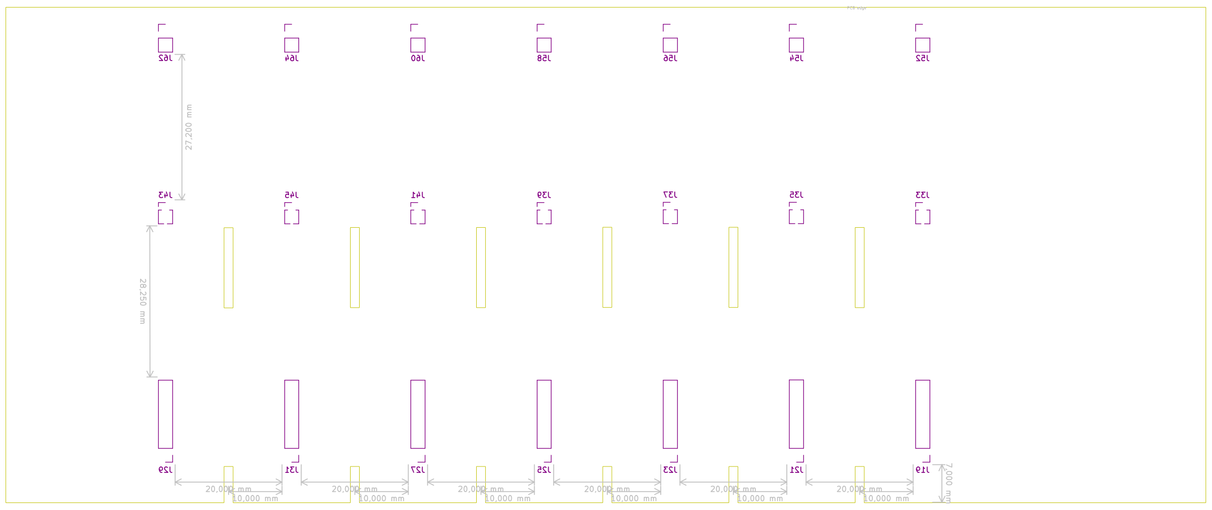
Size: A4 Date: **Rev:**

KiCad E.D.A. kicad (5.1.9)-1 Id: 10/16

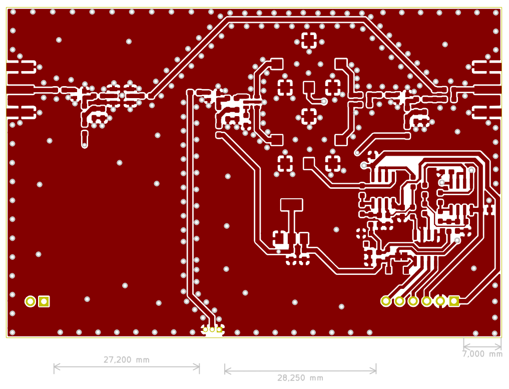



27,200 mm
28,250 mm
20,000 mm
20,000 mm
20,000 mm
20,000 mm
20,000 mm
20,000 mm
10,000 mm
10,000 mm
10,000 mm
10,000 mm
10,000 mm
10,000 mm
7,000 mm

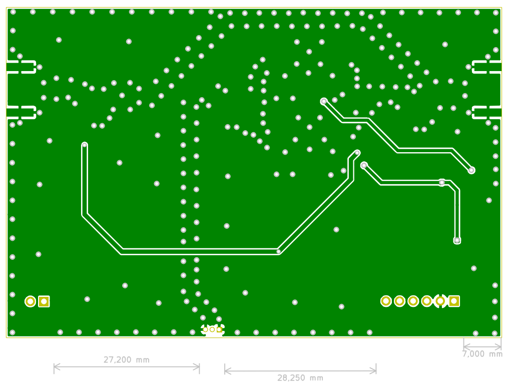

J11
R153
C9
L1
J6
C10 C11
C12
U7
C13
R2
R1
R3
U2
U1
U96
C199
C198
PCB edge
J9
JP1
C4
D1
C1
C2
U3
C6
D10 R13
D11 R32
U5
J10
C7
LQH32CN1R0M23L1
SRN6045-101M1
D2
C5
R5
R6
U4
C8
C3
R4
J8
27,200 mm
28,250 mm
20,000 mm
10,000 mm
7,000 mm
Sztike
2021.09.30
DVB-T signal suppressor
motherboard

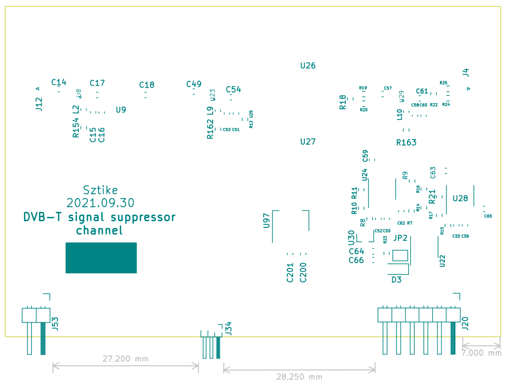


27,200 mm
28,250 mm
20,000 mm
10,000 mm
7,000 mm



27,200 mm
28,250 mm
7,000 mm


7,000 mm
27,200 mm
28,250 mm


Sztike
2021.09.30
DVB-T signal suppressor
channel
27,200 mm
28,250 mm
7,000 mm
J53
J34
J20
U26
U27
U97
C201
C200
JP2
D3
U28