RAZzies

Maandblad van de Radio Amateurs Zoetermeer

September 2022

Met in dit nummer:

- Micro SDR op een Pi-Pico deel 2 (B)
- Opa Vonk: Aflezen van een kruisnaaldmeter
- MQTT
- Dubbeltoon generator
- PA3CNO's Blog
- Afdelingsnieuws

Colofon

RAZzies is een uitgave van de Radio Amateurs Zoetermeer. Bijeenkomsten van de Radio Amateurs Zoetermeer vinden plaats op elke tweede en vierde woensdag van de maanden september - juni om 20:00 uur in het clubhuis van de Midgetgolfclub Zoetermeer in het Vernède sportpark in Zoetermeer.

Website:

http://www.pi4raz.nl

Redactie:

Frank Waarsenburg PA3CNO pa3cno@pi4raz.nl

Eindredactie:

Robert de Kok PA2RDK pa2rdk@pi4raz.nl

Informatie:

info@pi4raz.nl

Kopij en op- of aanmerkingen kunnen verstuurd worden naar razzies@pi4raz.nl

Nieuwsbrief:

http://pi4raz.nl/maillist/subscribe.php

Van de redactie

Ik ben dus niet gek. Volgens een artikel in de Telegraaf heeft één op de 300 Nederlanders een probleem met de zomer. Nou, daar hoor ik bij. Boven de 22 graden functioneer ik niet meer. Totale lusteloosheid wordt dan mijn deel, en er komt niets meer uit mijn handen. En dan moet je nog een RAZzie vol zien te krijgen. Gelukkig waren er amateurs die kopij aangeboden hebben zodat ik niet zelf iets hoefde te verzinnen. Maar ik moest het wel verwerken. Het lijkt de Halfgeleidergids van Elektuur wel... 46 pagina's deze maand. Ik hoop dat jullie de onderwerpen interessant vinden.

Nou is het niet zo dat ik helemaal niets gedaan heb, maar het meeste toch in de eerste helft van augustus, toen het nog niet zo heet was. Toen was het lekker weer om met de K1 aan een picknick tafel aan de Noord-Aa plas te gaan zitten en verbindingen te maken met de portable Magnetic Loop antenne. En daar kwam ik op 40m zowaar Henny PA3HK tegen die met zijn KX3 in de tuin zat... Dat zijn nog eens leuke verbindingen. Ook het ILLW weekend heb ik gebruikt om een reeks verbindingen te maken met vuurtorens en lichtschepen in Europa. Wist trouwens niet dat er vuurtorens in Zwitserland staan... De condities waren dat weekend niet best, wellicht door de sterke G3 uitbarsting op de zon die daarvoor net had plaatsgevonden. Afijn, 21 september is het weer herfst...

Micro SDR op een Pi-Pico (2)

Arjan te Marvelde, PE1ATM

4 Hardware prototype

Het uSDR-prototype is gebaseerd op standaardmodules en is geïnstalleerd in een Teco 1500-behuizing. Het dient voor-al als test- en experimenteeromgeving, mogelijke basis voor een meer geïntegreerde implementatie op langere termijn. De modulariteit van het prototype is echter goed voor experimenten, omdat het het mogelijk maakt om bepaalde functies van het systeem uit te wisselen die niet goed werken. Het grootste deel van de functionaliteit wordt in software gerealiseerd.

Micro SDR on a Pi-Pico (2)

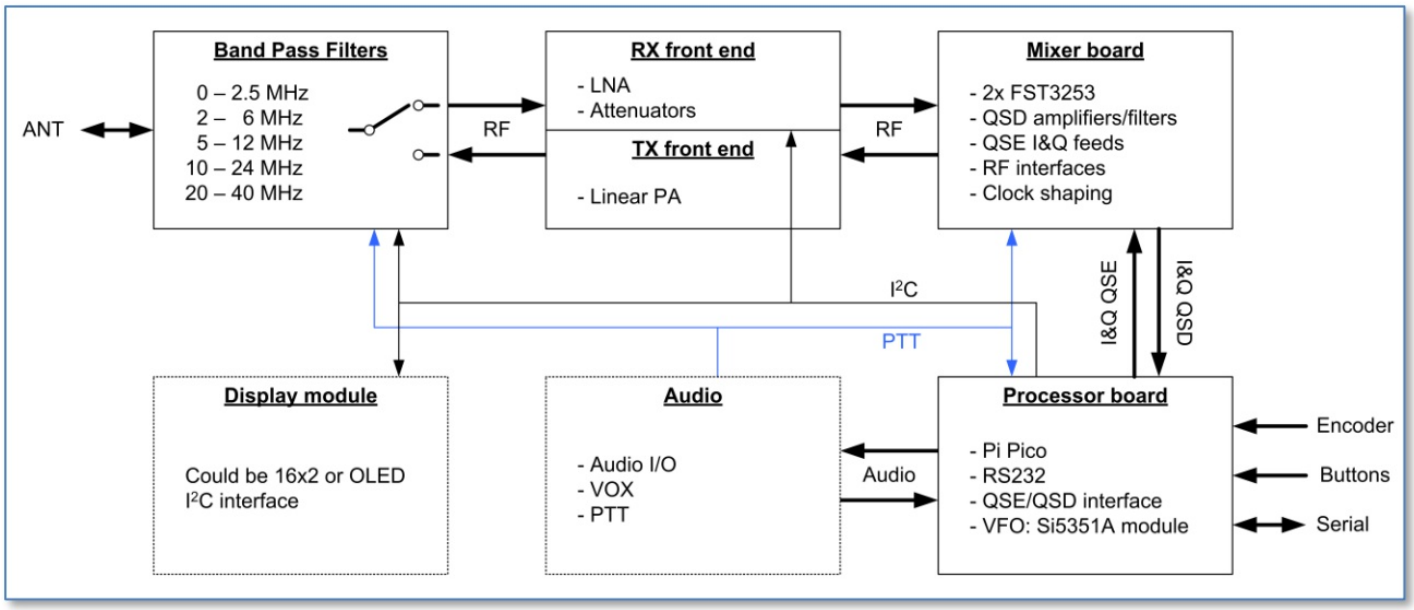
Arjan te Marvelde, PE1ATM

4 Hardware prototype

The uSDR prototype is based on off the shelf modules and is installed in a Teco 1500 enclosure. It mainly serves as a test and experimentation environment, possible basis for a more integrated implementation on the longer term. However, the modularity of the prototype is good for experimentation, since it allows to swap out certain functions of the system that do not function appropriately. The main part of the functionality is realized in software.

De modulaire architectuur is als volgt opgebouwd:

The modular architecture is set up as follows:



Het signaalpad loopt van audiokaart naar de banddoorlaatfilters en vice versa. Interne controle gebeurt via een I²C-bus en een discreet PTT-signaal. De laatste wordt ofwel doorgegeven via een microfoon, gegenereerd door de VOX of uitgegeven door SW in de processor.

The signal path leads from Audio board to BPF and vice versa. Internal control is done through an I²C bus and a discrete PTT signal. The latter is either passed through from a microphone, generated by the VOX or issued by SW in the Processor.

Samengevat zijn de modules:

- Processorbord: Dit is gewoon een drager voor zowel de Pi Pico als de VFO-module. De interfacing omvat de RS232-nivellering, switch-debouncing en PWM/DAC-filtering
- Audiokaart: dit bevat alle analoge audio-verwerking, VOX en externe PTT-interfaces.
- Displaymodule: De displaymodule kan van alles zijn met een I2C-interface. De SW is gemaakt voor een normaal 16x2 alfanumeriek type.
- Mixerboard: Dit is een ontwerp gebaseerd op twee FST3253 multiplexers. Klokvorming wordt gedaan met een 7400 ACT of HCT en analoge IF-signaalverwerking met drie LM4562.
- RX/TX front-ends: deze twee boards bevatten de TX PA en de RX LNA en verzwakkers.
- Filterbord: Dit bevat een set van 5 geschakelde banddoorlaatfilters, die tussen RF front-end en antenne in gaan. Het schakelen tussen het RX- en TX-pad wordt gedaan door een PTT gestuurd relais.

In summary, the modules are:

- Processor board: This is simply a carrier for the Pi Pico as well as the VFO module. The interfacing comprises the RS232 levelling, switch debouncing and PWM/DAC filtering
- Audio board: This contains all analogue audio handling, VOX and external PTT interfaces.
- Display module: The display module can be anything suitable with an I2C interface. The SW is made for a regular 16x2 alphanumeric type.
- Mixer board: This is a design based on two FST3253 multiplexers. Clock shaping is done with a 7400 ACT or HCT, and analogue IF signal handling with three LM4562.
- RX/TX front ends: These two boards contain the TX PA and the RX LNA and attenuators.
- Filter board: This contains a set of 5 switched bandpass filters, that go in between RF front-end and antenna. The switching between the RX and TX path is done by a PTT controlled relay.

De processor-, mixer-, RX- en TX-kaarten hebben allemaal dezelfde vormfactor, 51x82 mm (2" x 3,2"). Het filterbord paste niet en is iets groter (51x94mm).

Het kan de voorkeur hebben om de TX-PA-trap tussen antenne en BPF te verplaatsen. Dit zou echter een andere set laagdoorlaatfilters vereisen. De reden is om ongewenste signalen die in de mixer ontstaan, te filteren.

4.1 Processor board

Zie het schema op pagina 5. Het processorbord herbergt de Pico-module, de Si5351A VFO-module en de gebruikersinterfaces. Er is ook een seriële interface die een monitor voorziet van een opdrachtregelinterface. De audioverwerking en PTT/VOX-schakelingen zijn om modulariteitsredenen buiten het CPU/IO-bord geplaatst.

De I- en Q-uitgangsfilters hebben een afsnijfrequentie van 3,4 kHz, het lijnuitgangsfilter 7,2 kHz.

4.2 Mixer board

Zie het schema op pagina 6. Het mixerbord herbergt de twee mixers, QSD voor de RX-tak en een QSE voor de TX-tak. De schakelaars worden geklokt vanaf het Si5351-bord, via twee HCT NAND-poorten die zorgen voor niveauverschuiving en wat signaalverbetering. Het TX-pad kan worden uitgeschakeld als PTT niet actief is. Het RX-pad is altijd ingeschakeld.

De weerstanden van de TX ingangsopamps zijn verhoogd om een hogere ingangsimpedantie te verkrijgen. Dit zorgt ervoor dat het signaal niet te veel wordt verdeeld. De low-pass afsnijfrequentie is ongeveer 4,8 kHz.

Een aandachtspunt is de QSE-tak, het uitgangssignaal lijkt veel vervorming te hebben. Misschien zijn een paar belastingsweerstanden op de FST3253-uitgangen of een aanpassing

The Processor, Mixer, RX and TX boards all have the same form factor, 2” x 3.2” (51x82mm). The Filter board did not fit and is slightly bigger (51x94mm).

It could be preferrable to move the TX-PA stage between antenna and BPF. This would however require another set of low-pass filters. The reason is to filter unwanted signals that originate in the mixer.

4.1 Processor board

See the schematic diagram on page 5. The processor board hosts the Pico module, the Si5351A VFO module and the user interfaces. There is also a serial interface that provides a monitor with a command line interface. The audio handling and PTT/VOX circuitry is moved outside the CPU/IO board, for modularity reasons.

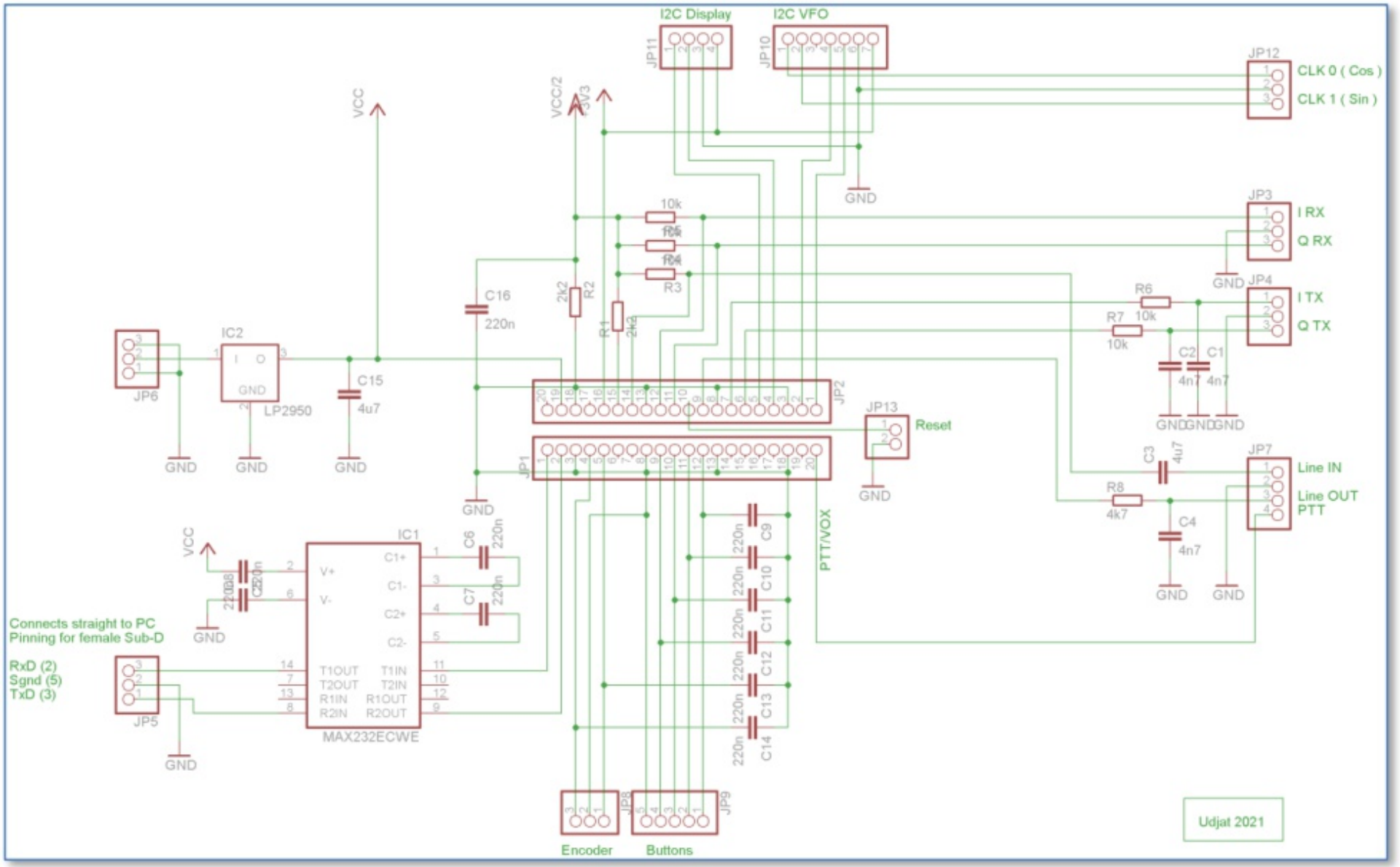
The I and Q output filters have a cutoff frequency of 3.4kHz, the line output filter 7.2kHz.

4.2 Mixer board

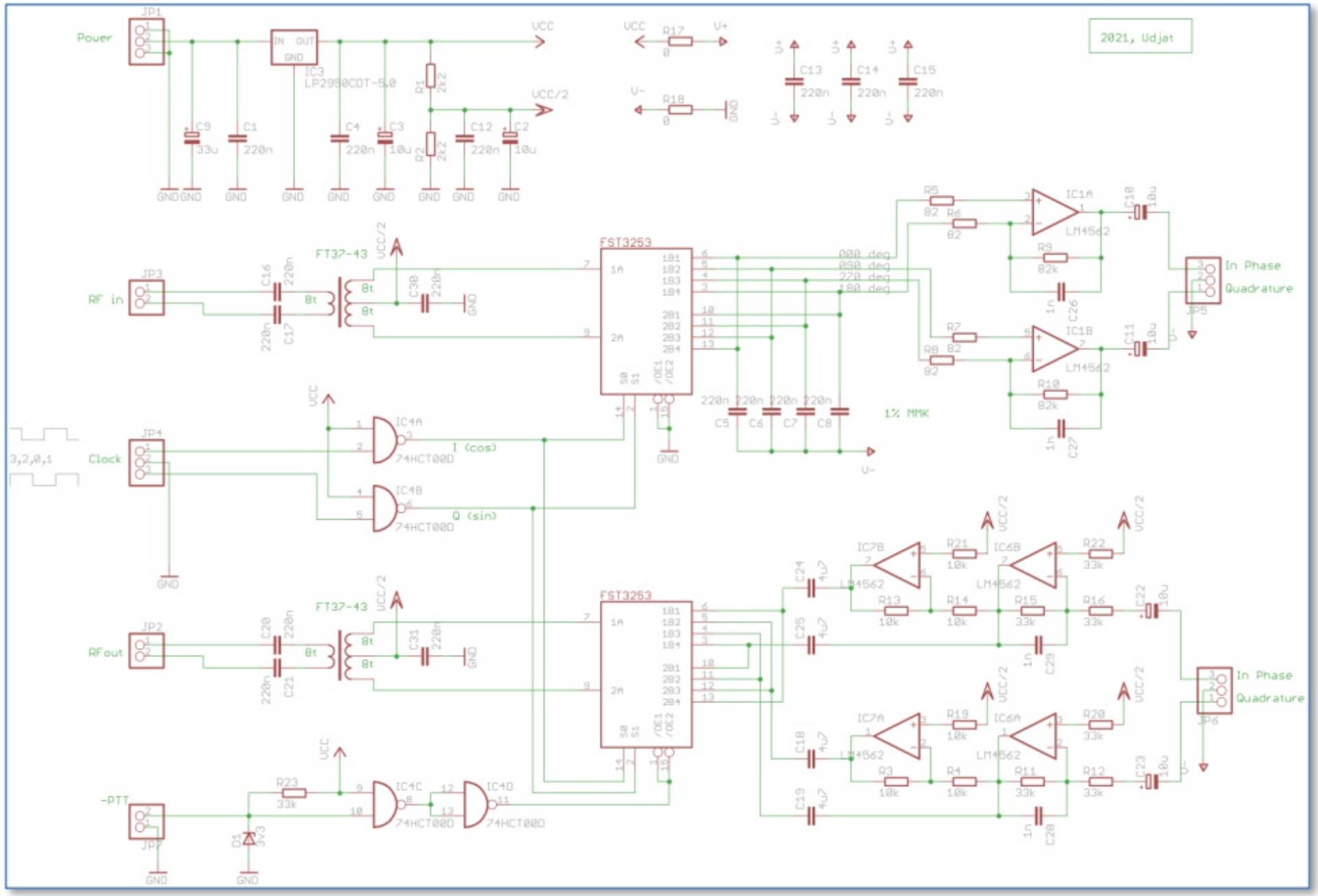
See the schematic diagram on page 6. The mixer board hosts the two mixers, QSD for RX branch and a QSE for the TX branch. The switches are clocked from the Si5351 board, through two HCT NAND gates that take care of level shifting and some signal cleanup. TX path can be disabled when PTT is not active. RX path is always enabled.

The resistors of the TX input opamps have been increased in order to obtain a higher input impedance. This makes sure that the signal is not divided too much. The low-pass cutoff frequency is about 4.8kHz.

A point of attention is the QSE branch, the output signal seems to have a lot of distortion. Maybe a couple of load resistors on the FST3253 outputs or some bias adjustment will



Processor Board


2021, Udjat
Power
JP1
IC3
LP2950CDT-5.0
VCC
VCC/2
C9 33u
C1 220n
C4 220n
C3 10u
C12 220n
C2 10u
R1 2k2
R2 2k2
R17 0
R18 0
C13 220n
C14 220n
C15 220n
RF in
JP3
FT37-43
8t
8t
C16 220n
C17 220n
C30 220n
FST3253
000 deg
090 deg
270 deg
180 deg
IC1A
IC1B
LM4562
R5 82
R6 82
R7 82
R8 82
R9 82k
R10 82k
C26 1n
C27 1n
C10 10u
C11 10u
In Phase
Quadrature
JP5
C5 220n
C6 220n
C7 220n
C8 220n
1% MMK
3,2,0,1
Clock
JP4
IC4A
IC4B
74HCT00D
I (cos)
Q (sin)
RFout
JP2
C20 220n
C21 220n
C31 220n
IC7B
IC6B
IC7A
IC6A
R21 10k
R22 33k
R13 10k
R14 10k
R15 33k
R16 33k
R19 10k
R20 33k
R3 10k
R4 10k
R11 33k
R12 33k
C24 4u7
C25 4u7
C18 4u7
C19 4u7
C29 1n
C28 1n
C22 10u
C23 10u
JP6
-PTT
JP7
R23 33k
IC4C
IC4D
D1 5V3
GND
V+
V-

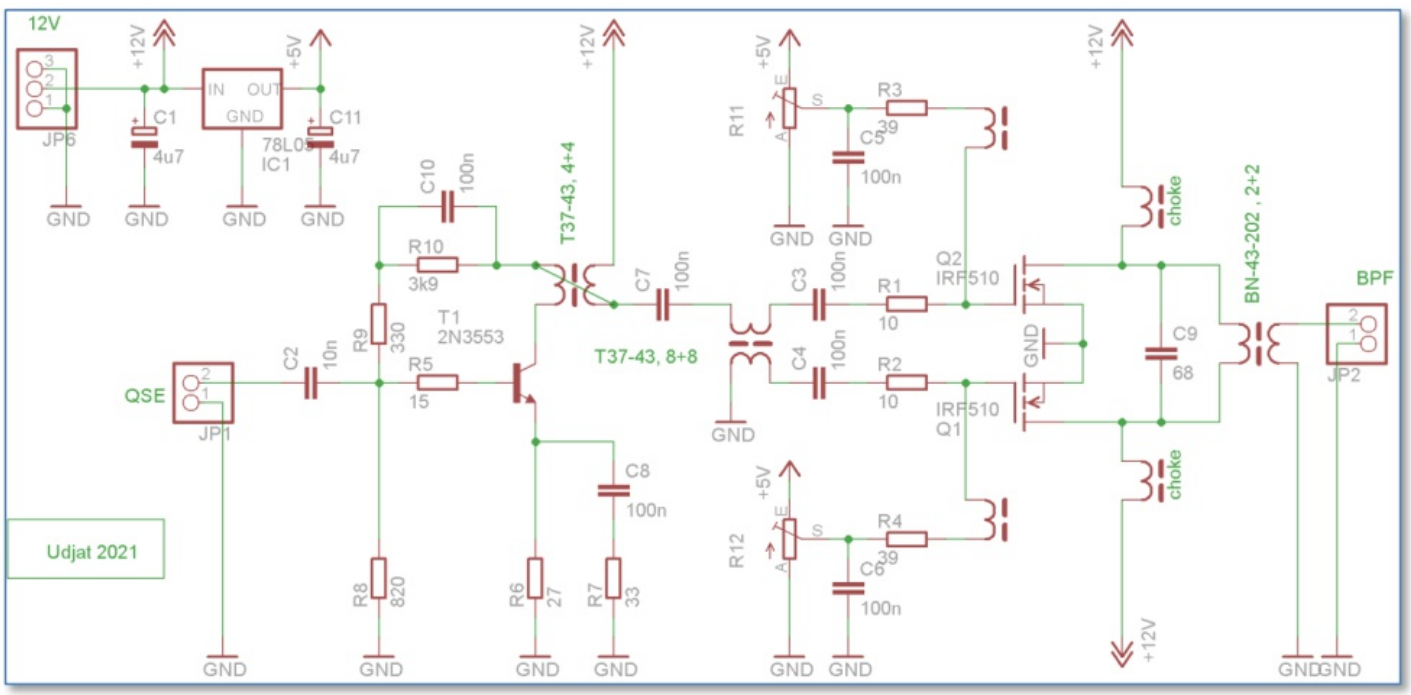
Mixer Board

van de bias hier voldoende. Een andere overweging is om de scheidingscondensatoren en het voorspanningscircuit te verwijderen, omdat de opamp-uitgangen al een voorspanning hebben. Deze moeten dan in alle 4 de paden identiek zijn, om voldoende draaggolf- en tegenoverliggende zijbandonderdrukking te behouden.

do the trick here. Another consideration is to remove the separation capacitors and the bias circuit, since the opamp outputs already have a bias. These then have to be identical in all 4 paths, in order to retain sufficient carrier and opposite sideband suppression.

4.3 TX board

4.3 TX board



Het nieuwe proto TX-bord bestaat uit een op IRF510 gebaseerde push-pull-versterker in klasse AB, aangestuurd door een standaard klasse A-trap. Deze driver heeft een 2N3553 voor proto, maar dat kan van alles zijn zoals 2N228018, 2N2219, 2N2222 etc.

The new proto TX board consists of a class AB, IRF510 based push-pull amplifier, driven by a regular class A stage. This driver has a 2N3553 for proto, but that could be anything like 2N228018, 2N2219, 2N2222 etc.

Bias wordt aangepast tot net boven cutoff, dus een ruststroom van ongeveer 20mA per MOSFET-drain, die moet worden aangepast voor signaalsymmetrie wanneer de versterker operationeel is.

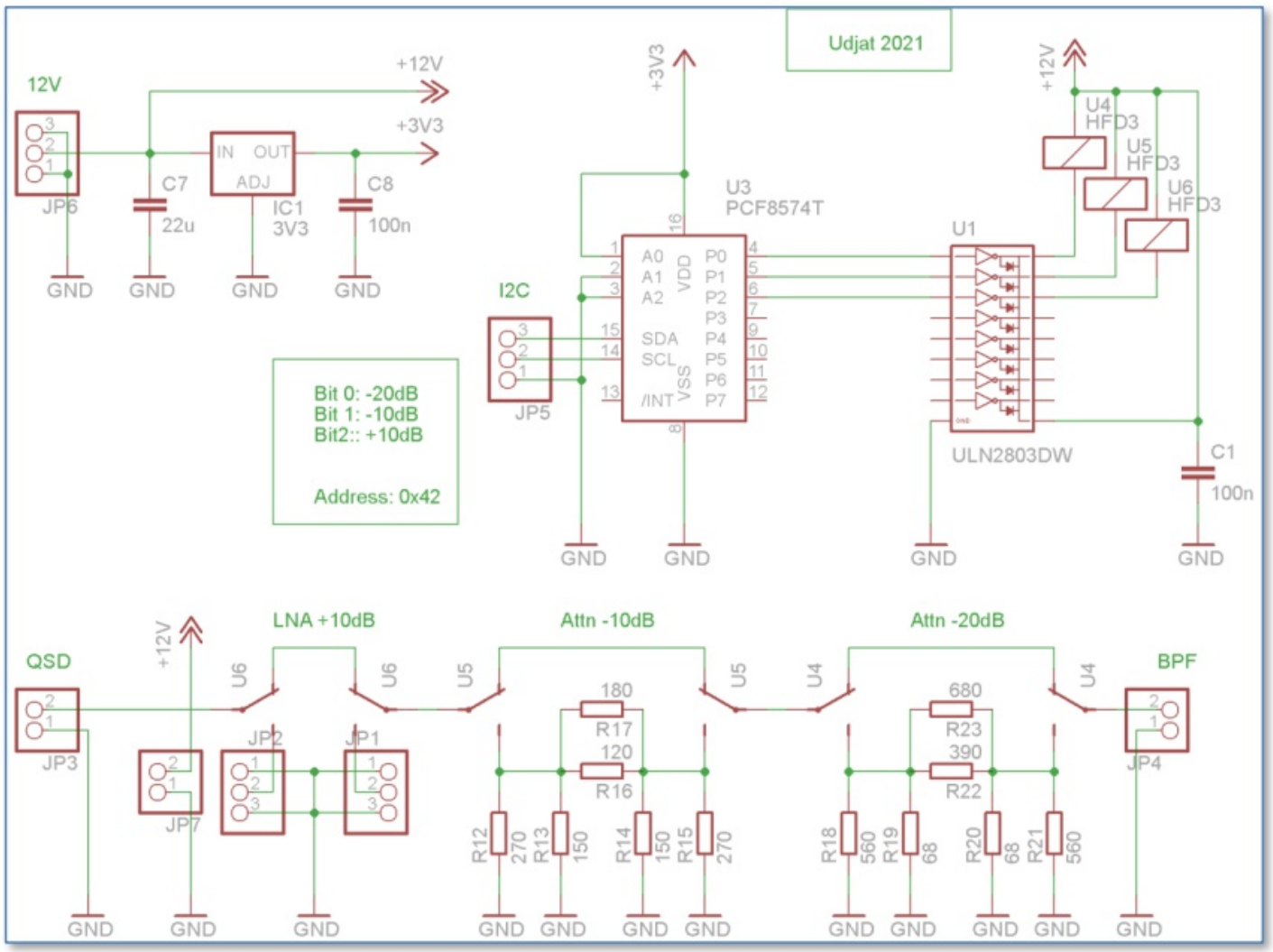
Bias is adjusted to just above cutoff, so a standing current of about 20mA per MOSFET drain, to be adjusted for signal symmetry when the amplifier is operational.

Dit is slechts een voorbeeld PA, die nog verder moet worden getest.

This is just an example PA, which needs to be further tested.

4.4 RX board

4.4 RX board



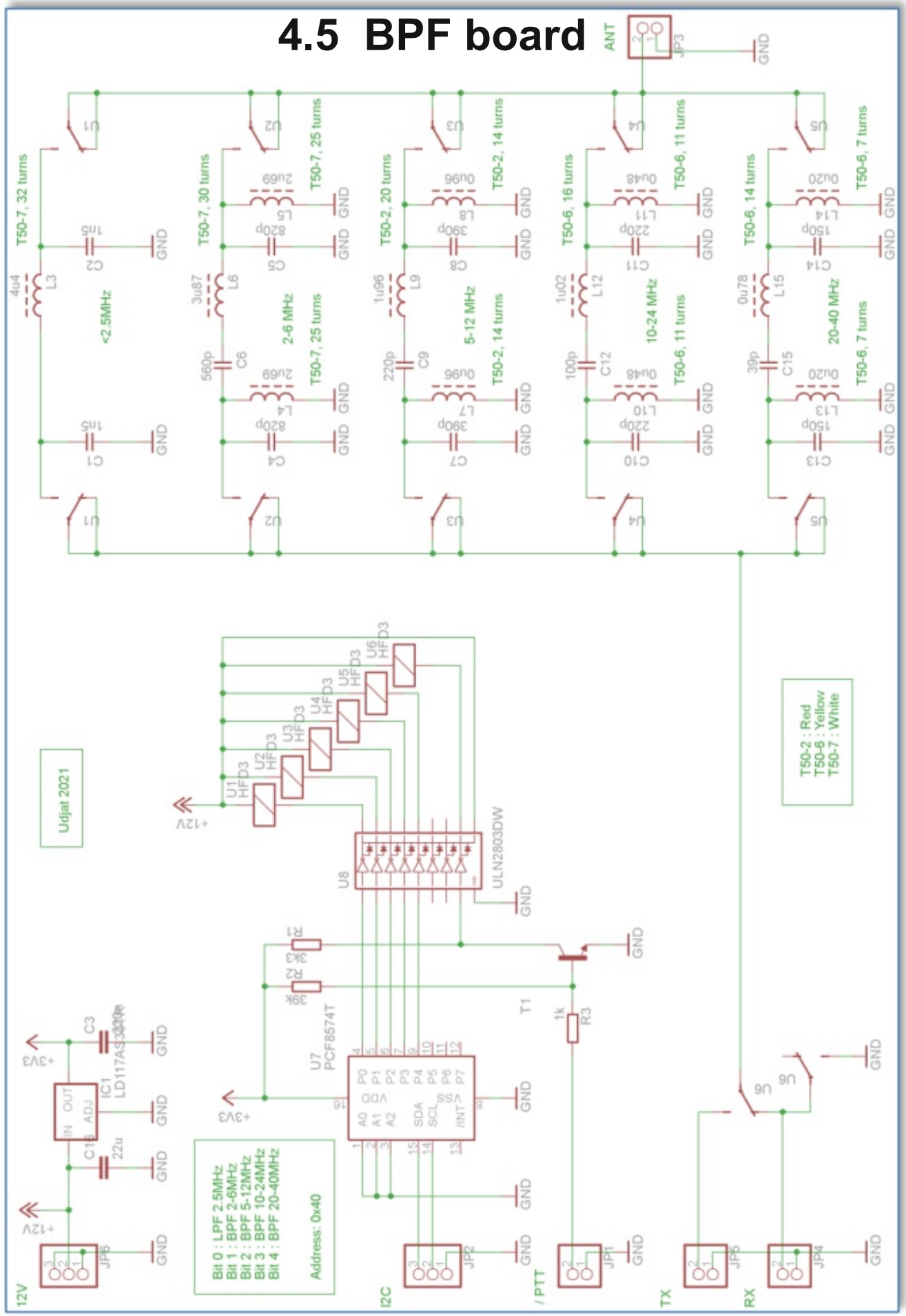
Het RX-bord bevat alleen de instelbare verzwakkers en een ruisarme versterker.

Een ding dat in volgende versies moet veranderen, is de interface PCF8574 – ULN2803, de I²C-expander kan nauwelijks de stuurstroom voor de Darlington-array leveren.

The RX board just contains the selectable attenuators and a low noise amplifier.

One thing to change in subsequent versions is the interface PCF8574 – ULN2803, the I²C expander can barely provide the drive current for the Darlington array.

4.5 BPF board

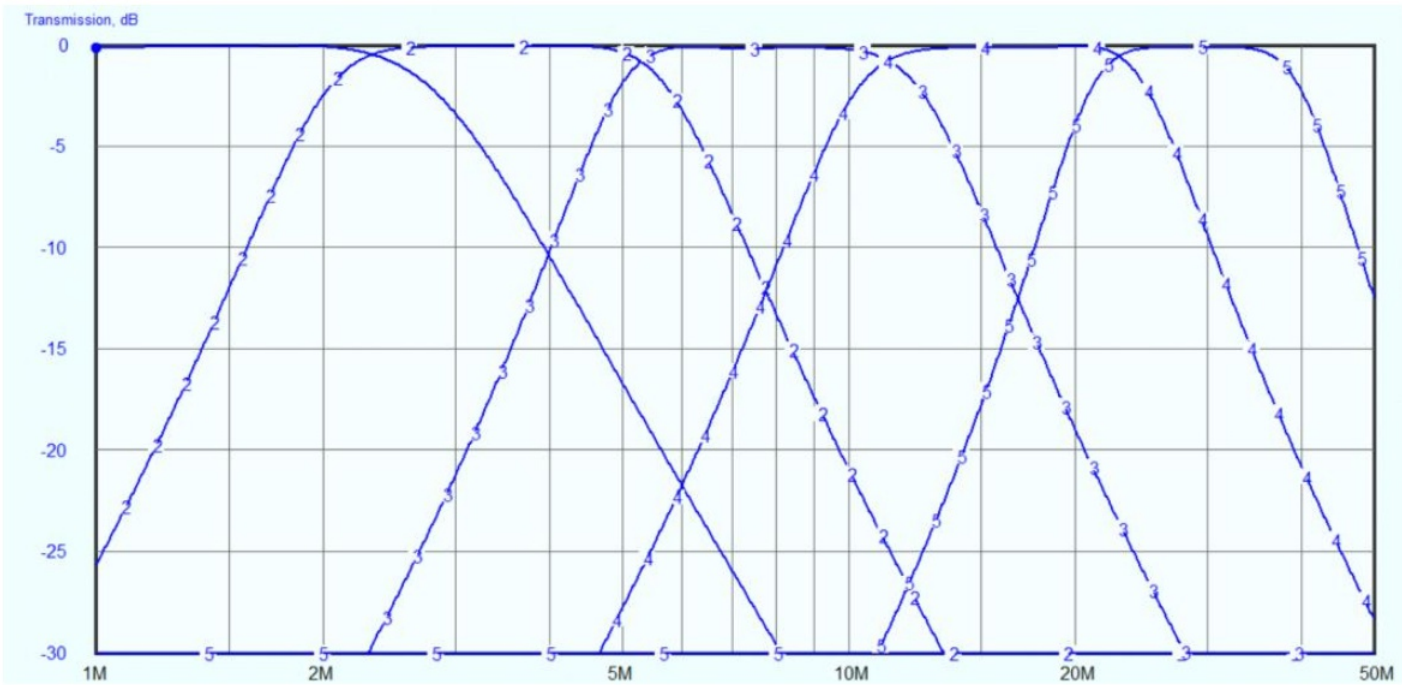

ANT
JP3
GND
T50-7, 32 turns
T50-7, 30 turns
T50-7, 25 turns
T50-2, 20 turns
T50-2, 14 turns
T50-6, 16 turns
T50-6, 11 turns
T50-6, 14 turns
T50-6, 7 turns
<2.5MHz
2-6 MHz
5-12 MHz
10-24 MHz
20-40 MHz
T50-2 : Red
T50-6 : Yellow
T50-7 : White
Udjat 2021
+12V
+3V3
ULN2803DW
PCF8574T
LD1117AS33
Bit 0 : LPF 2.5MHz
Bit 1 : BPF 2-6MHz
Bit 2 : BPF 5-12MHz
Bit 3 : BPF 10-24MHz
Bit 4 : BPF 20-40MHz
Address: 0x40
12V
I2C
/ PTT
TX
RX

De 5 bandfilters worden geselecteerd via relais, aangestuurd vanuit de Pico via de I²C-bus. Een zesde relais is verbonden met het PTT-signaal en verbindt het RX- of TX-pad met de filters.

The 5 band filters are selected through relays, controlled from the Pico via the I²C bus. A sixth relay is connected to the PTT signal, and connects either RX or TX path to the filters.

De (ongeveer octaaf) filters zijn berekend met ELSIE, en hebben de volgende doorlaat karakteristieken:

The (roughly octave) filters are calculated with ELSIE, and have the following pass-through characteristics:



Een andere kleine wijziging is R3 in het PTT-circuit, die nodig was omdat de basis van T1 het PTT-signaal te ver naar beneden trok.

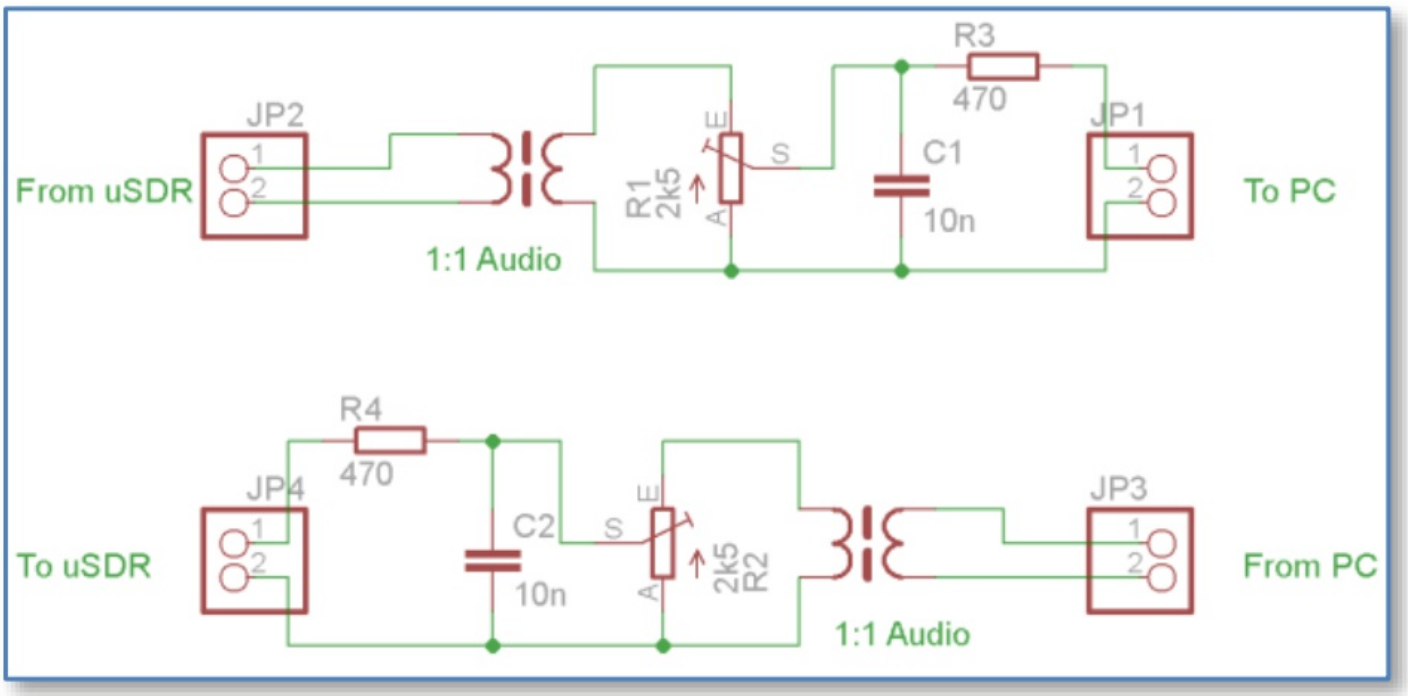
Another slight modification is R3 in the PTT circuit, which was needed because the base of T1 pulled the PTT signal too far down.

4.6 Audio module

Er zijn twee modules gemaakt op protoborden om de audioverwerking te implementeren. De eerste is het bufferen en voorversterken van de lijn- en microfoonkanalen, evenals een extra laagdoorlaatfilter van 3 kHz. Het line-in signaal wordt 6dB versterkt om wat extra bereik te krijgen voor niveau-optimalisatie. Het signaal dat naar de ADC gaat, moet dicht bij 3Vpp zijn. Het microfoonsignaal is min of meer gebalanceerd en versterkt met 10dB. In de huidige versie moet dit nog getest worden. Het microfoon PTT-signaal werkt alleen als No VOX is geselecteerd, anders wordt het PTT signaal niveaugestuurd.

4.6 Audio module

Two modules were made on proto boards to implement the audio handling. The first is buffering and pre-amplifying the line and microphone channels, as well as an additional 3kHz low pass filter. The line in signal is amplified 6dB to get some extra range for level optimization. The signal that goes to the ADC should be close to 3Vpp. The mike signal is more or less balanced and amplified 10dB. In the current version this is still to be tested. The mike PTT signal is only enabled when No VOX is selected, otherwise the PTT is level controlled.

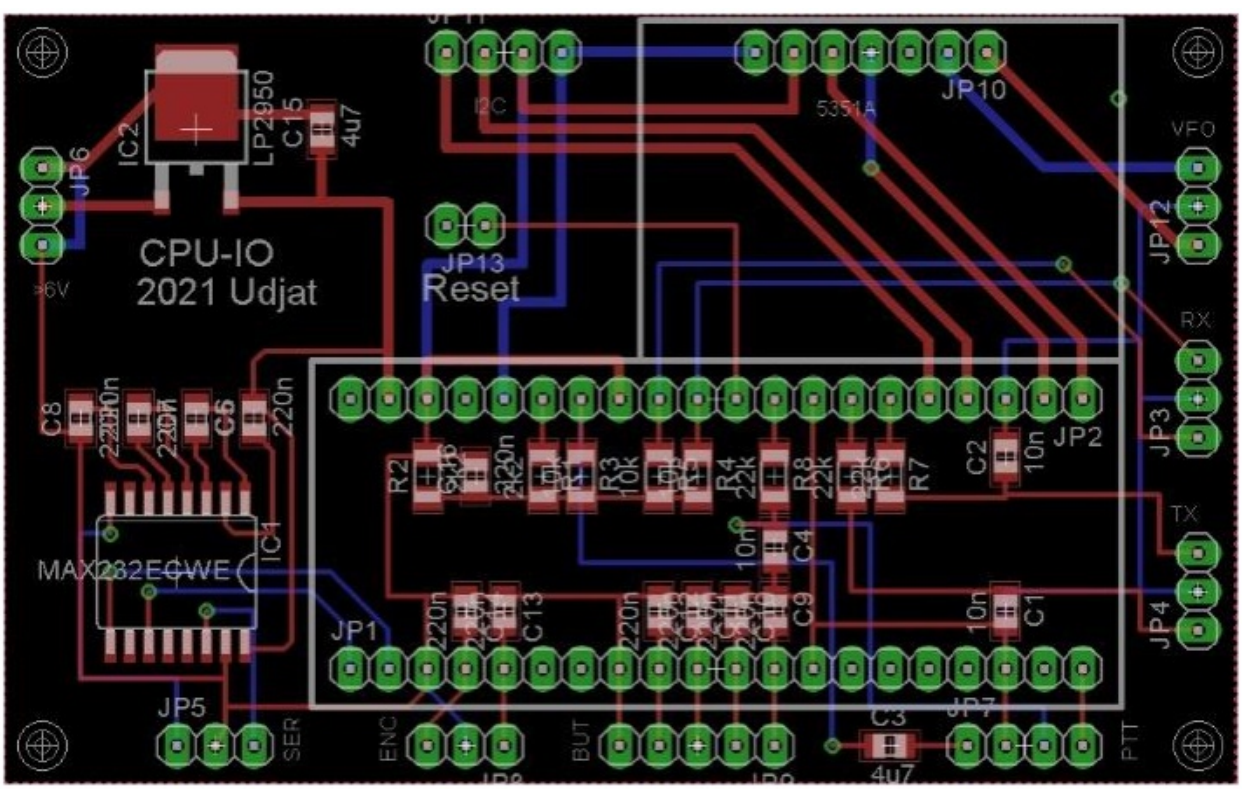


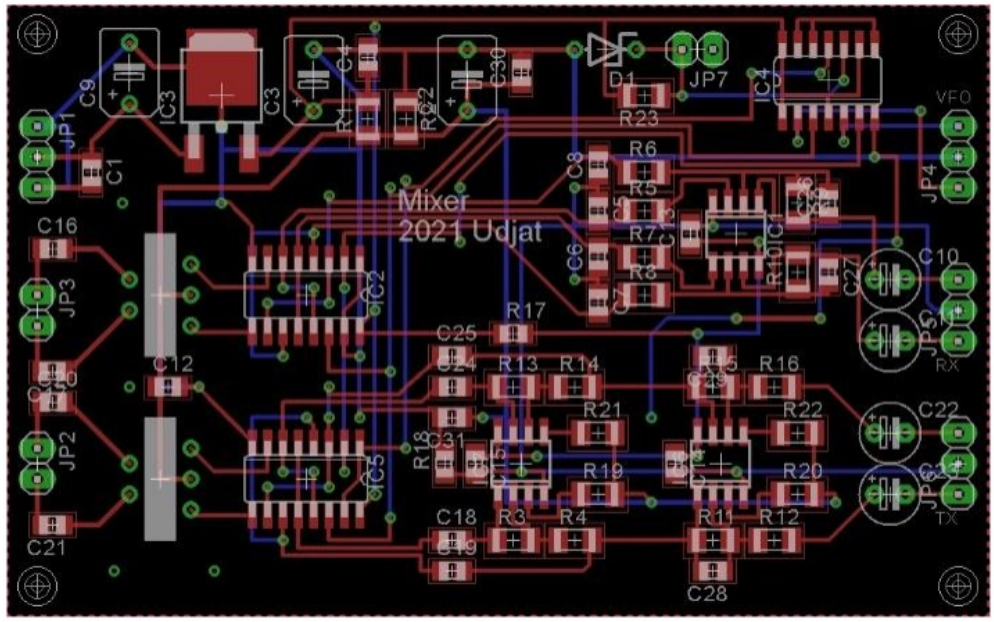
De tweede schakeling wordt gebruikt in de kabel die de pc verbindt met uSDR-Pico en dient als galvanische scheiding; Pc's kunnen nogal wat rommel produceren... De trimmers kunnen worden gebruikt om de signaalsterkte naar wens aan te passen.

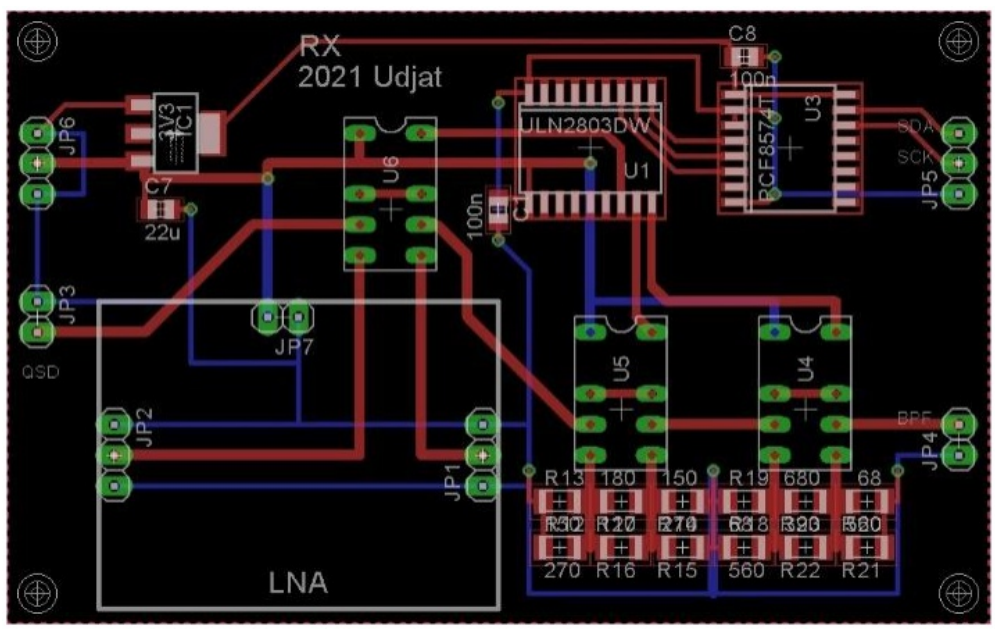
The second circuit is used in the cable that connects PC with uSDR-Pico, and serves as galvanic separation; PCs can be rather noisy… The trimmers can be used to adjust the required levels.

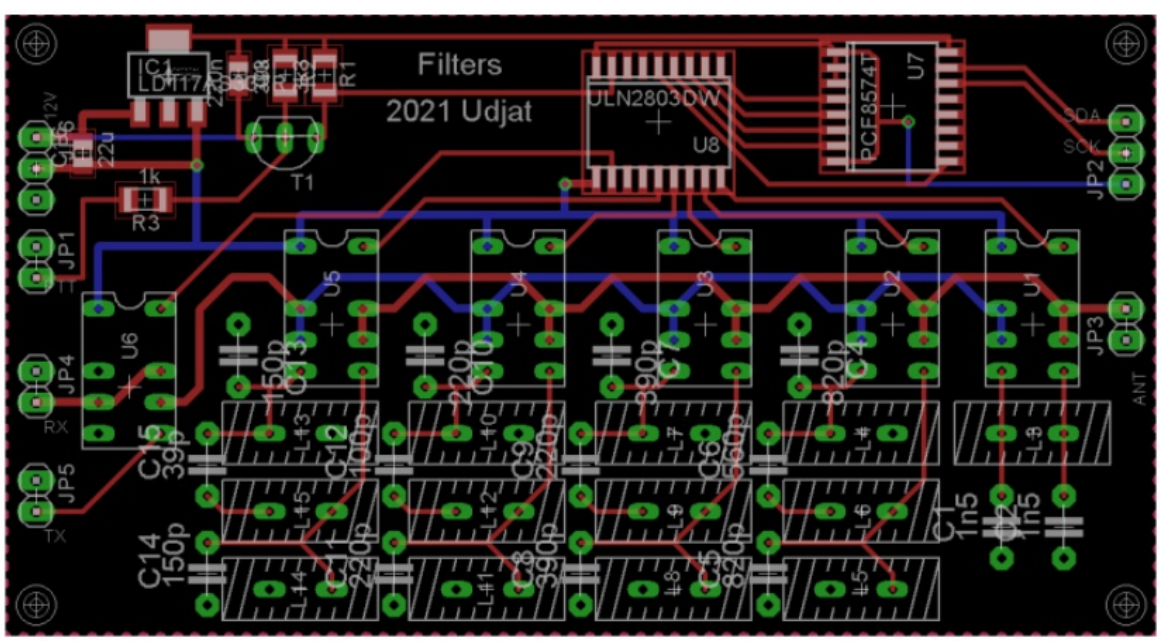
4.7 PCB layout

4.7 PCB layout




Mixer
2021 Udjat
VFO
RX
TX
JP1
JP2
JP3
JP4
JP5
JP7
D1
IC1
IC2
IC3
IC4
IC5
C1
C3
C4
C5
C6
C7
C8
C9
C10
C12
C16
C18
C21
C22
C25
C28
C31
R1
R2
R3
R4
R5
R6
R7
R10
R11
R12
R13
R14
R16
R17
R18
R19
R20
R21
R22
R23

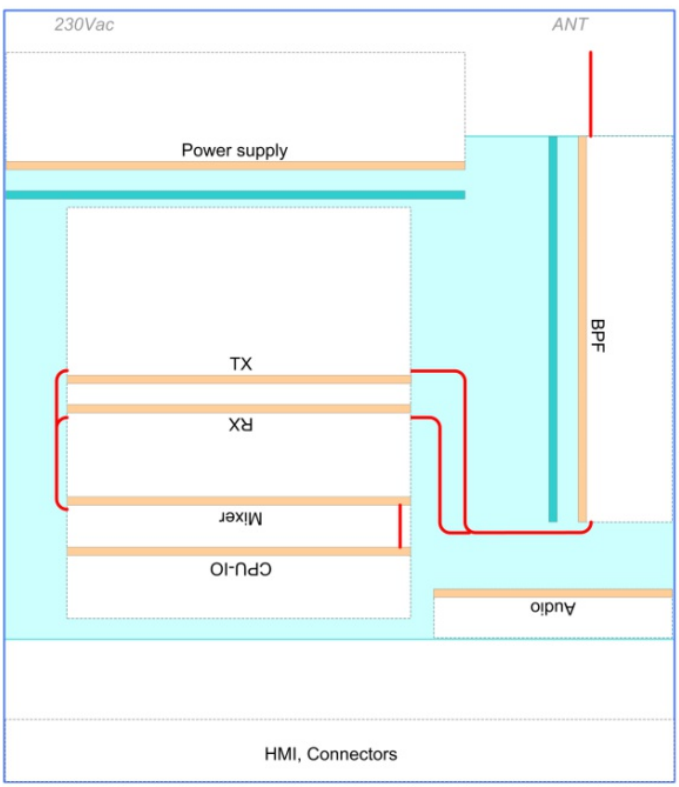

RX
2021 Udjat
3V3
IC1
ULN2803DW
U1
PCF8574T
U3
U4
U5
U6
SDA
SCK
JP1
JP2
JP3
JP4
JP5
JP6
JP7
C7
22u
100n
C8
QSD
BPF
LNA
R13 180 150 R19 680 68
270 R16 R15 560 R22 R21


Filters
2021 Udjat
IC1
LD1117
T1
R1
R3
1k
22u
ULN2803DW
U8
PCF8574T
U7
SDA
SCK
JP1
JP2
JP3
JP4
JP5
JP6
RX
TX
ANT
C13 150p
C10 220p
C9 390p
C4 820p
C15 39p
C12 100p
C11 220p
C8 390p
C5 820p
C14 150p
C1 1n5
C2 1n5

4.8 Mechanica

De mechanische constructie komt overeen met de pin-headers/connectoren op de verschillende PCB's voor een optimale bedrading. De enige fout op dit moment zijn de stroom- en I2C-aansluitingen op het BPF-bord, die idealiter aan tegenoverliggende zijden van het bord zouden moeten zijn.

De kavel wordt ingebouwd in een Teko Euro93 serie behuizing, type 936. De bodem heeft een gedeeltelijke aluminium bodemplaat, waarop de printplaten worden gemonteerd. De beschikbare ruimte is ongeveer als volgt ingedeeld:

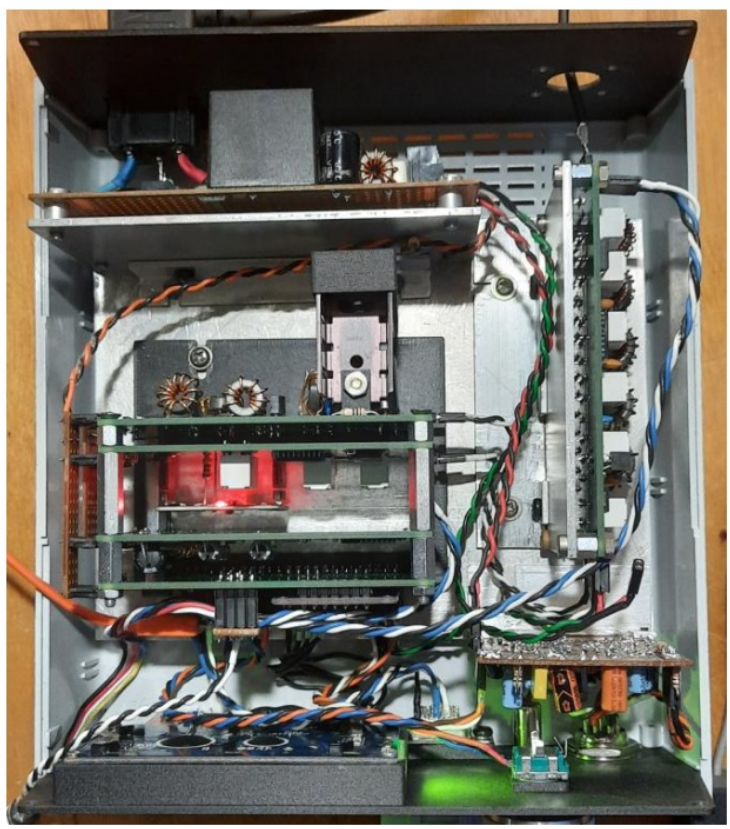


De grijze delen vertegenwoordigen aluminium steunen, die de verschillende PCB's dragen die in tan kleur zijn weergegeven. Component kant is waar de bordnamen staan. Gestreepte witte gebieden geven bij benadering de ruimte aan die door de componenten wordt ingenomen.

4.8 Mechanics

The mechanical construction matches the pin-headers/connectors on the different PCBs for an optimal wiring. The only error at present is the power and I2C connections on the BPF board, which should ideally be on opposite sides of the board.

The lot will be built into a Teko Euro93 series enclosure, type 936. The bottom has a partial aluminum base-plate, onto which the PCBs are mounted. The available space will be organized roughly as follows:



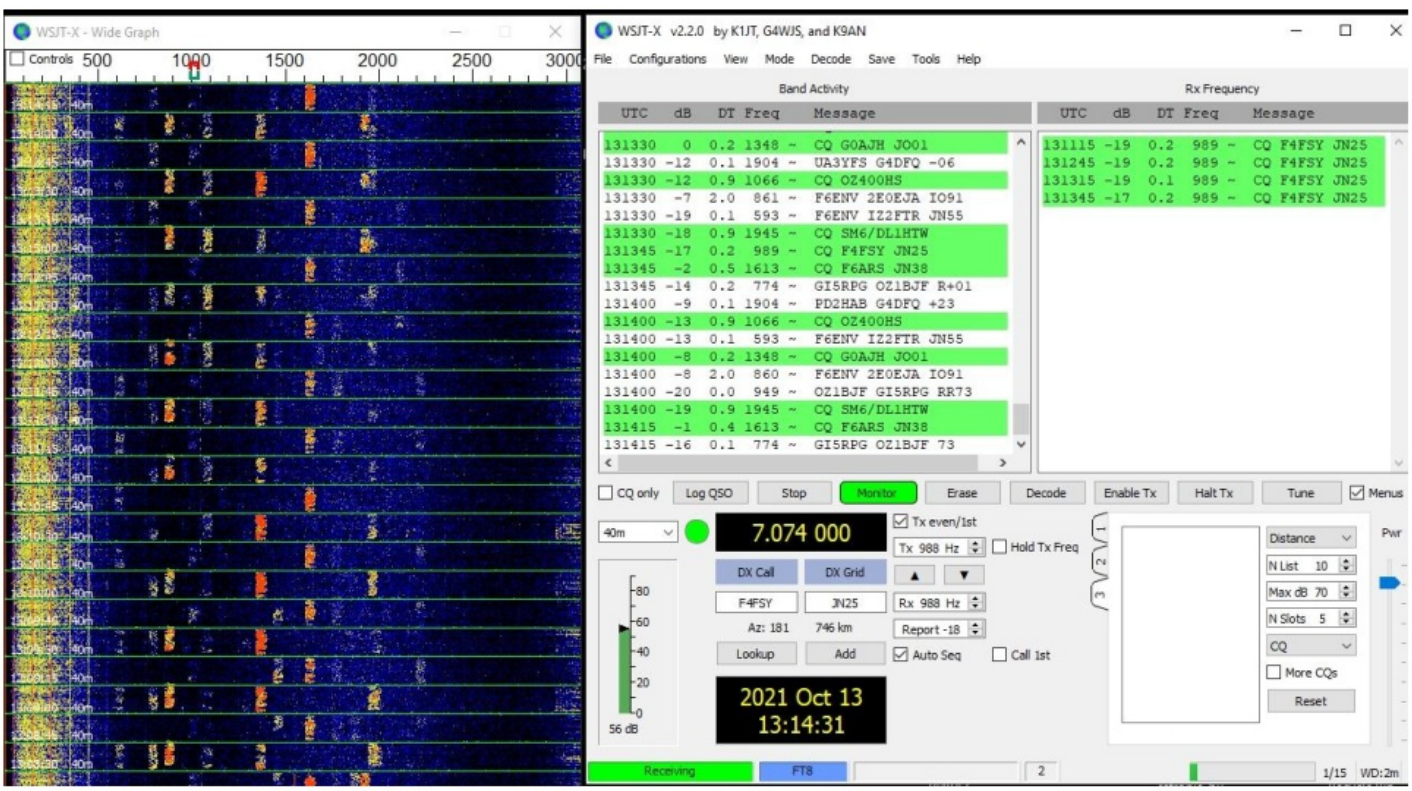
The grey parts represent aluminum supports, which carry the various PCBs shown in tan color. Component side is where the name tag is. Dashed white areas indicate approximate space taken by the components.

5 Testen

De mixer, processor en RX frontend werken prima, ze zijn in ieder geval een voldoende basis voor verdere softwareontwikkeling. BPF en RX/TX frontend zullen binnenkort als prototype worden gebouwd op een PCB. Vanaf nu is het de bedoeling om de audioverwerking van het processorbord te halen en een aparte module te maken die de PTT, de audio-ingangsversterker, de VOX-circuits en een audio-uitgangsbuffer bevat.

5 Testing

The mixer, processor and RX frontend work fine, at least they are a sufficient basis for further software development. BPF and RX/TX frontend are next to be prototyped on a PCB. As of now, the plan is to take the audio processing off the Processor board, and make a separate module that contains the PTT, the audio input amplifier, the VOX circuits and an audio output buffer.



Zoals de screendump laat zien, werkt de huidige versie prima op het 40m FT8-kanaal. De AGC werkt nu ook, zij het nog niet bestuurbaar vanuit het menu. Het zorgt ervoor dat het uitgangssignaal tussen 1.6Vpp en 3.3Vpp wordt gehouden.

As the screen-dump shows, the current version works nicely on the 40m FT8 channel. The AGC now also works, albeit not yet controllable from the menu. It makes sure that the output signal is maintained between 1.6Vpp and 3.3Vpp.

In de V2.0 werken de menu's voor bandfilterselectie, voorversterker of verzwakker, modulatiemodus en AGC, evenals een eenvoudige commandoshell op de seriële poort. Deze poort moet worden aangesloten op een pc met een rechte RS232-kabel.

In the V2.0 the menus for band filter selection, pre-amp or attenuator, modulation mode and AGC are working, as well as a simple command shell on the serial port. This port needs to be connected to a PC with a straight RS232 cable.

De QSE-kant lijkt een beetje ingewikkelder. Een

The QSE side seems a bit more complicated. An

modulatie van de bovenste zijband met een enkele toon zou een constante draaggolf-frequentie moeten opleveren die verschoven is door de toonfrequentie. Dit is een leuke manier om te controleren of de signalen in de modulatieketen in orde zijn (zie dsp.c). Een andere test is met twee tonen, die moeten verschijnen als een draaggolf die in amplitude wordt gemoduleerd met het frequentieverschil tussen de tonen. De draaggolf en de tegenoverliggende zijband moeten echter worden onderdrukt, daarom is het signaal niet precies AM.

upper sideband modulation of a single tone should give a constant carrier frequency up-shifted by the tone frequency. This is a nice way to check whether the signals in the modulation chain are in order (see dsp.c). Another test is with two tones, which should show up as a carrier that is modulated in amplitude with the frequency difference between tones. The carrier and opposite sideband should be suppressed though, hence the signal is not exactly AM.

Met deze achtergrondinformatie kan de output van de mixer worden getest. De schakeling zoals hierboven weergegeven lijkt wat problemen te hebben, hoewel een enkele toon en ook FT8 kunnen worden gedetecteerd.

The output from the mixer can be tested with this background information. The circuit as presented above seems to have some issues, although a single tone and also FT8 can be detected.

Annex A: Frequency domain processing

5.1 Overzicht

Een andere benadering is het gebruik van signaalverwerking in het frequentiedomein. Dit kan worden bereikt door de stroom van op tijd gebaseerde samples om te zetten in een spectrum. Door vast te houden aan de sample-frequentie die wordt gebruikt in de uSDR v2.0-implementatie, 15625 op integers gebaseerde complexe (I,Q) getallen per seconde, kan dit worden opgedeeld in brokken van de juiste grootte voor FFT. De brokgrootte wordt bepaald door de spectrumresolutie die moet worden bereikt. Wanneer bijvoorbeeld een FFT met een lengte van 1024 wordt gebruikt, is de frequentie-resolutie de samplefrequentie gedeeld door de FFT-lengte (d.w.z. 15 Hz). Het spectrum van 1024 punten dat resulteert uit de FFT bevat frequenties van DC tot de helft van de samplefrequentie (Nyquist).

De meeste signaalverwerking kan dan plaatsvinden in het frequentiedomein, waar bijvoorbeeld een banddoorlaatfilter kan worden geïmplementeerd als een venster over een deel

5.1 Overview

Another approach is to use frequency domain signal processing. This can be accomplished by transforming the stream of time-based samples into a spectrum. Sticking to the sample rate used in the uSDR v2.0 implementation, 15625 integer based complex (I,Q) numbers per second, this could be chopped up into suitably sized chunks for FFT. The chunk size is determined by the spectrum resolution that needs to be achieved. When for example a 1024 length FFT is used, the frequency resolution will be the sample rate divided by the FFT length (i.e. 15Hz). The 1024 point spectrum resulting from the FFT holds frequencies from DC up to half the sample rate (Nyquist).

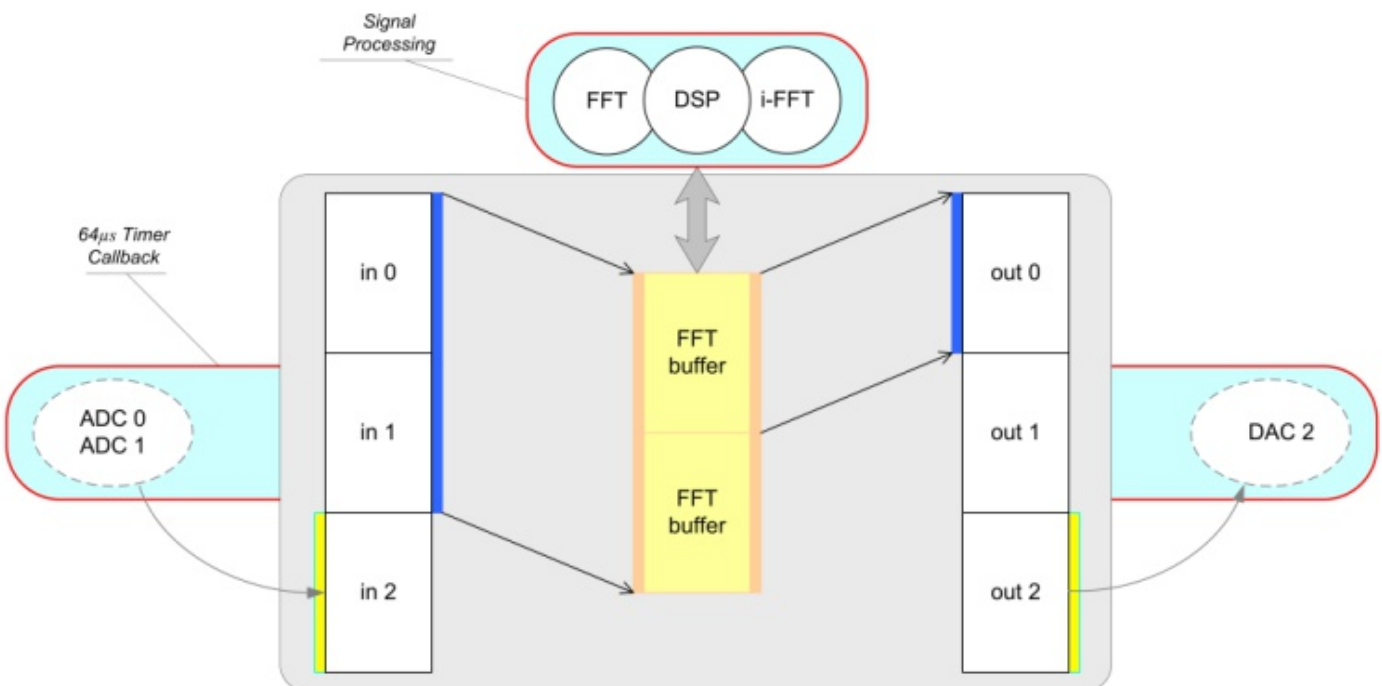
Most signal processing can then be done in the frequency domain, where for example a bandpass filter can be implemented as a window over a portion of the spectrum. The time

van het spectrum. Het tijdsdomein kan uiteindelijk worden hersteld door een inverse-FFT toe te passen, wat resulteert in een verwerkte (complexe) sample stroom met de oorspronkelijke invoersnelheid van 15625.

domain can finally be restored by applying an inverse-FFT, resulting in a processed (complex) sample stream at the original input rate of 15625.

Om problemen met aliasing aan te pakken, worden de transformaties uitgevoerd met enige overlap, bijvoorbeeld elke 512 samples voor een lengte van 1024 FFT. Dit betekent dat elke 512/15625=32 msec de 1024-punts FFT, de signaalverwerking en de iFFT moeten worden uitgevoerd plus wat overhead om de gegevens te beheren en de beoogde output te creëren.

To address aliasing issues the transformations are done with some overlap, for example every 512 samples for a 1024 FFT length. This means that every 512/15625=32 msec the 1024-point FFT, the signal processing and the iFFT need to be executed plus some overhead to manage the data and create the intended output.



Om de snelheid te optimaliseren, zal de implementatie vaste-komma berekeningen gebruiken, aangezien de Pico geen FPU heeft. Voor het bufferen van de data zijn drie invoerbuffers van 512 samples vereist, waarbij één buffer wordt gevuld door de ADC terwijl de andere twee worden gebruikt als FFT-invoer. Hetzelfde geldt voor de uitvoer, waar twee buffers het doel zijn van de i-FFT, terwijl de derde wordt gebruikt om de DAC-uitvoersamples te bewaren. Tussentijds is een 2x512-samplebuffer vereist om het spectrum vast te houden. De buffertoewijzingen worden elke 512 samples gewijzigd. Wetende dat elk sample een 2x16-bit complexe waarde bevat, brengt dit het totale RAM-gebruik op 8x512x2x2=16kBytes, wat gemakkelijk in de

In order to optimize for speed, the implementation will use fixed-point arithmetic, since the Pico features no FPU. For buffering the data, three 512-sample input buffers are required, where one buffer is being filled by the ADC while the other two are used as FFT input. Likewise for the output, where two buffers are target of the i-FFT, while the third is used to hold the DAC output samples. Intermediately, a 2x512 sample buffer is required to hold the spectrum. The buffer assignments are changed every 512 samples. Knowing that each sample contains a 2x16-bit complex value, this brings the total RAM use to 8x512x2x2=16kBytes, which should easily fit into the available 256kB.

beschikbare 256kB zou moeten passen.

5.2 Fysieke betekenis van de FFT

Een HF-signaal wordt gemengd in een kwadratuurmixer, wat resulteert in een in-fase (I) en een kwadratuur (Q) signaal, gecentreerd rond DC. Dus wat betekent dit, bijvoorbeeld om negatieve frequenties te hebben? Als je bedenkt dat het originele HF-signaal twee zijbanden heeft door amplitudemodulatie, zijn de zijbanden net iets hoger of lager dan de draaggolf-frequentie. Wanneer je dit signaal omlaag mengt met de draaggolffrequentie, wordt de onderste zijband als wiskundig resultaat weergegeven als een negatieve frequentie.

Stel dat het omlaag gemixte signaal wordt weergegeven door de tijdafhankelijke vector (I_t,Q_t). De rotatiesnelheid van de vector vertegenwoordigt de frequentie, de rotatie-richting geeft aan of de frequentie positief of negatief is. De werkelijke beweging van de vector is grillig en bevat een hele reeks frequenties. Met de fouriertransformatie willen we eigenlijk voor deze set van snelheden (frequenties) analyseren in hoeverre deze aanwezig zijn in de (I_t,Q_t)-stroom.

Hiertoe worden de I- en Q-stromen geconver-teerd in discrete (I_k, Q_k) sampleparen, die de complexe tijdsdomeininvoer voor de FFT zijn. Op regelmatige tijdstippen worden de laatste N-samples (d.w.z. de FFT-grootte) omgezet in een frequentiespectrum. Dit transformatie-interval moet korter zijn dan wat wordt weergegeven door N-monsters, dus er is enige overlap.

2N *real*-time samples zouden resulteren in N *complexe* frequenties (zie Nyquist), waarbij de ontbrekende negatieve complexe frequenties slechts een spiegel zijn van de N positieve frequentiesegmenten. Daarentegen resulteren 2N *complexe* tijdmonsters in N positieve + N negatieve *complexe* frequenties.

Het segment met index 0 vertegenwoordigt de DC-component en het segment met index N-1

5.2 Physical meaning of the FFT

An RF signal is mixed down with a quadrature mixer, resulting in an in-phase (I) and a quadrature (Q) signal, centered on DC. So what does this mean, for example to have negative frequencies? If you consider the original RF signal having two sidebands from amplitude modulation, the sidebands are just a bit higher or lower than the carrier frequency. When you mix this signal down with the carrier frequency, the lower sideband as a mathematical consequence is represented by a negative frequency.

Suppose that the mixed down signal is represented by the time dependent vector (I_t,Q_t). The rotation speed of the vector represents the frequency, the rotation direction represents whether the frequency is positive or negative. The actual movement of the vector is erratical, and contains a whole set of frequencies. With the fourier transform we actually want to analyze for this set of speeds (frequencies) to what extent these are present in the (I_t,Q_t) stream.

To this purpose, the I and Q streams are converted in discrete (I_k,Q_k) sample pairs, which are the time-domain complex input to the FFT. At regular moments in time the latest N samples (i.e. the FFT size) are transformed into a frequency spectrum. This transformation interval has to be shorter than what is represented by N samples, so there is some overlap.

2N *real* time samples would result in N *complex* frequencies (cf. Nyquist), where the missing negative complex frequencies are just a mirror of the N positive frequency bins. In contrast, 2N *complex* time samples result in N positive + N negative *complex* frequencies.

The bin with index 0 represents the DC component, and the bin with index N-1

vertegenwoordigt de Nyquist-frequentie. De segmenten N en verder vertegenwoordigen de negatieve frequenties.

Voor representatie zal het roteren van de set frequentiesegmenten met N de DC-component in segment N plaatsen en de bovenste/onderste zijbanden aan weerszijden.

Demodulatie van SSB zou neerkomen op het wegfilteren van alles behalve de ongeveer 3 kHz aan de rechterkant van het DC-segment voordat terug naar het tijdsdomein wordt getransformeerd. Aangezien de bovenste en onderste zijbanden verschillend zijn, is het duidelijk dat complexe tijdsamples nodig zijn om de juiste transformatie te verkrijgen.

Om ruis rond DC te verwijderen, zou de mengfrequentie lager kunnen worden gekozen dan de werkelijke HF-draaggolf, waardoor het basisbandsignaal ergens in het midden van de positieve frequentiesegmenten terechtkomt, d.w.z. rond N/2. Na filtering kan de basisband worden verschoven met N/2 voordat de inverse FFT wordt toegepast.

5.3 De FFT-wiskunde

De Fast Fourier Transform is een geoptimaliseerde implementatie van een Discrete Fourier Transformatie. Deze transformatie verandert een reeks tijdsdomein-samples in een frequentiespectrum.

De DFT-vergelijking wordt gegeven door:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}nk}$$

Voor de inverse operatie (i-DFT) geldt een vergelijkbare vergelijking:

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi}{N}nk}$$

De DFT vertegenwoordigt een vermenigvuldiging van een N-grootte tijdsamplesvector met een N2-grootte vierkante matrix, wat resulteert in een andere N-grootte frequentiespectrum-

represents the Nyquist frequency. The bins N and beyond represent the negative frequencies.

For representation, rotating the set of frequency bins with N will place the DC component at bin N and the upper/lower sidebands on either side.

Demodulation of SSB would boil down to filtering away everything but the 3kHz or so to the right of the DC bin before transforming back to the time domain. Since Upper and lower sidebands are different, it is clear that complex time samples are required to obtain the right transformation.

To get rid of noise around DC, the mixing frequency could be chosen lower than the actual RF carrier, causing the baseband signal to land somewhere in the middle of the positive frequency bins, i.e. around N/2.Then after filtering, the baseband can be shifted down by N/2 before applying the inverse FFT.

5.3 The FFT mathematics

The Fast Fourier Transform is an optimized implementation of a Discrete Fourier Transform. This transform changes a series of time-domain samples into a frequency spectrum.

The DFT equation is given by:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}nk}$$

For the inverse operation (i-DFT) a similar equation applies:

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi}{N}nk}$$

The DFT represents a multiplication of a size-N time samples vector with a size-N2 square matrix, resulting in another size-N frequency spectrum vector. The samples are in fact the I

vector. De samples zijn in feite de I- en Q-streams afkomstig van de QSD en kunnen worden weergegeven als:

$$x_n = I_n + j \cdot Q_n$$

De complexe coëfficiënten van de DFT kunnen worden weergegeven als:

$$e^{-j\frac{2\pi}{N}nk} = cos\left(\frac{2\pi}{N}nk\right) - j \cdot sin\left(\frac{2\pi}{N}nk\right)$$

In deze complexe coëfficiënten vertegenwoordigt **n** de tijd, die één element van de samplevector aangeeft, terwijl **k** de frequentie vertegenwoordigt, die één element van de spectrumvector aangeeft. Toenemende **k** betekent toenemende frequentie, d.w.z. het aantal volledige fasegolven dat binnen de totale sampleperiode past. De factor **n/N** stapt in de tijd door deze periode. Daarom wordt voor elke mogelijke frequentie het interne product tussen een matrixrij en de samplevector berekend.

Elke *complexe* vermenigvuldiging in dat interne product bestaat in feite uit 4 vermenigvuldigingsacties en 2 optellingen:

$$(I_n + j \cdot Q_n) \cdot \left(cos\left(\frac{2\pi}{N}nk\right) - j \cdot sin\left(\frac{2\pi}{N}nk\right)\right)$$
$$= \left(I_n \cdot cos\left(\frac{2\pi}{N}nk\right) + Q_n \cdot sin\left(\frac{2\pi}{N}nk\right)\right) + j \cdot \left(Q_n \cdot cos\left(\frac{2\pi}{N}nk\right) - I_n \cdot sin\left(\frac{2\pi}{N}nk\right)\right)$$

Dus om één element van de spectrumvector **X(k)** te verkrijgen, moet deze complexe vermenigvuldiging **N** keer worden uitgevoerd naast ook nog **2N** optellingen.

In het geval van ***uSDR*** zowel **n** als **k ϵ [0, N-1]** en dus resulteert dit in **N²** complexe vermenigvuldigingen en optellingen. De FFT-implementatie van de DFT maakt echter gebruik van de inherente symmetrieën in de bewerking om dubbele vermenigvuldigingen te voorkomen.

Deze symmetrieën komen voort uit de periodiciteit van de **sin()** en **cos()** functies in de DFT-coëfficiënten, waar een halve periode van faseverschuiving alleen maar resulteert in een omslag van het teken. Herhaaldelijk toepassen van deze optimalisatie voor alle **N** elementen van de samplevector reduceert het aantal

and Q streams coming from the QSD, and can be represented as:

$$x_n = I_n + j \cdot Q_n$$

The complex coefficients of the DFT can be represented as:

$$e^{-j\frac{2\pi}{N}nk} = cos\left(\frac{2\pi}{N}nk\right) - j \cdot sin\left(\frac{2\pi}{N}nk\right)$$

In these complex coefficients **n** represents the time, indicating one element of the sample vector, while **k** represents the frequency, indicating one element of the spectrum vector. Increasing **k** means increasing frequency, i.e. the number of full phase waves that fit inside the total sample period. The factor **n/N** steps timewise through this period. Hence the internal product between a matrix row and the sample vector is calculated for each possible frequency.

Each *complex* multiplication in that internal product in fact consists of 4 multiply actions and 2 additions:

So, to obtain one element of the spectrum vector **X(k)** this complex multiplication has to be performed **N** times as well as **2N** more additions.

In case of ***uSDR*** both **n** and **k ϵ [0, N-1**] and hence this results in **N²** complex multiplications and additions. However, the FFT implementation of the DFT utilizes the inherent symmetries in the operation to avoid doing duplicate multiplications.

These symmetries originate from the periodicity of the **sin()** and **cos()** functions in the DFT coefficients, where half a period of phase shift just results in a sign flip. Repeatedly applying this optimization for all **N** elements of the samples vector reduces the number of multiplications per element of the resulting

vermenigvuldigingen per element van de resulterende spectrumvector van **N** tot **^{2}log(N)** en reduceert daardoor het totale aantal tot **N·^{2}log(N)**. Om dit te illustreren: voor een array met een grootte van 1024 zou de CPU-belasting voor het berekenen van een FFT ongeveer 100 keer minder zijn dan voor een DFT.

Een recursieve FFT-implementatie zou kunnen worden toegepast door de invoersample-array herhaaldelijk in tweeën te splitsen met een vast faseverschil. Een meer realistische implementatie voor ***uSDR*** zal echter niet recursief zijn om te voorkomen dat er te veel cycli worden besteed aan het verplaatsen van gegevens. In plaats daarvan wordt het buffermechanisme gebruikt zoals beschreven in het overzicht, dergelijke implementaties worden in-place genoemd. De invoerarray wordt gekopieerd naar de uitvoerlocatie en vervolgens wordt de opslag van die kopie gebruikt om de daadwerkelijke transformatie in-place uit te voeren.

Er zijn verschillende voorbeelden van fixed-point FFT-implementaties in C te vinden, maar elk voorbeeld moet worden geoptimaliseerd en aangepast aan de specifieke doelomgeving. Voor ***uSDR*** is de eerste go gebaseerd op **fix_fft.c**, oorspronkelijk geschreven in 1989 door Tom Roberts, maar sindsdien verschillende keren verbeterd door anderen. Een andere goede bron is fxtbook.pdf, te vinden op internet. Dit laatste algoritme wordt gebruikt in ***uSDR***.

5.4 FFT-implementatie

De eerste fase van het FFT-algoritme is het opnieuw ordenen van de samples; om precies te zijn, de samples met indexen die bit-reverse van elkaar zijn, moeten worden verwisseld. De arraygrootte is 1024 samples, dus de index is 10 bits. De swap is dan bijvoorbeeld tussen samples [1111000001b] en [1000001111b]. Deze herordening wordt gedaan voordat de FFT wordt berekend, en daarom Decimation in Time (DIT) genoemd, in tegenstelling tot de post-FFT DIF. De herschikking is nodig om een efficiënte keten van butterfly executies mogelijk te maken.

spectrum vector from **N** down to **^{2}log(N)** and by doing so reduces the total number to **N·^{2}log(N)**. To illustrate this, for a 1024 size array the CPU burden for calculating a FFT would be approximately 100 times less than for a DFT.

A recursive FFT implementation could be applied by repeatedly splitting the input sample array in halves with a fixed phase difference. A more realistic implementation for ***uSDR*** is not going to be recursive however, in order to prevent spending too many cycles on moving data around. Instead, the buffer mechanism as described in the overview is used, such implementations are referred to as in-place. The input array is copied to the output location and then the storage of that copy is used to perform the actual transformation, in-place.

Several examples of fixed-point FFT implementations in C can be found, but any example needs to be optimized and adapted to the specific target environment. For ***uSDR*** the first go is based on **fix_fft.c**, originally written in 1989 by Tom Roberts but since then improved several times by others. Another good source is fxtbook.pdf, to be found on the internet. This latter algorithm is used in ***uSDR***.

5.4 FFT implementation

The first stage of the FFT algorithm is the reordering of the samples; to be precise, the samples with indexes that are bit-reverse of each other need to be swapped. The array size is 1024 samples, so the index is 10 bits. The swap is then for example between samples [1111000001b] and [1000001111b]. This re-ordering is done before the FFT is calculated, and therefore named Decimation in Time (DIT), as opposed to the post-FFT DIF. The re-ordering is needed to enable an efficient chain of butterfly executions.

De crux van het algoritme is hoe je de array maar één keer kunt doorlopen, d.w.z. voorkomen dat je samples weer terugwisselt. Een algemene benadering zou zijn:

```
for (i=0; i<1024; i++)
{
   j = bit_reverse(i);
   if (i < j)
      swap(data[i], data[j]);
}
```

maar de bit_reverse-routine kan behoorlijk wat tijd in beslag nemen. Een sneller alternatief (gebruik makend van de relatief overvloedige RAM) is om in plaats daarvan een opzoektabel te gebruiken. Daarna is het verwisselen van de monsters eenvoudig.

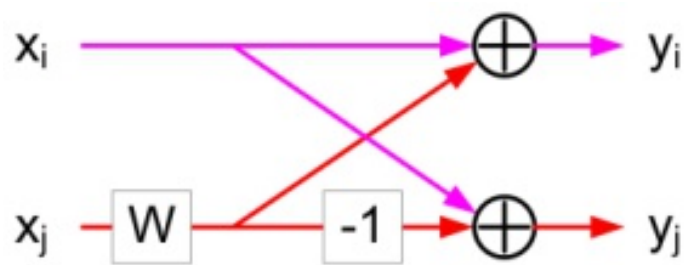
The crux of the algorithm is how to go through the array only once, i.e. avoid swapping samples back again. A general approach would be:

```
for (i=0; i<1024; i++)
{
   j = bit_reverse(i);
   if (i < j)
      swap(data[i], data[j]);
}
```

but the bit_reverse routine could take quite some time. A faster alternative (utilizing the relatively abundant RAM) is to use a lookup table instead. After that, the swapping of the samples is straightforward.

De tweede fase bestaat uit een aantal geneste lussen, het berekenen en toevoegen van de butterflies. Een dergelijke butterfly kan als volgt worden weergegeven:

The second stage consists of a number of nested loops, calculating and adding the butterflies. One such butterfly can be represented as follows:



Eén (complexe) invoer wordt vermenigvuldigd met een "Wiggle-factor" en afgetrokken van of opgeteld bij de andere invoer. Dit wordt herhaald over de volledige array in ^{2}log(N) opeenvolgende fasen, waarbij de combinaties en W-factoren per fase veranderen. Het resultaat van elke fase wordt op dezelfde locatie opgeslagen, vandaar de notatie "in-place".

One (complex) input is multiplied with a “Wiggle factor” and either subtracted from or added to the other input. This is repeated over the complete array in ^{2}log(N) subsequent stages, where the combinations and W factors change per stage. The result of each stage is stored in the same location, hence the notation “in-place”.

5.5 Volgorde

5.5.1 ADC lezen en DAC schrijven

De ADC's draaien in RR-mode, net als in de uSDR-implementatie van het tijdsdomein. Het verschil is dat na 3 samples de conversies tijdelijk worden gestopt. Dit betekent dat voor 3x 2usec cycli conversies worden uitgevoerd voor

5.5 Sequencing

5.5.1 ADC read and DAC write

The ADCs are running in RR mode, just like in the time domain uSDR implementation. The difference is that after 3 samples the conversions are temporarily stopped. It means that for 3x 2usec cycles conversions are done

ADC-kanalen 0..2, en het in de FIFO wordt geschoven, waarbij een IRQ op niveau 3 wordt gezet. De ADC-FIFO-interrupt-handler stopt de conversies, die opnieuw worden gestart door de timer-callback-routine. Op deze manier is de samplefrequentie 16,625 kHz.

De samples worden gekopieerd van de ADC FIFO en opgeslagen in de bijbehorende samplebuffer door deze timer-callback. De callback-routine verwerkt ook de uitvoer naar de DAC's vanuit de gerelateerde samplebuffer. Wanneer een halve FFT_SIZE buffer vol is, wordt de DSP-loop aangeroepen, die de FFT-, DSP- en iFFT-bewerkingen uitvoert gedurende een ½ FFT-grootte x 64usec interval, en dat is 32,768 msec voor een 1024 FFT-grootte.

Het signaal moet worden versterkt om het dynamische FFT-bereik te vullen, d.w.z. tot ongeveer het bereik van int16_t. Hiervoor wordt een gemiddelde signaalsterkte-indicator berekend, zodat een vermenigvuldigingsfactor (zeg maar AGC) kan worden afgeleid. Normaal gesproken is het ingangssignaal minder dan 1 Vpp, wat 1/3 is van het ADC-bereik van 12 bits, dus de vermenigvuldiger zal ongeveer 100 zijn.

Een soortgelijk proces moet worden toegepast op de uitvoer, omdat het DAC-bereik slechts 8 bits is. Realistisch gezien blijft er slechts een deel van het signaal over na verwerking, dus moet er opnieuw een signaalsterkte-indicator worden bijgehouden om een vermenigvuldiger te berekenen.

Voor een relatief gelijkmatig signaal kan de piek-signaalsterkte-indicator worden benaderd door 2x de gemiddelde absolute waarde te nemen.

5.5.2 <u>Bufferverwerking</u>

De bufferstructuur is opgebouwd uit buffers van ½ FFT-formaat, met het dubbele voor de complexe kant van de verwerking. De Overlap-Add methode wordt gebruikt om te zorgen voor een soepele samenvoeging van de opgeknipte samplestromen.

for ADC channels 0..2 , and the result is shifted into the FIFO, flagging an IRQ at level 3. The ADC-FIFO interrupt handler will stop the conversions, which will be restarted by the timer callback routine. This way the sample frequency is 16.625 kHz.

The samples are copied from the ADC FIFO and stored in the related sample buffer by this timer callback. The callback routine also handles the output to the DACs from the related sample buffer. When a half FFT_SIZE buffer is full, the DSP-loop is signaled, which performs the FFT, DSP and iFFT operations during a ½ FFT-size x 64usec interval, which is 32.768msec for a 1024 FFT-size.

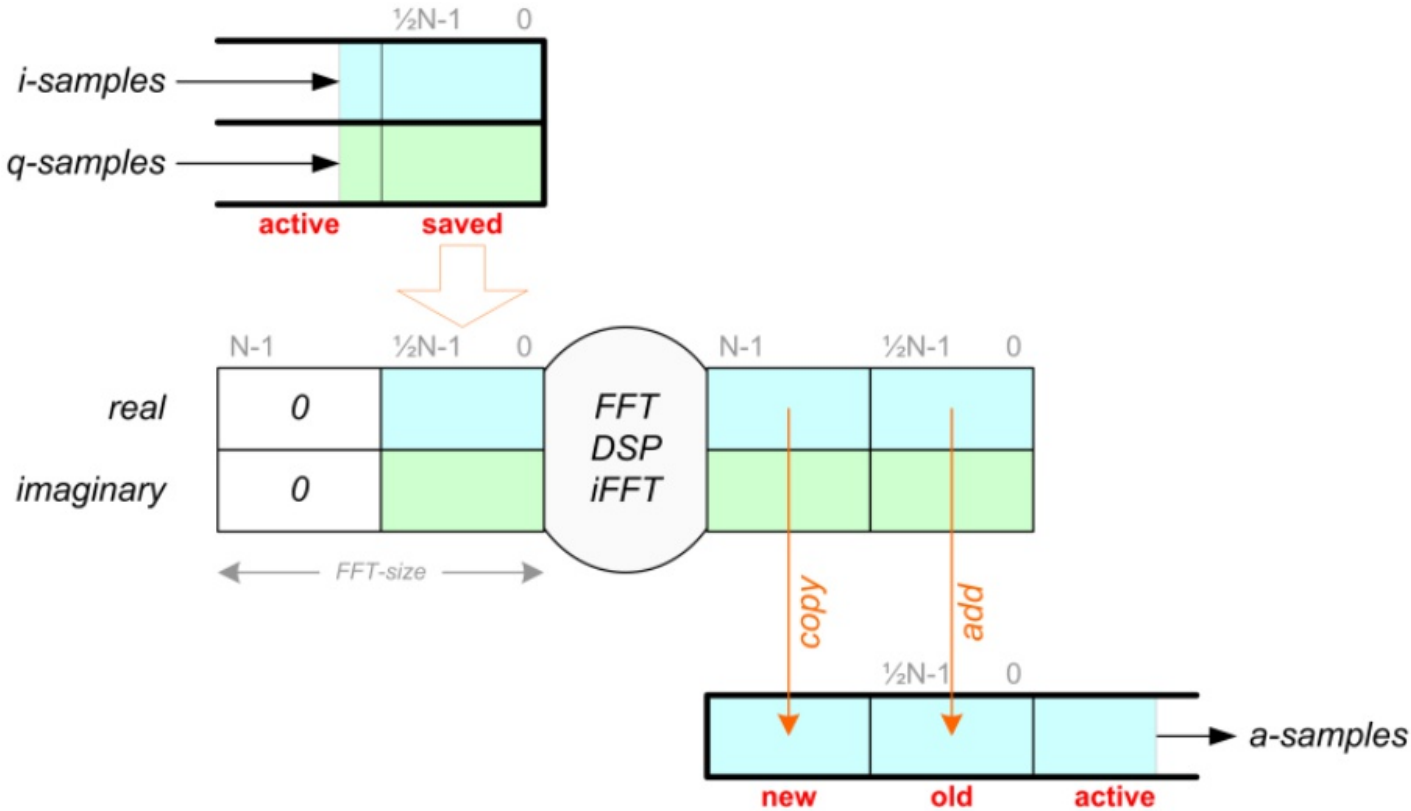
The signal should be amplified to fill the FFT dynamic range, i.e. up to about the range of int16_t. To do this, an average signal strength indicator is calculated so that a multiplier (call it an AGC) can be deduced. Normally the input signal is less than 1Vpp, which is 1/3 of the ADC range of 12 bits, so the multiplier will be approximately 100.

A similar process must be applied to the output, because the DAC range is only 8 bits. Realistically only a part of the signal remains after processing, so again a sgnal strength indicator must be maintained to calculate a multiplier.

For a relatively smooth signal the peak signal strength indicator can be approximated with 2x the average absolute value.

5.5.2 <u>Buffer handling</u>

The buffer structure is built up from ½ FFT-size buffers, doubled for the complex side of processing. The Overlap-Add method is used to ensure a smooth glueing of the chopped-up sample streams.



De figuur toont de RX-toestand weer, de toegewezen buffers worden in feite hergebruikt maar werken in tegengestelde richting t.o.v. van de TX-mode. De actieve interfacebuffer is een van een 2-bufferwachtrij, de andere bevat de opgeslagen samples van het vorige interval. De actieve buffers verzamelen de I- en Q-samples die zijn vastgelegd door de timer-callback-routine. Wanneer de actieve buffer vol is, worden de invoer- en uitvoerbuffers gereorganiseerd en wordt de DSP-loop gestart.

Het reële deel van het vorige resultaat van de signaalverwerkingscyclus wordt toegevoegd aan/gekopieerd naar de uitgangsbuffers. Vervolgens worden de opgeslagen invoerbuffers gekopieerd naar de onderste helft van de FFT-buffers, terwijl de bovenste helften met nullen gevuld worden. Vervolgens wordt de signaalverwerkingscyclus gestart, wat een nieuw resultaat oplevert.

5.5.3 Signaalverwerking

De signaalverwerking volgt de volgorde van transformatie naar het frequentiedomein, waarbij de filtering/verschuiving en transformatie terug naar het tijdsdomein wordt toegepast. De

The figure represents the RX case, the allocated buffers are in fact re-used but work in opposite direction for the TX case. The active interface buffer is one of a 2-buffer queue, the other being the saved samples of previous interval. The active buffers collect the I and Q samples captured by the timer callback routine. Whenever the active buffer is full, the input and output buffers are reorganized and the DSP loop is signaled to start.

The real part of the previous signal processing cycle result is added/copied to the output buffers. Then the saved input buffers are copied into the lower half of the FFT buffers, while the upper halves are zero padded. Then the signal processing cycle is begun, yielding a new result.

5.5.3 Signal processing

The signal processing follows the sequence of transformation to the frequency domain, applying the filtering/shifting and transformation back to the time domain. The samples in the

samples in het audiodomein zijn real, dus een conversie van (en naar) de complexe weergave is vereist.

RX-mode:

<<Shift & Filter>>

Om de filtering mogelijk te maken, moet de draaggolffrequentie niet omlaag worden geconverteerd naar 0 Hz, maar naar ergens in het midden van de frequentieband die het resultaat van de FFT is. Met een samplefrequentie van 15,625 kHz zou de offset-frequentie Fc ergens rond de 3,9 kHz moeten liggen. Afhankelijk van de gewenste modulatiemode worden verschillende bereiken met betrekking tot deze offset uitgefilterd en naar de juiste plaats in de spectrumbuffer verschoven. Bijvoorbeeld (alleen het reële spectrum wordt getoond) - zie volgende bladzijde.

Voor AM bevatten de bovenste en onderste zijbanden dezelfde informatie, d.w.z. het spectrum rond de Fc is symmetrisch. Dit houdt in dat de bijbehorende zijbanden kunnen worden gespiegeld en opgeteld wat een versterking van 3 dB oplevert.

Voor CW kan het filter smal zijn rond Fc en de effectieve verschuiving moet worden verminderd met de gewenste toon (bijvoorbeeld 900 Hz). Na de iFFT van dit gefilterde spectrum wordt het reële deel van de complexe tijdsamples gekopieerd naar de audiosample buffer.

TX-mode:

De omgekeerde acties worden uitgevoerd bij zenden. De audiosamples worden gekopieerd naar het reële gedeelte van de FFT-buffer, terwijl het imaginaire deel op 0 gezet wordt. Na het uitvoeren van de FFT wordt het spectrum omhoog geschoven met de offset, wordt het gewenste spectrum eruit gefilterd en de iFFT uitgevoerd. Zowel reële als imaginaire delen worden gekopieerd naar de I- en Q-buffers.

audio domain are real, so a conversion from (and to) the complex representation is required.

RX case:

<<Shift & Filter>>

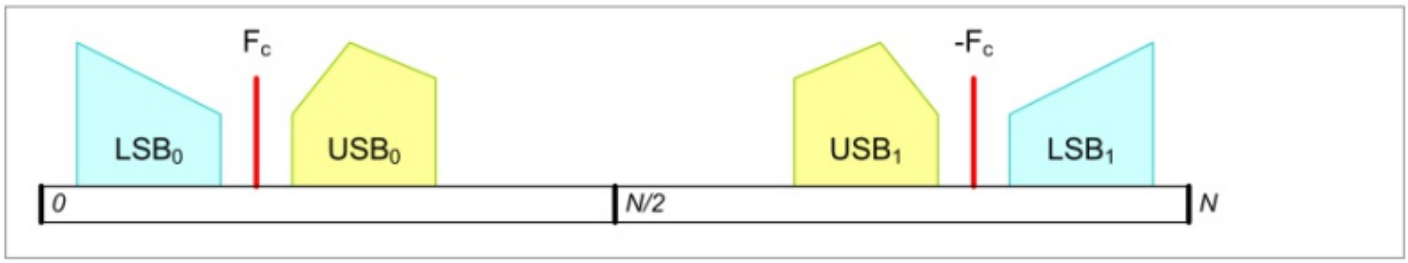
To enable the filtering, the carrier frequency must not be downconverted to 0 Hz, but rather to somewhere in the center of the frequency band resulting from the FFT. With a sampling rate of 15.625kHz, the offset frequency Fc should be somewhere around 3.9kHz. Depending on the desired modulation mode, different ranges with respect to this offset are filtered out, and shifted to the proper place in the spectrum buffer. For example (only real spectum shown) - see next page.

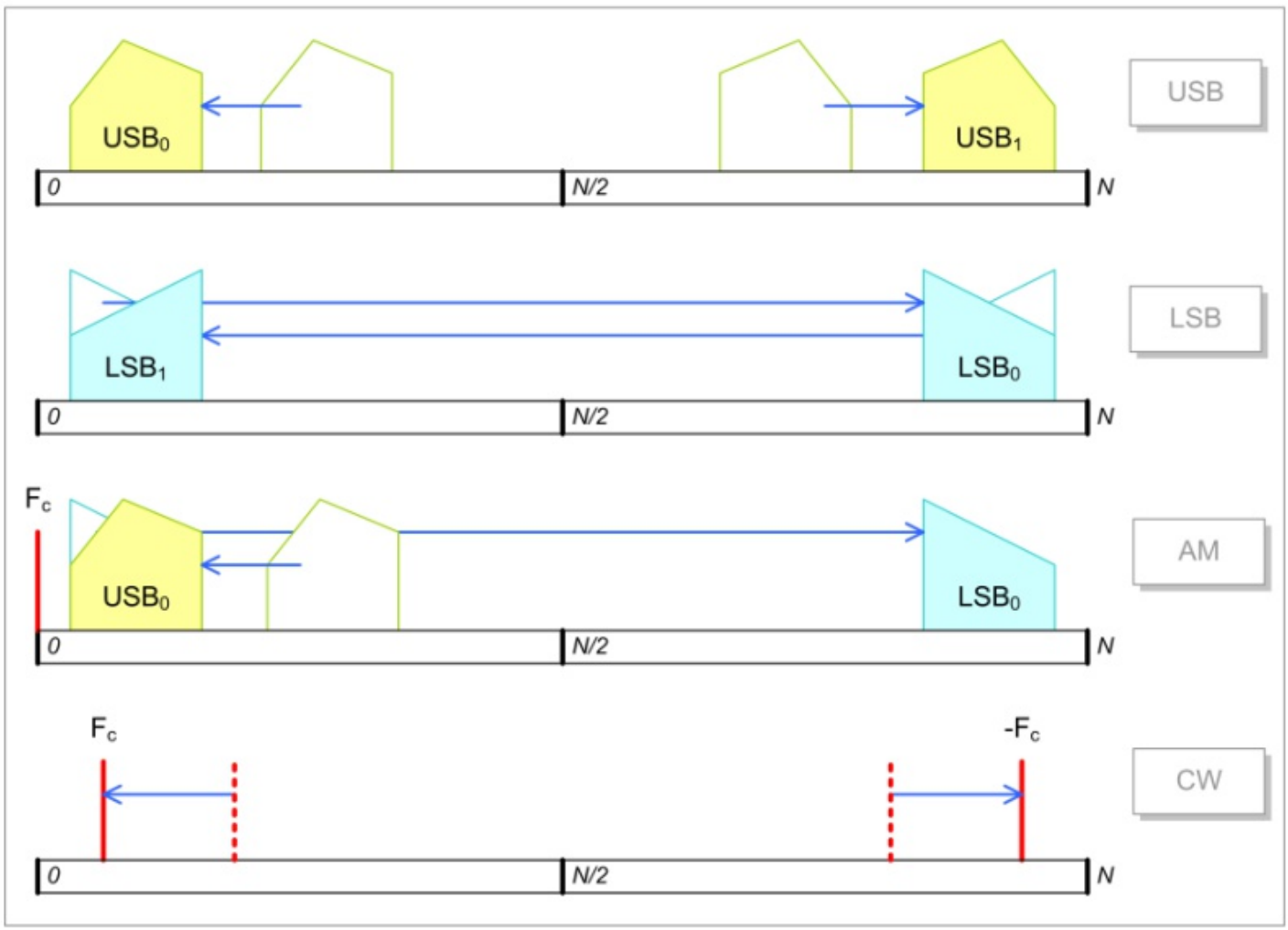
For AM the upper and lower sidebands contain the same information, i.e. the spectrum about the Fc is symmetric. This implies that the corresponding sidebands could be mirrored and added for a 3dB gain.

For CW the filter can be narrow around Fc and the effective shift should be reduced with the desired tone (e.g. 900Hz). After the iFFT of this filtered spectrum, the real part of the complex time samples are copied to the audio samples buffer.

TX case:

The reverse actions are performed for transmission. The audio samples are copied in the real part of the FFT buffer, while the imaginary part is set to 0. Then after performing the FFT shift the spectrum up with the offset, filter out the desired spectrum and do the iFFT. Both real and imaginary parts are copied to the I and Q buffers.





Opmerkingen:

Bij het verschuiven van het spectrum wordt in feite een rotatie gedaan; segmenten die over de rand van de FFT-buffer schuiven, komen aan de andere kant weer binnen.

Bij het toevoegen van een draaggolf om een AM-basisbandsignaal te verkrijgen, moet deze draaggolf twee keer (?) de amplitude van een zijbandsignaal bevatten

Notes:

When shifting the spectrum, in fact a rotation is done; bins that shift beyond the FFT-buffer edge will re-enter on the other side.

When adding a carrier to obtain an AM baseband signal, this carrier should contain twice (?) the amplitude of any sideband signal

Overlap-Add method

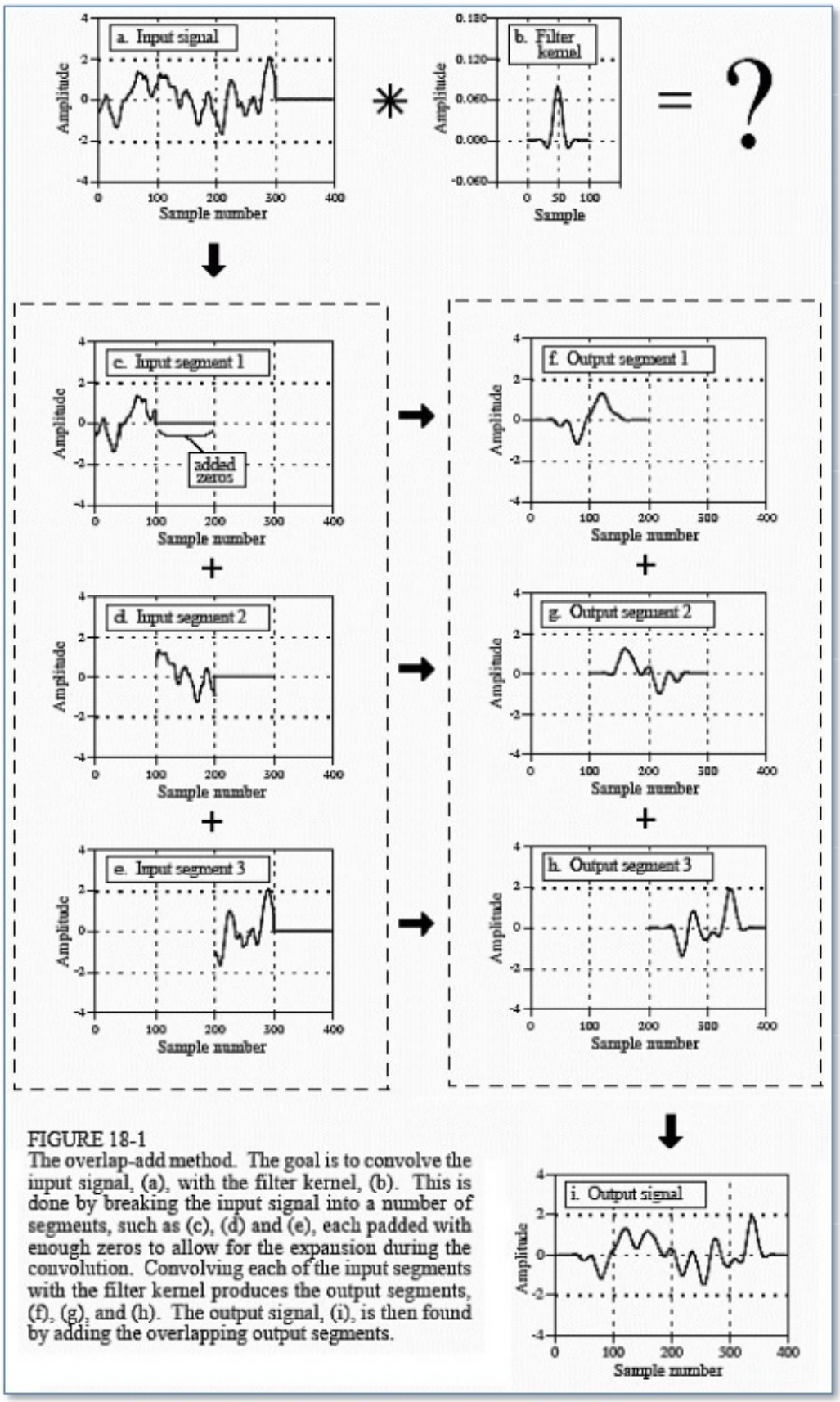
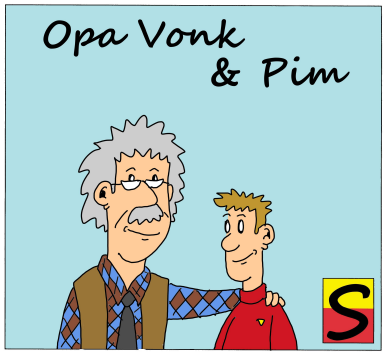
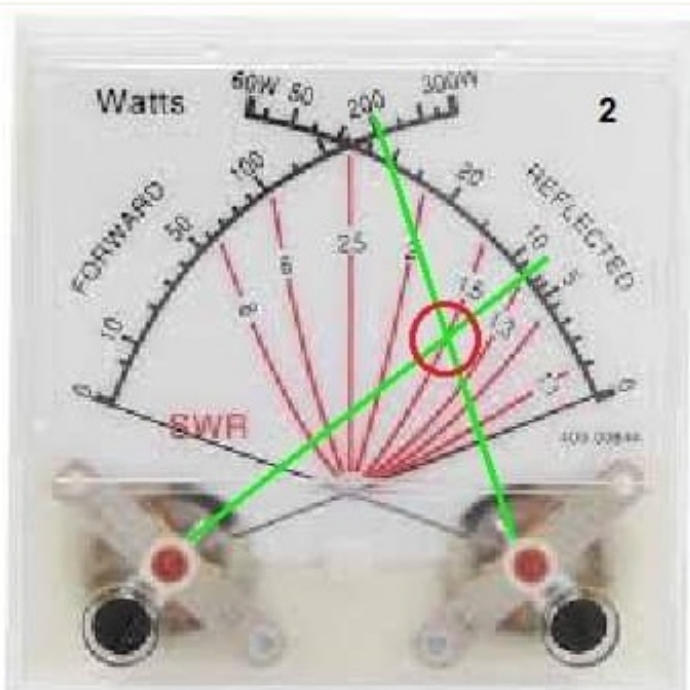


FIGURE 18-1
The overlap-add method. The goal is to convolve the input signal, (a), with the filter kernel, (b). This is done by breaking the input signal into a number of segments, such as (c), (d) and (e), each padded with enough zeros to allow for the expansion during the convolution. Convolving each of the input segments with the filter kernel produces the output segments, (f), (g), and (h). The output signal, (i), is then found by adding the overlapping output segments.

Source: https://www.eetimes.com/fft-convolution-and-the-overlap-add-method/



Opa Vonk hoorde zijn kleinzoon Pim mopperen, maar kon uit het gemompel niet destilleren waar Pim's probleem zat. "Lukt het niet?" informeerde Opa. "Nou, ja en nee", zei Pim. "Ik ben de antenne aan het afstemmen, maar ik zie niet wat die ouwe SWR meter van u nou precies bedoelt. Waar is die elektronische SWR meter die u had en die gewoon aangaf wat de SWR is?" vroeg hij. Opa keek een beetje beschaamd. "Ik heb 'm opgeblazen. Hij ligt nog op de werkbank om gerepareerd te worden. Maar wat is er mis met deze?" vroeg Opa. "Nou, ik probeer de SWR van de antenne te meten, maar deze meter heeft twee wijzers in plaats van één. Mijn meter heeft zo'n FWD/REV schakelaartje waarbij je eerst op FWD de meter op 100% volle schaal regelt en je daarna op REV gewoon de SWR af kan lezen. Maar bij deze ontbreekt de schakelaar en de regelaar voor 100%. Die wijzers zwabberen wel als ik vermogen toevoer, maar als ik het vermogen vergroot, gaan ze allebei omhoog. Hoe lees je nou de SWR af?" vroeg Pim. "Ja, die 27Mc meters werkten bijna allemaal op die manier", zei Opa terwijl Pim een platgeslagen mug op het behang bestudeerde, om een preek over de tegenwoordige 27Mc band te voorkomen. "Maar zo ingewikkeld is het niet. Feitelijk zijn het twee meters in 1 behuizing. De rechter meter geeft het heengaande vermogen weer, wat bij jouw meter FWD was, en de linker meter geeft het gereflecteerde vermogen weer, wat bij jouw meter REV was. Maar je moet niet naar het gereflecteerde vermogen kijken, hoewel dat natuurlijk wel een indicatie is voor de SWR, maar naar het punt waar de twee wijzers elkaar kruisen. Ik geef je een voorbeeld. Hieronder heb ik twee keer een meting ingetekend in de meter. Links zie je dat er 100W richting de antenne gaat, en dat er 4W terug komt. Merk op dat de schaal van de twee meters anders is. De rechter meter gaat tot 300W, waar de linker meter maar tot 60W gaat. Het idee erachter is dat er over het algemeen toch meer vermogen richting de antenne gaat dan dat er terugkomt. Dat lijkt me in het kader van de efficiëncy ook wel zo wenselijk. Kijk nu langs het punt waar de twee wijzers elkaar kruisen naar het rode lijntje en volg dat tot je het bijbehorende getal ziet. Wat staat daar?" vroeg Opa. "1,5", antwoordde Pim. "Inderdaad. En dat is de SWR. Als ik het vermogen nu verhoog naar 200W, dan zie je dat er nu 8W terug komt. Lees de meter nog eens af?" vroeg Opa. "Weer 1,5", zei Pim verbaasd. "Waarom ben je verbaasd?"



vroeg Opa. "De verhouding tussen heengaand en teruggaand vermogen blijft immers hetzelfde zolang de SWR niet verandert: de afkorting staat immers niet voor niet voor "Standing Wave Ratio" waarbij Ratio Engels is voor verhouding. Dus ja, bij het verhogen van het vermogen lopen beide wijzers op, maar langs hetzelfde lijntje dat een SWR van 1,5 aangeeft. Je zult je nog wel herinneren van je eigen oude meter met schakelaar dat je af en toe weer terug moest schakelen naar FWD om de volle schaal weer opnieuw af te regelen, want vooral als een set terugregelde was het heengaande vermogen lang niet altijd constant. Dat is bij deze meter niet nodig. Je kunt zonder schakelen aflezen hoeveel vermogen er naar de antenne gaat, hoeveel er terugkomt én wat de SWR is. De meeste kruisnaaltmeters hebben ook nog wel een knopje om de gevoeligheid een factor 10 te verhogen zodat de volle schaal 30W wordt. Want tunen bij 100W is om verschillende redenen niet gewenst en zo kan je met wat minder vermogen toch de kruising van de twee wijzers nog goed aflezen. Zorg dat je altijd de documentatie leest die met de meter geleverd wordt - dat geldt overigens voor elk apparaat - en gebruik niet de amateurmethode 'als niets meer helpt, lees dan de gebruiksaanwijzing'. Kies ook een meter die voor je station geschikt is. Dus een meter met hooguit 10W volle schaal als je veel QRP doet, of met 2kW volle schaal als je denkt dat je niet zonder lineair kan. Verwissel ook niet de antenne- en transceiver-aansluiting, want dan slaat de aanwijzing nergens meer op en je loopt het risico je Reverse meter op te blazen omdat die een factor 5 gevoelinger is dan je Forward meter. Controleer de werking van je SWR meter door een goede dummy aan te sluiten. De meter moet dan een SWR dicht bij de 1:1 aangeven; bij de kruisnaaldmeter zal de Reverse wijzer niet uit de hoek komen. En dan heb je een meter die prima af te lezen is. Overigens hebben ook veel kruisnaaldmeters nog een knopje om de meter nog enige tijd te laten 'hangen'. Tijdens het tunen is dat vervelend omdat de meter dan traag reageert op het tunen terwijl je dan juist een snelle reactie wil, Maar heel veel amateurs worden uitermate zenuwachtig als de meter tijdens het moduleren in SSB niet de 100W aantikt. Door het knopje wordt de meter een piek-detector waardoor hij een tijdje op het maximum vermogen blijft hangen. En dat stelt veel amateurs gerust", besloot Opa. Pim knikte begrijpend. "Nu heb ik het door. Ik snapte die vertikale lijntjes niet. Maar die moet je dus door de kruising van de wijzers aflezen en dan geven die de SWR aan. Misschien is deze dan nog wel handiger dan die elektronische meter die u eerst had, Want ik kan zowel heengaand als gereflecteerd vermogen én de SWR in één keer aflezen zonder dat ik op knopjes hoef te drukken voor het omschakelen van de uitlezing. Eigenlijk ben ik er dus nu wel blij mee", zei Pim. "Mooi", zei Opa. "Dan hoef ik ook geen haast te maken met de reparatie van de elektronische meter", zei hij, alvorens de soldeerbouw weer op te pakken voor een ander apparaat dat zijn aandacht nodig had.

MQTT

Robert de Kok, PA2RDK

Inleiding

Bij de RAZ zijn we verslingerd geraakt aan de Arduino, ESP32 en andere soortgelijke microprocessors. De mooiste apparaten worden er mee gemaakt: radio’s, CW-keyers en -decoders, weerstations, APRS-toepassingen maar ook complete transceivers.

Vaak worden de processors in een standalone toepassing ingezet en is (draadloze) communicatie met andere apparatuur geen vereiste.

Persoonlijk heb ik daarbij ook een fascinatie voor alles wat met Domotica te maken heeft en daarbij is bijvoorbeeld de ESP32 goed bruikbaar

voor het verrichten van allerlei taken die met meten en schakelen te maken hebben.

Juist de ESP32 is daarbij een geschikte processor omdat die standaard is voorzien van een heleboel vrije pootjes die zowel digitaal als analoog zijn in te zetten, maar vooral omdat die is voorzien van een Wifi en Bluetooth interface, hetgeen communicatie met deze processor mogelijk moet maken.

Middels Wifi is het bijvoorbeeld mogelijk om de actuele datum en tijd op te halen en om (via een API) informatie van een website af te halen. Hiervan maken we gebruik in het weerstation dat recent in de RAZZies is gepubliceerd, waarbij we de weersinformatie van verschillende websites ophalen. Ook de APRS-gateway maakt gebruik van de Wifi-interface om de APRS-gegevens op aprs.fi te zetten.

Genoemde voorbeelden gaan uit van communicatie met een gedefinieerd koppelvlak, maar het komt ook (veel vaker) voor dat we een bericht naar een apparaat willen sturen zonder dat er hiervoor een standaard interface beschikbaar is. Bijvoorbeeld het versturen van meetresultaten naar een Domotica server in de vorm van een PC of Raspberry.

Om maar eens een heel praktisch voorbeeld te nemen, we willen via Wifi een draadloze remote deurbel maken, hiervoor willen we aan de fysieke deurbel een ESP32 hangen die een seintje geeft aan een tweede ESP32 waar we een bel, speaker of buzzer aan bevestigen. We kunnen hierbij een ESP32 als server inzetten en de tweede als client daaraan verbinden. Logischer is het om beide ESP32's als client te verbinden met de aanwezige Wifi router. Dit komt ongetwijfeld de stabiliteit en het bereik ten goede en op deze manier kunnen we prima een fysieke verbinding opzetten. Maar we zijn er niet met alleen een fysieke verbinding, we moeten ook een seintje geven van de een naar de andere ESP. Hierbij zijn er vele wegen die naar het spreekwoordelijke Rome leiden, we kunnen bijvoorbeeld communiceren via TCP of UTP of websockets of zelfs FTP of het heel luxe maken en een REST-interface (Representational State Transfer interface -red) bouwen. Al deze oplossingen hebben een aantal nadelen zoals de noodzaak om het IP-adres van de ontvanger te kennen en te kunnen configureren, maar waarbij vooral de foutafhandeling een uitdaging gaat worden.

Over juist dit vraagstuk, weliswaar niet per se met een deurbel, zat ik na te denken en een beetje te browsen op het almachtige internet toen ik stuitte op MQTT, waarbij de M staat of stond voor Message. Het zei mij niets, maar ik vond het wel lekker klinken: het project waarmee ik bezig was ging even opzij om ruimte te maken voor een beetje cursus…

MQTT stond voor Message Queuing Telemetry Transport, of in goed nederlands het transporteren van berichten tussen machines middels gebruik van een wachtrij. Het bericht wordt dus verstuurd naar een wachtrij (Queue) en doorgestuurd naar of opgehaald door de machine(s) waarvoor het bericht bedoeld is. Precies de oplossing voor het hierboven omschreven probleem.

Onderhand is het protocol zover uitgebreid dat de naam de lading niet meer dekt. Daarom is ervoor gekozen om MQ niet meer als een afkorting te zien maar als een naam. De volledige naam is nu dus MQ Telemetry Transport.

Wat is MQTT

MQTT bestaat uit clients en een broker:

De client: Er zijn 2 soorten clients: de publisher, deze publiceert of verstuurt berichten en de subscriber of abonnee, deze ontvangt berichten. Natuurlijk kan een machine zowel publisher als subscriber zijn om zodoende tweeweg communicatie op te zetten.

De broker: (of in het nederlands een makelaar) ontvangt berichten van de publisher, slaat deze op in een wachtrij en biedt deze aan bij de

subscribers.

Het MQTT-verkeer loopt gewoon via internet, standaard via poort 1883 of beveiligd met SSL via poort 8883, dus de infrastructuur is meestal wel voorhanden.

In ons voorbeeld zijn de 2 ESP's dus clients maar hebben we ook nog behoefte aan een broker. Een broker mag overal staan, als die maar via internet bereikbaar is. Op het internet zijn veel (gratis) brokers beschikbaar, hier vind je er een aantal: https://mntolia.com/10-free-public-private-mqtt-brokers-for-testing-prototyping/.

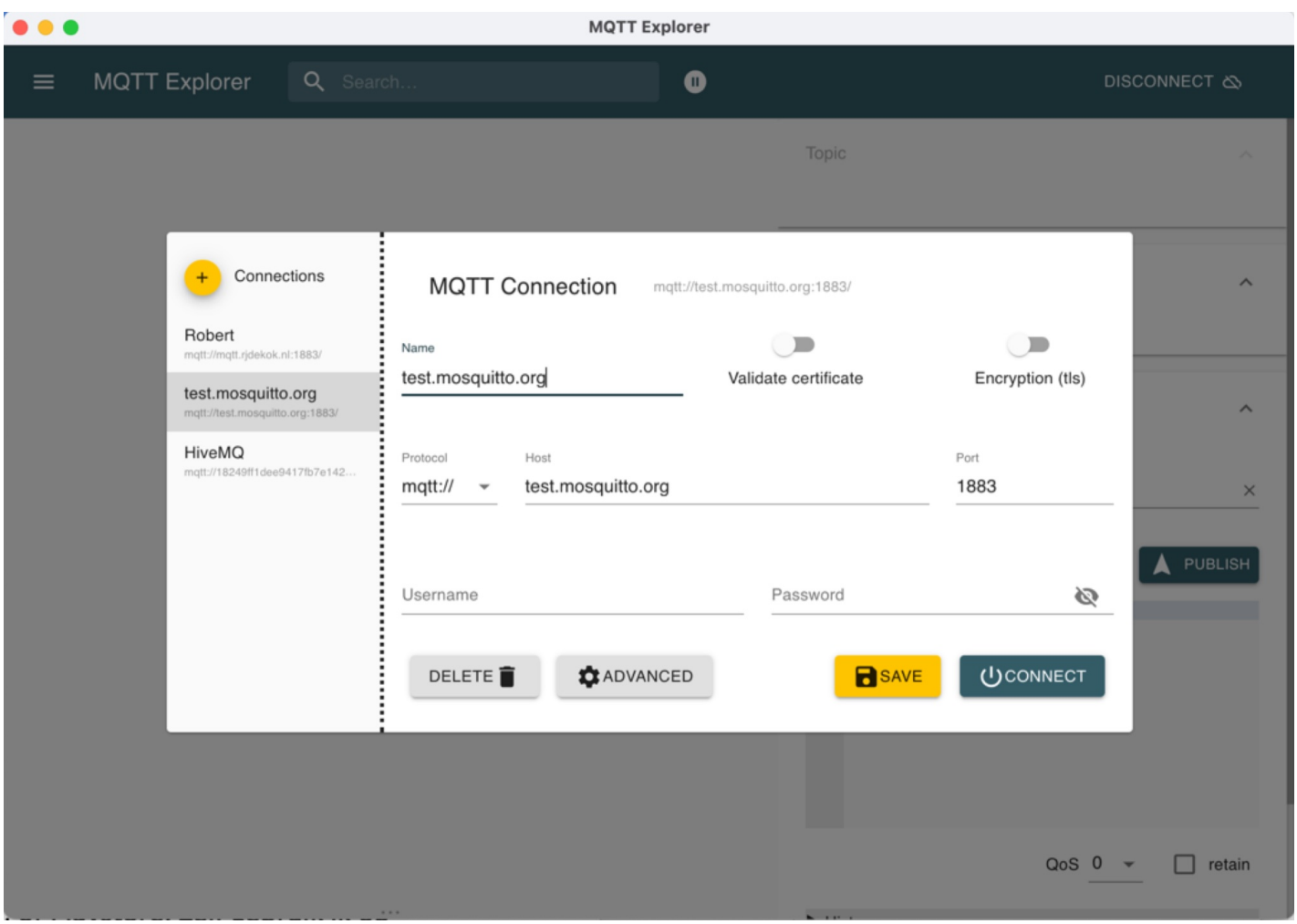
Brokers zijn er in vele smaken, met en zonder authenticatie, met en zonder encryptie (SSL), via TCP maar ook via web sockets.

Realiseer je wel dat MQTT-verkeer standaard niet beveiligd is, hieraan helpt ook authenticatie niet, dus er kan gewoon worden meegelezen. Authenticatie zorgt er alleen voor dat je dient in te loggen om gebruik te kunnen maken van de broker. Bij gebruik van SSL is dat natuurlijk beter geregeld, maar het implementeren van SSL op een ESP32 is wel een dingetje. Je kunt natuurlijk ook jouw eigen broker hosten, daar kom ik verderop nog op terug.

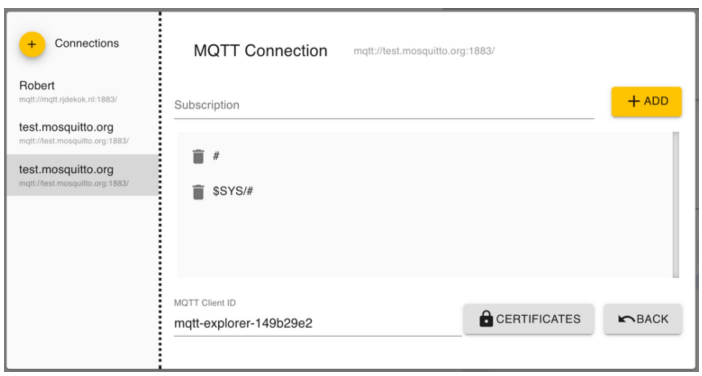
Om te kunnen meelezen met MQTT, bestaan er voor Windows, OSX en Linux vele clients. Hier vind je de MQTT Explorer voor verschillende operating system. Maar er zijn er meer, ook voor IOS en Android, zoek maar eens in de Appstore, of Playstore.

Zelf gebruik ik op mijn Mac MQTT Explorer. Deze ziet er uit zoals te zien is op onderstaand plaatje.

Op mijn iPhone gebruik ik MQTTool.



We maken om lekker laagdrempeling te beginnen gebruik van de Mosquitto testserver. Deze is te vinden op https://test.mosquitto.org. Dit is een echte testserver, waarbij gebruik gemaakt kan worden van verschillende poorten om de verschillende mogelijkheden te testen. In ons voorbeeld maken we gebruik van poort 1883, deze is publiek te gebruiken zonder authenticatie. Er hoeft dus geen Username/ Password te worden ingegeven.



Een MQTT bericht bestaat uit een topic (onderwerp) en het bericht zelf. Je dient de client te abonneren op een of meer topics, anders krijg je geen berichten binnen. Het abonneren kan onder de knop ‘ADVANCED’. Topics bieden verschillende mogelijkheden, de meest eenvoudige is een enkel woord, bijvoorbeeld ‘PA2RDK’. Hierbij krijgen we alle berichten met dit topic binnen. Maar we kunnen ze ook nesten en gebruik maken van subtopics, bijvoorbeeld ‘PA2RDK/Deurbel’ en PA2RDK/Weerstation’.

Hierbij kunnen we abonneren op exact deze subtopics, maar ook een wildcard, dit is in dit voorbeeld ‘PA2RDK/#’ of ‘PA2RDK/+ waarbij # en + dus de wildcards zijn. In dit geval krijgen we de berichten binnen voor ‘PA2RDK/Deurbel’ en PA2RDK/Weerstation’. Ook is het mogelijk te abonneren op een subtopic ongeacht het topic. Dit is bijvoorbeeld ‘+/Deurbel’. In dit geval komen de berichten van ‘PA2RDK/Deurbel’, maar ook van ‘TEST/Deurbel’ binnen. Nesten kan tot een groot aantal levels, de maximale lengte van het totale topic is 65535 bytes.

Er is een verschil tussen # en + als wildcard; Bij gebruik van de + kan er niet dieper worden genest, ‘PA2RDK/Deurbel’ komt binnen, maar ‘PA2RDK/Deurbel/Voordeur’ komt niet binnen.

Bij gebruik van het # mogen er op de plaats van het hekje meerdere subtopics genest worden, hierbij komen ‘PA2RDK/Deurbel’ en ‘PA2RDK/Deurbel/Voordeur’ dus beide binnen. Het # mag alleen aan het eind van de reeks worden toegepast, de + kan overal in de reeks staan. Dit is dus ook toegestaan: ‘PA2RDK/+/Voordeur’.

Maak GEEN gebruik van het $ teken in topics, deze is gereserveerd voor intern gebruik. Wel kan het interessant zijn te abonneren op ‘$SYS/#’. De broker publiceert naar deze topics de interne statistieken.

Het bericht dat wordt verstuurd kan in principe alles zijn met maximaal een lengte van 256Mb. Dit kan gewone tekst, XML of JSON zijn, maar ook een plaatje kan worden verstuurd, converteer die dan wel naar base64, om te zorgen dat die niet verminkt overkomt.

Het versturen van een bericht

Ik ga er in onderstaande tekst vanuit dat de basiskennis over het programmeren met Arduino van een ESP32 bekend is. De getoonde code is dan ook geen volwaardig en bruikbaar project maar zijn slechts voorbeelden om de functionaliteit te tonen. We maken gebruik van de standaard MQTT library. Deze is hier te vinden, maar ook is deze in de Arduino omgeving standaard beschikbaar:

De Arduino code om een bericht te publishen ziet er als volgt uit:

```
#include <MQTT.h>   //Load the MQTT library
#include "WiFi.h"

WiFiClient net;
MQTTClient client;  //Create an instance of the MQTT client and call it 'client'
int lastMillis = 0;
int counter = 0;

struct StoreStruct {
   char SSID[25];
   char pass[25];
   char mqtt_broker[50];
   char mqtt_user[25];
   char mqtt_pass[25];
   int mqtt_port;
};

StoreStruct storage = {
   "SSID",
   "WiFiPassword ",
   "test.mosquitto.org",
   "UnUsed",
   "UnUsed",
   1883
};

void setup() {
   Serial.begin(9600);
   Serial.println("PA2RDK MQTTTester");
   client.begin(storage.mqtt_broker, storage.mqtt_port, net);  //Start the MQTT client
}

void loop(){
   check_connection(); //Connect and check connection to the WiFi en MQTT broker
```

```
  delay(10);
  if (millis() - lastMillis > 10000) {
    counter++;
    lastMillis = millis();
    Serial.println("Send MQTT Message:" + String(counter));
    client.publish("PA2RDK/Deurbel", "Testbericht:" + String(counter)); //Send a message every 10 seconds
  }
}

boolean check_connection() {
  if (WiFi.status() != WL_CONNECTED) InitWiFiConnection();

  if (WiFi.status() == WL_CONNECTED){
    if (!client.connected()) InitMQTTConnection();
  }

  return (WiFi.status() == WL_CONNECTED) &&client.connected();
}

void InitWiFiConnection() {
  Serial.println("Connecting to WiFi");
  WiFi.begin(storage.SSID,storage.pass);

  while (((WiFi.status()) != WL_CONNECTED)){
    Serial.print(".");
    delay(1000);
  }

  if (WiFi.status()==WL_CONNECTED){
    Serial.print("WiFi Connected to: ");
    Serial.println(storage.SSID);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }
}

void InitMQTTConnection() {
  Serial.println(F("Connecting to MQTT."));

  while (!client.connect("MQTTGateway", storage.mqtt_user, storage.mqtt_pass)) {  //Connect to the MQTT
broker
    Serial.print(".");
    delay(1000);
  }
  Serial.println(".");
```

```
  if (client.connected()){
     Serial.print(F("MQTT connected to: "));
     Serial.println(storage.mqtt_broker);
        client.publish("PA2RDK/Status","Online",1,0);      //If connected, send the message 'Online' to topic 'PA2RDK/Start
  }
}
```

Er zijn maar een paar regels in gebruik om een bericht te kunnen versturen. Deze zijn in de code voorzien van commentaar. Deze spreekt voor zich, behalve wellicht het publish commando: In de functie 'InitMQTTConnection' die wordt aangeroepen vanuit 'check_connection()' wordt met de regel client.publish("PA2RDK/Start","Online",1,0); een bericht verstuurd met als topic 'PA2RDK/Start' en met de inhoud 'Online'. Dit bericht wordt dus verstuurd als er een verbinding met de broker wordt gemaakt. In de loop controleer ik continu of de verbinding WiFi en MQTT-verbindingen nog bestaan en zet deze zonodig opnieuw op. In de loop wordt elke 10 seconden een bericht verstuurd met de regel client.publish("PA2RDK/Deurbel", "Testbericht:" + String(counter)); naar de broker met als topic 'PA2RDK/Deurbel' en met de inhoud 'Testbericht:<counter>'. Bij het eerste bericht zijn 2 extra parameters in gebruik: retained en QOS.

Een bericht waarvan de vlag 'retained' (behouden) aan staat (=1) wordt ALTIJD gestuurd naar een client die zich opnieuw aanmeldt. Dit heeft een groot voordeel, de client weet op deze manier dat die naar het juiste topic luistert, tenminste als het bericht voldoende inhoudelijk is. Als er geen retained bericht klaar staat, kan het wel minuten of uren duren voor er een bericht door de publisher wordt verstuurd en door de subscriber wordt ontvangen. Maar vergis je niet! De ontvangst van dit bericht door de client wil niet zeggen dat de publisher online is en werkt. De service wordt verleend door de broker, dus het bericht wordt ook verstuurd als de publisher off-line is. Als je alle berichten 'retained' verstuurt, weet je wel zeker dat de subscriber altijd het laatste bericht ontvangt. Een topic kan maar 1 retained bericht bevatten, het laatste bericht overschrijft het voorgaande bericht. Als je het retained bericht wilt wissen, stuur dan een leeg bericht met de retained vlag aan.

De QOS staat voor Quality Of Service, de zekerheid waarmee een bericht aankomt. De standaard mode is 0, het bericht wordt 1 keer aangeboden en ontvangst wordt niet gecontroleerd. Ingeval de QOS=1 stuurt de subscriber een bevestiging van ontvangst, het bericht wordt net zo lang gestuurd totdat de bevestiging is ontvangen. Hierbij kan een bericht dus meermaals aankomen, als de subscriber het bericht ontvangt maar de publisher de bevestiging niet ontvangt dan stuurt de publisher het bericht nogmaals. Hierbij bestaat het risico dat een ouder bericht later aankomt bij de subscriber dan een nieuwer bericht.

Als de QOS=2 dan wordt het bericht zeker 1 keer aangeboden. Hierbij wordt gebruik gemaakt van een 4 staps handshake. Dit is de meest betrouwbare methode, maar geeft wel de meeste belasting van de broker, clients en het netwerk.

Het ontvangen van een bericht

De code voor een subscriber wijkt weinig af van die van de publisher:

```
#include <MQTT.h>   //Load the MQTT library
#include "WiFi.h"

WiFiClient net;
MQTTClient client;  //Create an instance of the MQTT client and call it 'client'
int lastMillis = 0;
int counter = 0;

struct StoreStruct {
   char SSID[25];
   char pass[25];
   char mqtt_broker[50];
   char mqtt_user[25];
   char mqtt_pass[25];
   int mqtt_port;
};

StoreStruct storage = {
   "SSID",
   "WiFiPassword!",
   "test.mosquitto.org",
   "UnUsed",
   "UnUsed",
   1883
};

void messageReceived(String &topic, String &payload) { //De CallBack functie die wordt aangeroepen
   Serial.println("incoming: " + topic + " - " + payload);
   Serial.println(payload);
}

void setup() {
   Serial.begin(9600);
   Serial.println("PA2RDK MQTTTester");
   client.begin(storage.mqtt_broker, storage.mqtt_port, net);
   client.onMessage(messageReceived);       //Initialisatie van de CallBack functie vanuit de MQTT library
}

void loop(){
   check_connection(); //Connect and check connection to the WiFi en MQTT broker
   client.loop();
   delay(10);
}

boolean check_connection() {
   if (WiFi.status() != WL_CONNECTED) InitWiFiConnection();

   if (WiFi.status() == WL_CONNECTED){
```

```
    if (!client.connected()) InitMQTTConnection();
  }

  return (WiFi.status() == WL_CONNECTED) &&client.connected();
}

void InitWiFiConnection() {
  Serial.println("Connecting to WiFi");

  WiFi.begin(storage.SSID,storage.pass);

  while (((WiFi.status()) != WL_CONNECTED)){
    Serial.print(".");
    delay(1000);
  }

  if (WiFi.status()==WL_CONNECTED){
    Serial.print("WiFi Connected to: ");
    Serial.println(storage.SSID);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }
}

void InitMQTTConnection() {
  Serial.println(F("Connecting to MQTT."));

  while (!client.connect("MQTTGateway", storage.mqtt_user, storage.mqtt_pass)) {
    Serial.print(".");
    delay(1000);
  }
  Serial.println(".");

  if (client.connected()){
    Serial.print(F("MQTT connected to: "));
    Serial.println(storage.mqtt_broker);
    client.subscribe("PA2RDK/#");  //Subscribe to the desired topic.
  }
}
```

De eerste afwijkende functie is void messageReceived(String &topic, String &payload). Deze functie wordt aangeroepen als er een bericht wordt ontvangen. De aanroep van deze functie gebeurt middels een 'callback' vanuit de MQTT library. Om de MQTT library uit te leggen welke functie moet worden aangeroepen, staat in de setup in de regel client.onMessage(messageReceived); De tekst spreekt voor zich, als er een bericht wordt ontvangen, wordt deze functie aangeroepen met de parameters topic en payload (bericht). Aan jou om de bel of buzzer in deze functie zijn werk te laten doen. De derde aanpassing vinden we in de functie InitMQTTConnection. De regel client.subscribe("PA2RDK/#"); is verantwoor-

delijk voor het subscriben aan het gewenste topic. Kom nu niet in de verleiding te subscriben aan #, dan krijgt de ESP het heel druk en geeft het vast op. In de 'loop' gebeurt eigenlijk niets meer, behalve dat we de status van de Wifi en MQTT-connectie controleren en de loop functie van de library aanroepen.

Je kunt met de functie client.unsubscribe("PA2RDK/#"); ook unsubscriben van het topic.

Een volgende heel bruikbare functie is 'setWill'. Hiermee maakt de publisher een testament. Dat wil zeggen dat de broker dit bericht verstuurt als het constateert dat de publisher off-line is. Hierbij kennen we ook de parameters voor retained en QOS. Deze functie, samen met een retained bericht maken het mogelijk om continu te monitoren of de publisher nog on-line is.

De setWill moet gezet worden voor de connect, dus als volgt:

```
void InitMQTTConnection() {
  Serial.println(F("Connecting to MQTT."));

  client.setWill("PA2RDK","OffLine",1,0);
  while (!client.connect("MQTTGateway", storage.mqtt_user, storage.mqtt_pass)) {
    Serial.print(".");
    delay(1000);
  }
  Serial.println(".");

  if (client.connected()){
    Serial.print(F("MQTT connected to: "));
    Serial.println(storage.mqtt_broker);
    client.publish("PA2RDK","Online",1,0);
    client.subscribe("PA2RDK/#");
  }
}
```

Op deze wijze wordt door de regel client.publish("PA2RDK","Online",1,0); altijd een bericht gestuurd naar de subscriber dat de publisher online is, zodra de subscriber zich aanmeldt. Mocht de publisher off-line gaan, dan is deze regel client.setWill("PA2RDK","OffLine",1,0); er verantwoordelijk voor dat alle subscribers hiervan een bericht ontvangen, ook als ze zich later aanmelden.

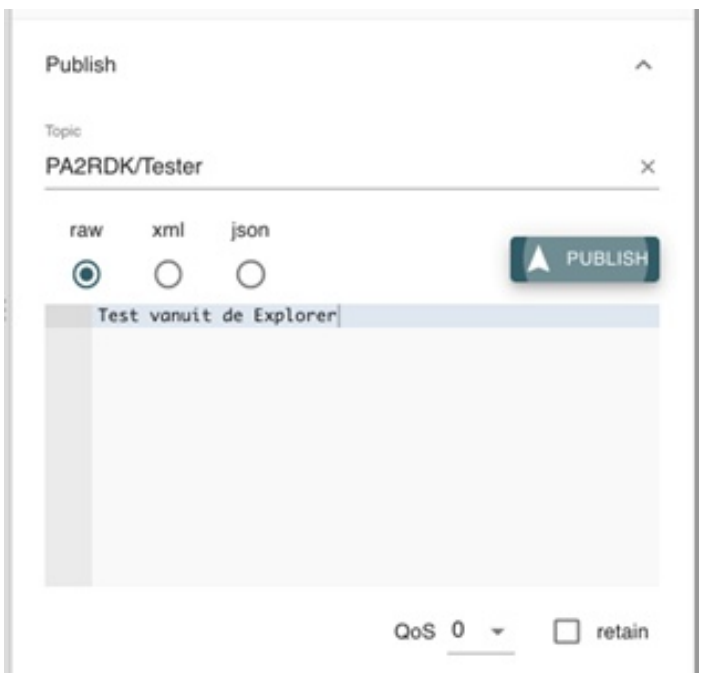
Met de functie void clearWill(); kan een testament eventueel ongedaan gemaakt worden.

Een laatste functie van de library die ik wil benoemen is 'setOptions', met deze functie kunnen 3 opties worden gezet:

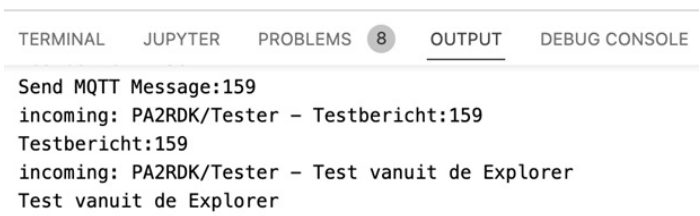
```
int keepAlive=10;   //keepAlive interval in seconden, dus na 10 seconden heeft de broker in de gaten dat de publisher off-line is.

bool cleanSession = true;   //zie uitleg

int timeOut = 1000;  //timeOut in miliseconden, we verwachten dus binnen een seconde reactie.

client.setOptions(keepAlive, cleanSession, timeout);
```

cleanSession: Standaard start een verbinding van een subscriber ‘clean’. Het ontvangt het laatste retained bericht en alle nieuwe berichten. We kunnen deze optie ook op false zetten, dan gaat de broker er van uit dat de subscriber er altijd moet zijn. Het effect hiervan kan heel bruikbaar zijn; als een subscriber berichten heeft gemist en dit bericht heeft een QOS van 1 of 2, wordt deze alsnog aangeboden. Een subscriber zal dus nooit een bericht missen!

Je wil de subscriber natuurlijk ook testen en dit kan ook met de MQTT Explorer. Rechts zit een knop ‘PUBLISH’. Zet het topic waarin je wilt publishen, type een bericht, kies de gewenste QOS en zet, als je dat wilt, retain aan en druk op PUBLISH.



Als alles goed gaat, komt het bericht binnen in het terminalscherm van Arduino:

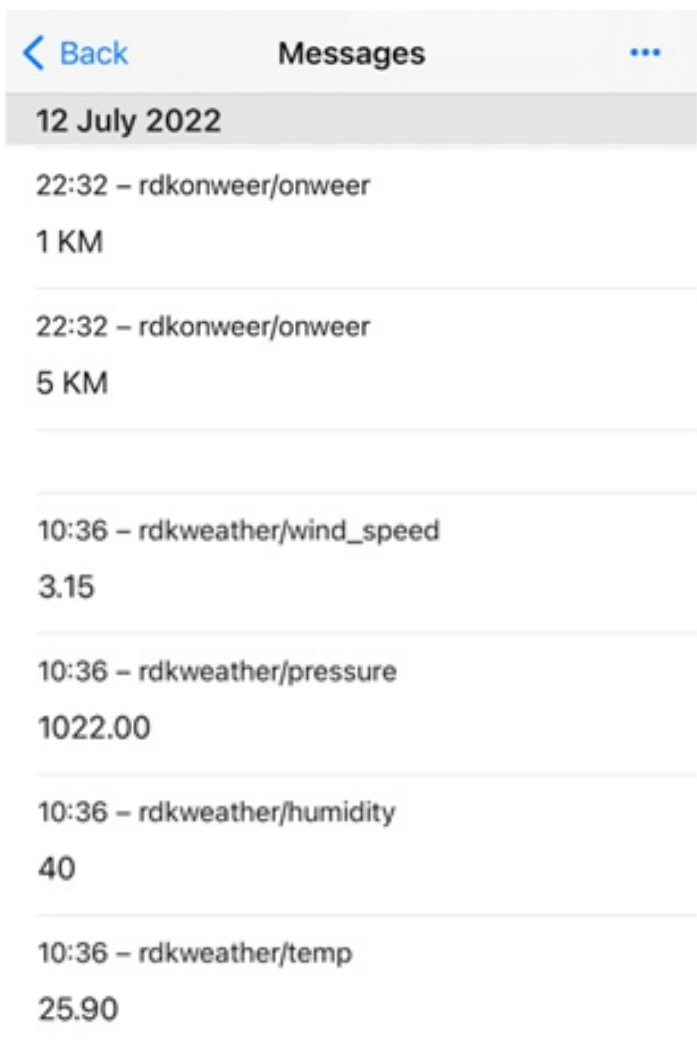


Wat kan je met MQTT?

De meeste Domotica systemen kunnen met MQTT overweg, door MQTT te implementeren in het eerder genoemde weerstation en ook in de onweersdetector heb ik de gegevens in mijn domotica systeem beschikbaar.

Voor IOS en Android is er de app ‘MQTTPushClient’ beschikbaar. Hiermee kun je subscriben aan een broker en topic en pushberichten naar jouw telefoon laten sturen. Ik krijg dus een pushbericht als er onweer op komst is. Maar ook de meetgegevens van het weerstation zijn beschikbaar.

In en om het huis heb ik een aantal bewegings-sensoren, deze zijn via LoRA verbonden met een (gebruik makend van een Heltec ESP32LoRa) LoRa2MQTT interface die de LoRa bewegingsmeldingen ontvangt en via MQTT deze doorgeeft aan mijn Domotica en telefoon. De meest zinloze implementatie is die in de webradio, via MQTT geeft die continu door welk nummer er nu wordt gespeeld ;-)
Maar, je kunt vast tientallen heel nuttige toepassingen bedenken.



Een eigen MQTT broker

In bovenstaand verhaal hebben we gebruik gemaakt van een publiek beschikbare broker. Maar het is natuurlijk ook leuk om zelf een broker te bezitten. Dit stelt wel eisen aan de aanwezige infrastructuur:

- jouw IP adres moet (min of meer) vast zijn, mijn ervaringen met Ziggo zijn goed, alleen in heel extreme gevallen wordt het IP adres gewijzigd.
- Bij voorkeur heb je een DNS beschikbaar, zodat je mqtt.jouwnaam.nl kunt routeren naar jouw IP adres. Als dan jouw IP adres wijzigt, hoef je alleen de DNS aan te passen en niet alle subscribers en publishers.
- De poort op jouw router (1883 en 8883) moet open staan en bijvoorbeeld geNAT worden naar de server waarop de broker draait.
- Er moet een machine beschikbaar zijn die altijd aan staat.

Voor vrijwel elk platform is wel een broker beschikbaar, Linus, OSX, Windows, het mag allemaal. Ik heb het mijzelf eenvoudig gemaakt door gebruik te maken van een beschikbare Docker container en deze geïnstalleerd op mijn NAS.

Vergeet niet om tenminste authenticatie aan te zetten op de broker. De beveiliging is beperkt, maar de kanst dat jouw broker wordt misbruikt is een stuk kleiner.

Bronvermelding en verder leesvoer:

https://www.hivemq.com/mqtt-essentials/?utm_source=bing&utm_medium=ppc&utm_campaign=EU%20%7C%20MQTT%20Main&utm_content=MQTT%20Essentials&utm_term=mqtt&msclkid=1f7b8dcca96215e587b0ef1f55e111f6

https://test.mosquitto.org

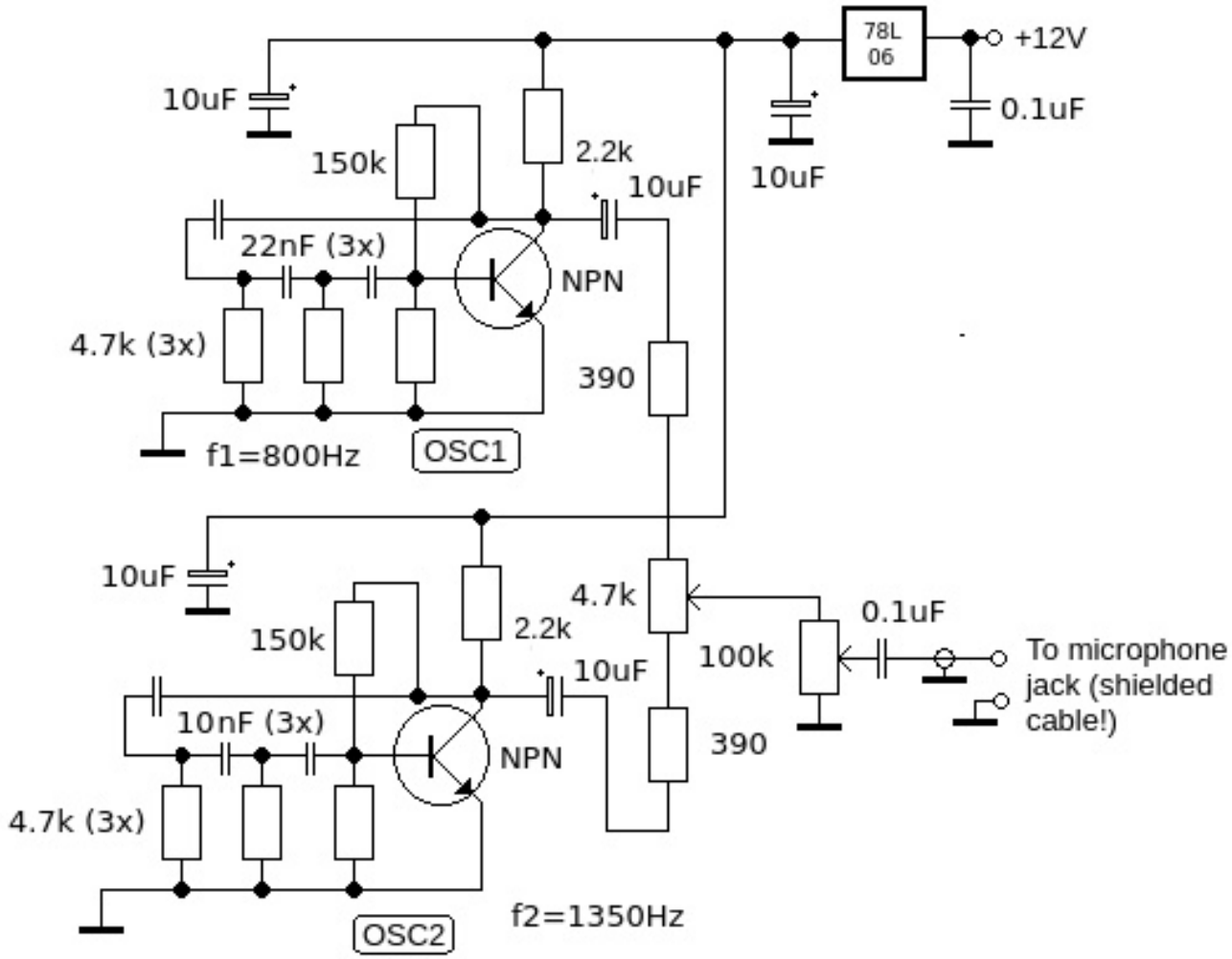
https://mosquitto.org/man/mqtt-7.html

Dubbeltoongenerator voor het testen van SSB zenders

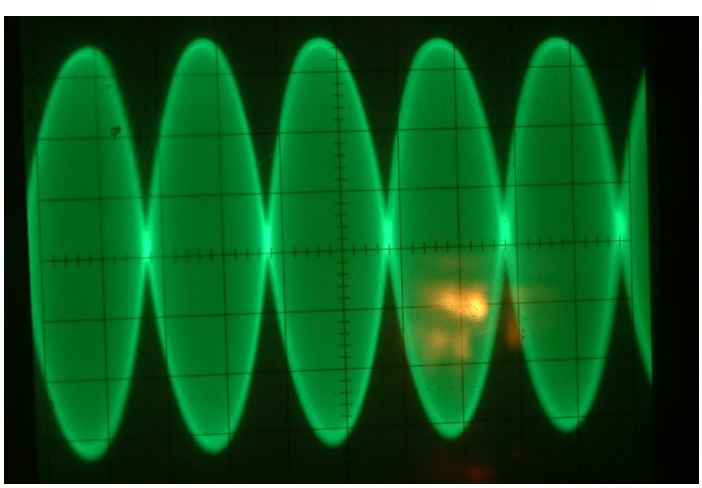
Dit apparaat is even eenvoudig als handig. Met 2 transistoren en een oscilloscoop is het vrij eenvoudig om de lineariteit van je SSB zender te testen. De 2-tone-methode is een gestandaardiseerde testmethode waarmee het maximale vermogen en de versterkingskarakteristieken van een zijbandzender kunnen worden bepaald.

De methode omvat 2 audiotonen die tegelijkertijd worden toegevoerd aan de microfooningang van het DUT (apparaat dat wordt getest). De twee frequenties mogen niet harmonisch gerelateerd zijn en moeten beide binnen de audiodoorlaatband van de zender vallen. Ze moeten ook een aanzienlijke afstand hebben in het audiospectrum. 850 Hz en 2200 Hz zijn bijvoorbeeld een goede keuze. Indien je in het bezit bent van een spectrumanalyzer met een passende resolutie kunnen ook de Third Order Intermodulation producten (IM3) zichtbaar worden gemaakt. Ik zal daar op een latere pagina naar verwijzen, vandaag willen we het alleen hebben over de amplitude-kant. Het schema vind je bovenaan de volgende bladzijde

Het hart van het apparaat zijn twee oscillatoren die 2 sinusgolven produceren die een nieuwe golfvorm genereren die wordt gevormd door superpositie van de 2 enkelvoudige signalen. Het lijkt op een amplitude-gemoduleerde draaggolf en zou er als volgt uit moeten zien wanneer het wordt weergegeven met een oscilloscoop, zie ook weer volgende bladzijde.



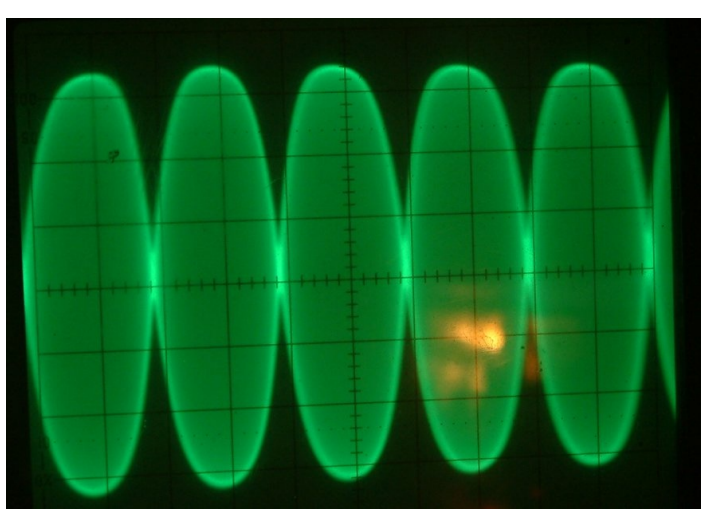
TwoTone audio generator for linearity test in SSB transmitters



SSB 2-toon test signaal: de zender is correct afgeregeld

De "balans" van het toongeneratorcircuit moet zorgvuldig worden afgesteld om de amplitudes van de twee audiosignalen zo te regelen dat de kruising die de golfvorm centreert zo scherp mogelijk is. Dan zijn de twee audiosignalen bijna 100% gelijk, wat zal resulteren in de getoonde signaalamplitudegolfvorm.

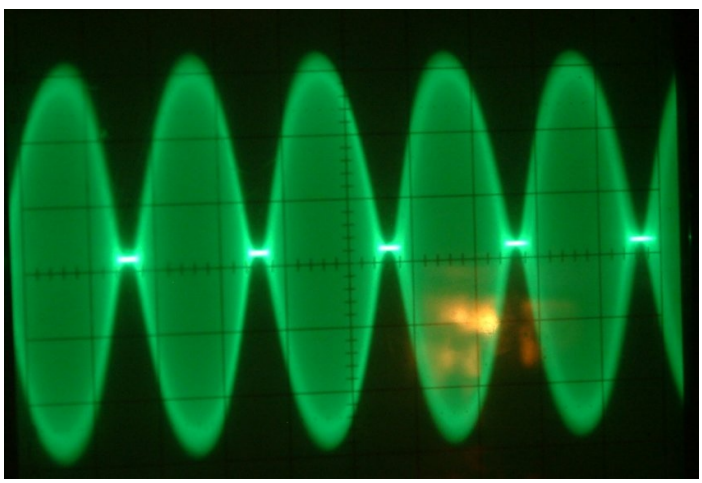
Verkeerde afstellingen van de zender kunnen gemakkelijk worden waargenomen, zoals de volgende afbeeldingen laten zien. De eerste is een versterker die overstuurd is:



Overstuurde SSB zender dicht tegen verzadiging

Dit is een voorbeeld van de klassieke "flat topping". Ten minste één versterkertrap in de stuurtrappen werkt in verzadiging en kan dus geen goede versterking van de hogere amplituden van het signaal leveren. De zender zal erg vervormd klinken. Er zal ook een hoge reeks IM3-producten verschijnen, waardoor je signaal veel breder is dan acceptabel.

De volgende afbeelding toont een SSB-versterker met onvoldoende bias-instelling:



Verkeerde bias in SSB-stuurtrappen, verhoging van ruststroom sterk aanbevolen!

De ruststroominstelling is in dit geval veel te laag, zodat gebieden met een lagere spanning van de amplitude niet voldoende worden versterkt. Bias-instelling moet worden verhoogd om een goede versterking te garanderen.

Om de piekoutput van je transceiver te meten, kan je de amplitudemeting van je oscilloscoop gebruiken. Deel de piek-piekspanning door 2,81 (twee keer de vierkantswortel van 2) om p.-p-spanning om te zetten in effectieve spanning, dan kwadrateren en het resultaat delen door de weerstandsbelasting van 50 ohm.

Veff. = Vpp. / 2.81 (I)

$P = Veff.^2 / R$ (II)

Of je gebruikt een VTVM (valve tube voltmeter, buisvoltmeter) met een HF-meetkop die je de RMS-waarde geeft van de uitgangsspanning van de zender.

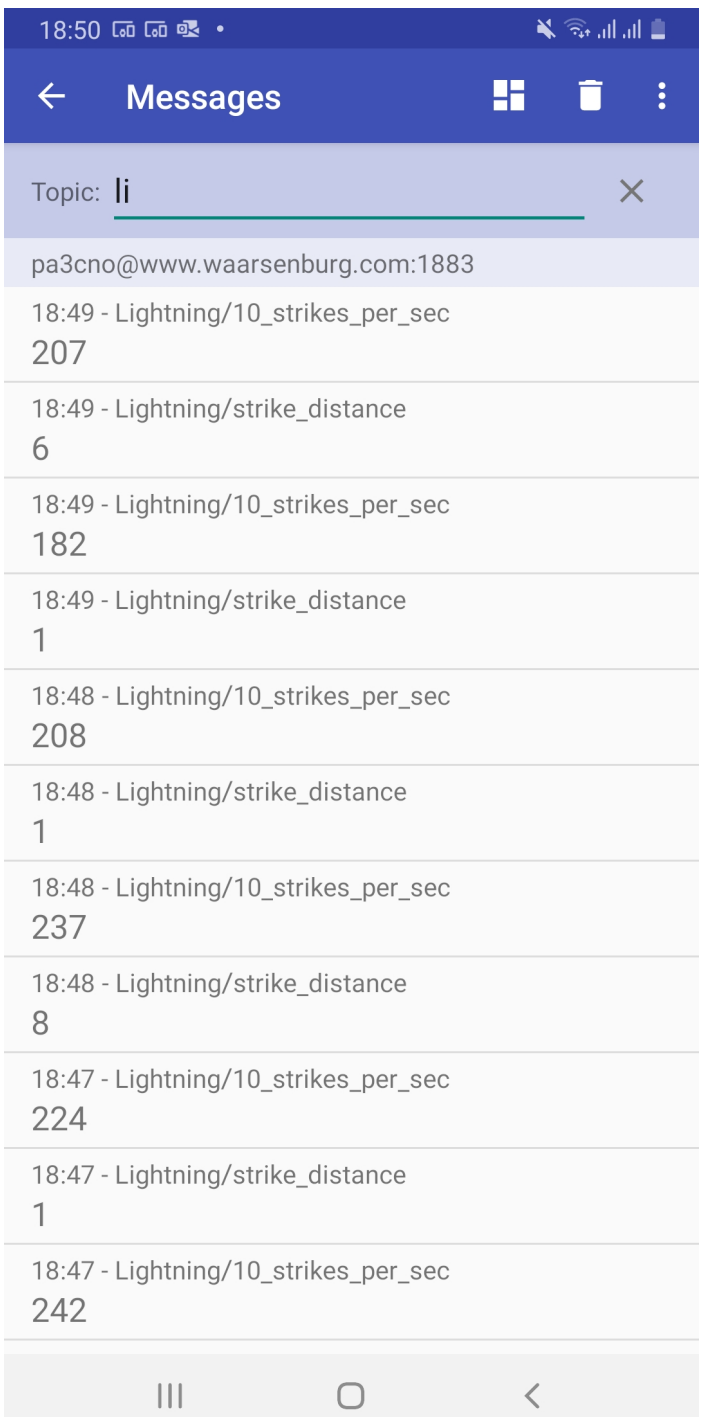
Houd er rekening mee: het vermogen dat je krijgt is de helft van het maximale uitgangsvermogen van je zender!

Voor wat betreft de bouw: maak 'm volgens de dode kever methode op een stukje printplaat (voor gevorderden), of koop de kale print, of gemonteerde print of complete apparaat in de webshop van DK7IH. Voor de prijs hoef je het niet te laten.

PA3CNO's Blog

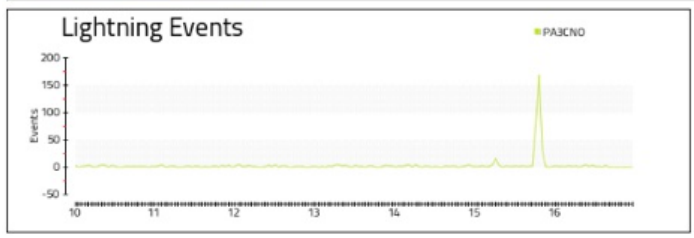
Ook ik ben aan het knutselen geslagen met MQTT, na door Robert enthousiast gemaakt te zijn door de mogelijkheden. Omdat ik al een Linux server thuis heb draaien die dienst doet als firewall, DHCP server, file server, media server, web server, database server en proxy server, kon er ook nog wel een MQTT server bij. Ik installeerde Mosquitto, alleen niet als docker container maar gewoon als package. Uiteraard heb ik er authenticatie op gezet, en met MQTT Explorer kon ik al gauw de default configuratie van de server zien. Het eerste apparaat wat ik verbouwde, was de onweerdetector. Ik voegde een paar regels code toe zodat hij bij detectie van een bliksem ontlading dat niet alleen op het display zette, maar ook naar mijn MQTT server stuurde. Met de app MQTT Push Client op mijn Android telefoon kon ik de berichten uitlezen. Je kunt in de app aangeven of je een alarmsignaal bij ontvangst van een topic wil krijgen of niet. Ik had dat eerst aan staan, maar dat heeft een heel lage WAF (Wife Acceptance Factor). Er vinden over de dag zo af en toe best wel wat knetters plaats in de atmosfeer (of in huis, kan natuurlijk ook...) en dat gepiep van je telefoon voegt niets toe als er niet echt onweer aankomt. Dus splitste ik de meldingen in tweeën: de ene soort melding

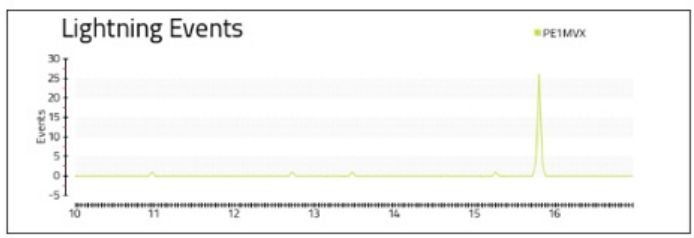
geeft een ontlading aan met daarbij de afstand en daar staat geen alarm op, en de tweede soort melding gedraagt zich net zo als het donderwolkje op de RAZ onweerswebsite: als er binnen een half uur een x-aantal ontladingen plaatsvinden, dan wordt deze melding gestuurd en daar staat wél een alarm op.



Op 15 augustus zaten we met de kleinkinderen te eten toen mijn telefoon opeens begon te piepen. En ja hoor, de onweerdetector helemaal in paniek. Aan de lucht was nog niet veel te zien, maar op de buienradar wel. We hebben snel alle kleden en stoelkussens binnengehaald en ik heb de antenne losgekoppeld, en een half uur later ging het los.

PI4RAZ Lightning Database

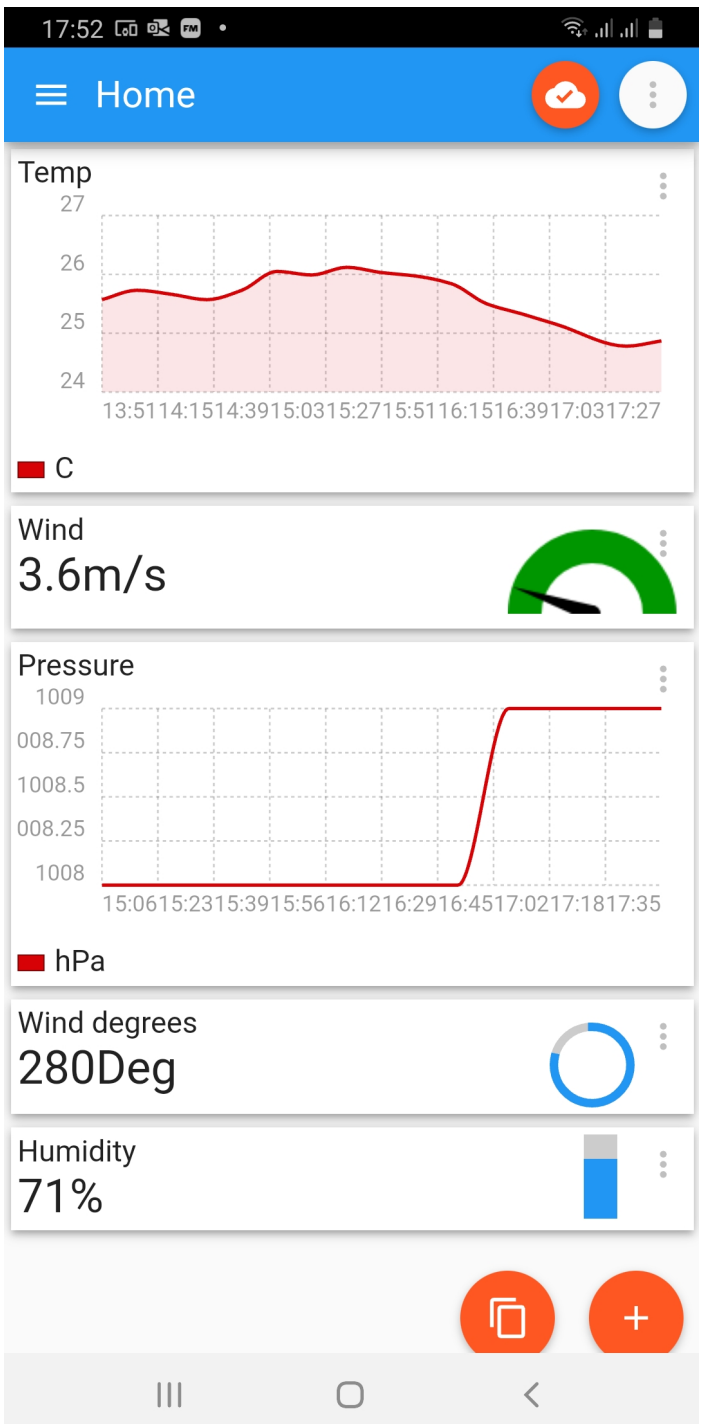




Zoals je ziet was ik niet de enige die het onweer constateerde. De onweerdetector heeft zelf ook wel een pieper aan boord, maar die staat bij mij altijd uit (iets met WAF) en bovendien staat dat ding in de shack en had ik 'm nooit gehoord met twee enthousiaste kleinkinderen aan tafel. Maar door de melding op mijn telefoon werd ik op tijd gewaarschuwd voor het naderende onweer. Merk overigens op dat ik in het plaatje hiernaast een filter op de topics had gezet om alleen de onweerwaarschuwingen waar te geven. Het intikken van 'li' is al genoeg om op Lightning te filteren.

Het volgende apparaat wat er aan moest geloven was het weerstation dat in de RAZzies van juni dit jaar beschreven is. Een paar regels code erbij zorgde ervoor dat ook het weerstation zich ging melden bij de MQTT server. Ik stuur 11 parameters door, waaronder de locatie. In eerste instantie stuurde ik alleen de weerparameters door, maar als de XYL het weerstation omschakelt naar b.v. de locatie van onze zoon die bij Brussel woont, krijg ik op mijn telefoon allemaal enthousiaste weerparameters door zonder dat ik door heb dat dat niet voor Zoetermeer geldt... Daarom wordt nu ook de locatie doorgegeven. Dit alles elk kwartier. Dat zijn 1056 meldingen per dag. Maar die lijst met

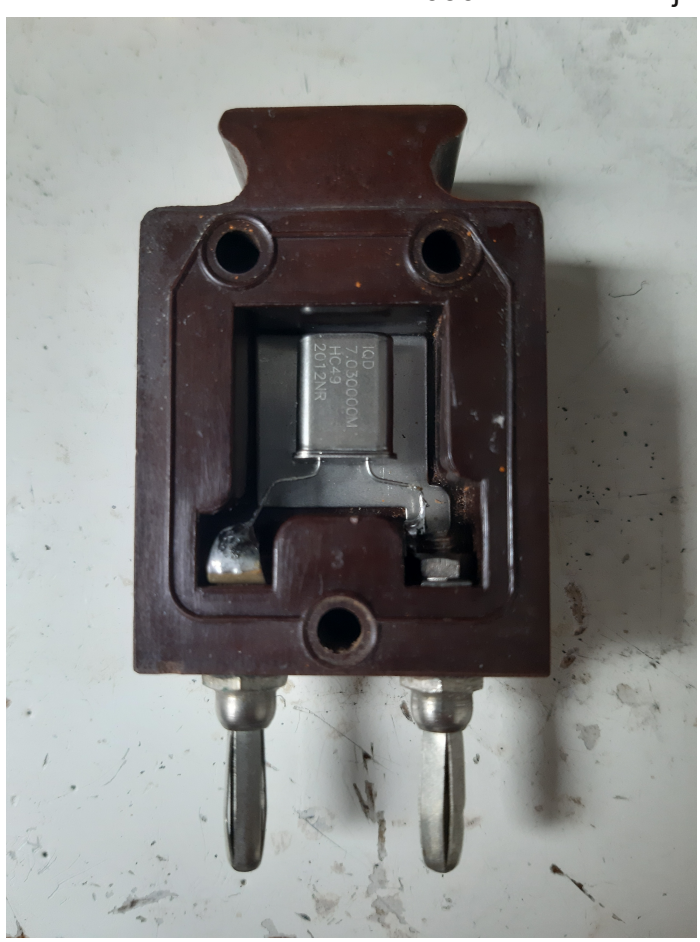
meldingen is best lastig te lezen. In de Playstore (Android) vond ik nog een app, IoTMQTTPanel. Daarmee kan je allerlei grafische weergaven aan je Topic parameters toevoegen. Dat heb ik voor 5 veelgebruikte parameters gedaan: Temperatuur, luchtdruk, vochtigheid, windsnelheid en windrichting. En dan krijg je zoiets:



Je kunt nu dus ook een stukje historie zien; ik configureerde zowel de temperatuur als de luchtdruk voor het weergeven van 10 punten die overeenkomen met een kwartier: in het totaal dus 2,5 uur zodat je een idee hebt van de trend.

De windweergave is een Gauge weergave, waarbij ik de achtergrond bij 6m/s oranje laat worden (ca. windkracht 4) en rood bij 15m/s (ca. windkracht 7). Een echte windroos was er niet in de grafische weergaven, maar een 'progress indicator' met een maximum op 360 graden geeft mooi de windrichting weer. Leuk toch? Dit is nog maar een fractie wat je met MQTT kunt doen.

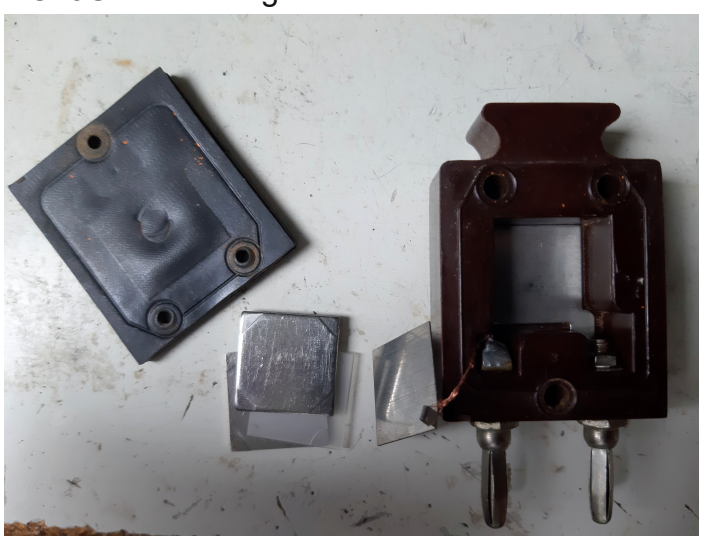
Even weer iets heel anders. Al een tijdje is er in de Paraset groep sprake van het laten maken van kristallen voor een Paraset frequentie op 40m en 80m zodat we elkaar wat makkelijker kunnen vinden. Ik heb de QRP frequenties 3560 en 7030 voorgesteld. De Parasets werken met FT-171-b kristallen. Dat zijn van die blokken met banaanstekkerpennen op stopcontact afstand (19mm). Niet in het stopcontact steken natuurlijk, ook al past het. Het duurde mij allemaal veel te lang. Ik had zoals ik al eerder meldde een reeks 80m kristallen op de kop getikt, waaronder eentje op 3750kHz. Daar ga ik nooit CW doen, dus ik besloot er eentje op te offeren: ik bestelde een 7030kHz kristalletje



ergens in Italië en soldeerde die in de kristalhouder.

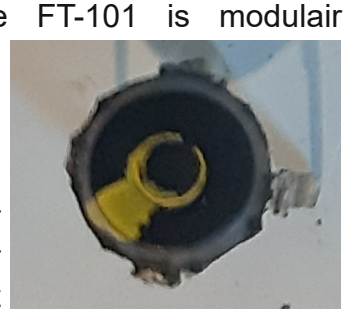
Ik hoor historici al steigeren omdat ik een goed 80m kristal heb opgeofferd door er een 40m kristal in te solderen, maar ik kan dat kristal moeilijk ten eeuwige dagen op de plank laten liggen tot mijn kinderen het na mijn overlijden in de container storten.

Puristen zullen zeggen dat zo'n HC49U kristal niet bestand is tegen het geweld van een penthode oscillator, maar dat valt in de praktijk gelukkig mee. Ik krijg ruim 4W uit de Paraset op 40m met dit kristal en tot nu toe blijft-ie heel. Om je een idee te geven van het verschil tussen een HC49U en een originele FT-171-b:



Dat melkglazen plaatje tussen die twee metalen vierkantjes is het kristal. Dat zit normaal tussen die twee metalen plaatjes ingeklemd en het geheel wordt met een behoorlijk grofstoffelijke veer aangedrukt als de behuizing dicht geschroefd wordt. Alleen het kristallen plaatje is al groter dan een hele HC49U, en daarom zijn dit soort originele kristallen veel beter bestand tegen de stromen die een penthode oscillator er doorheen laat lopen. Maar goed, ik heb nu een 40m kristal voor 7030kHz. 3560kHz had ik al in FT243 uitvoering en ik heb een verloopje van FT243 naar 19mm stekker zodat ik die kristallen ook kan gebruiken in de Paraset. Ik heb er nog geen verbindingen mee gemaakt op 40m, maar dat heeft meer te maken met de temperaturen in de shack dan met de werking van de Paraset met 40m kristal...

Nog een laatste onderwerp. Ik was met mijn meetzender aan het stoeien en had daar de FT-101 Boat Anchor aan geknoopt. Eigenlijk om eens te kijken of -73dBm op alle banden wel S9 op de meter zet. Dat klopte lang niet op alle banden, maar wat mij voornamelijk opviel was dat S9 nogal ruiserig klonk. Bij S9 verwacht je een nagenoeg ruisvrij signaal. Ik draaide de meetzender wat omlaag en zo rond de -95dB was het signaal al niet meer te horen. Dat is ca. 7uV antennesignaal en dat is ronduit beroerd. De B2 replica is nog beter; die heeft ca. 3uV gevoeligheid en dat is al niet best naar huidige maatstaven. Kortom: de set was "stone deaf" zoals de Engelsen zo mooi zeggen. Ik begon eens wat te googlen en vond al gauw dat in het front-end van een FT-101 een 3SK40 dual gate MOSFET zit; een type zonder enige bescherming. Een van de problemen van de FT-101 is het beschadigen van deze FET als gevolg van statische spanningen op de antenne ingang, met doofheid tot gevolg. Ik voer weer blind op de beweringen van Google, nam dit voor waar aan en bestelde op eBay zonder verder onderzoek een zakje met 5 3SK40's. Na een maand of 2 kwamen ze binnen en ik haalde de FT-101 uit elkaar. De FT-101 is modulair opgebouwd (een revolutie in die tijd) dus ik kon de HF module er zo uittrekken. En wat bleek? Een vorige eigenaar had er een of ander droppie zonder opschrift ingeprakt, met waarschijnlijk de aanname dat dit de gevoeligheid zou verbeteren. Nou, integendeel. Eerst maar de originele 3SK40 erin geplaatst, waarbij het een voordeel is dat de desbetreffende FET in een voetje zit:



Dat scheelde een slok op een borrel. In de eerste onderstaande screenshot zie je wat er nodig was op de diverse banden om S9 op de meter te krijgen. Dat klopte alleen vanaf 20m redelijk. Vooral 40m was slecht. Zie de kolom "oud". De kolom "nieuw" geeft de waardes aan na vervanging van het droppie door de 3SK40. 15dB verbetering! Dat is 2,5 S-punt. Dat een Japanse S-meter in de 80-jaren niets waard was, blijkt uit een paar andere metingen die ik deed na vervanging van de FET. S9+20 op de meter is maar 7 echte dB's meer dan S9, geen 20. Hetzelfde voor +40 en +60dB: in het echt is dat +16 en +28. Omgekeerd: S7 zou t.o.v. de -88dB die nu S9 aangeeft, -100dB moeten zijn. Dat is -93. S5 zou -112 moeten zijn en dat is -96. Zo zou S3 dan -124dB moeten zijn (-100) en S1 -136dB (-105). Kortom, een echte leugen-detector.

Na het vervangen van de FET regelde ik zo goed en zo kwaad als het ging de S-meter weer op S9 bij -73dB op 20m en regelde alle ingangskringen nog een keer na. Het resultaat staat nu in de onderste tabel. De meter zit er nog een half S-punt naast maar de potmeter om dat af te regelen is nogal itchy dus ik laat het maar zo. Nauwkeurig is-ie sowieso niet. Je ziet dat 80m en 20-15-10m nu aardig goed zijn. 160m blijft 10dB achter en dat is 1,5 S-punt. Maar de achtergrondrotzooi op 160m is dusdanig hoog dat een signaal toch al sterk moet zijn wil je het horen. Tenminste, in Zoetermeer. En mijn 2x13m Inverted-V is ook niet echt een goede antenne voor 160m. Dat 40m 16dB achterblijft is wel jammer. Dat is 2,5 S-punt. Ik vermoed dat de reden gelegen is in de zuigkringen die de FT-101 heeft op 5900kHz. Die zitten daar om de doorbraak van die signalen als gevolg van menging met de VFO naar het middenfrequent tegen te gaan, maar waarschijnlijk is de demping op 7MHz nog merkbaar. Aan de andere kant: zolang een signaal boven de S3 is, kan ik het nog horen. In de praktijk meer dan voldoende voor mij. Op foxtango.org wordt nog wel een FET aangeboden die wél beveiliging op de gates heeft en die een stuk beter zou zijn dan de 3SK40. Voorlopig heb ik er nog 4. Als ik die opgestookt heb, ga ik die andere FET wel eens overwegen...

	oud	nieuw	+20	+40	+60	S7	S5	S3	S1
160m	-60	-69							
80m	-65	-82							
40m	-51	-75							
20m	-73	-88	-81	-72	-60	-93	-96	-100	-105
15m	-72	-85							
10m	-73	-85							

Gevoeligheidsmeting. De tabel Oud geeft aan hoeveel signaal er nodig was om S9 op de meter te krijgen. -73dB is S9 volgens de definitie. 1 S-punt is 6dB. De tabel Nieuw geeft de waarden aan na vervanging van de FET. Daarna is nog gekeken wat er nodig is voor +20dB, +40dB, +60dB, S7, S5, S3 en S1.

160m		-67							
80m		-77							
40m		-61							
20m		77	-60	-35	-16	-85	-92	-98	-106
15m		-76							
10m		-75							

Meting na opnieuw afregelen van zowel S-meter als ingangskringen. De meter zit er een half S-punt naast, maar dat zal in de praktijk niet hinderlijk zijn. Merk ook op dat +20, +40 en +60dB nu veel beter kloppen! Wat enorm verbeterd is, is dat ik een signaal van -120dBm nog goed kan horen, waar dat eerst -95dBm was. En dat is een verschil van dag en nacht.

Afdelingsnieuws

Allereerst een ingezonden mededeling: **N-cursus Radioclub De Bevelanden**. Radioclub De Bevelanden organiseert ook komende winter weer een cursus voor de N-machtiging. De cursus bestaat uit 17 digitale lessen via Google Meet, start op 24 oktober 2022 en is op tijd klaar voor het N-examen van 1 maart 2023 in Houten. Aan de cursus is een vergoeding van €15,00 voor de clubkas van de radioclub verbonden. Verder gaat de cursus er vanuit dat de deelnemers in het bezit zijn van het Cursusboek voor het N-examen van de Veron. Wil je meer weten stuur dan een bericht naar Ruud Jongeling, pe2bs@kpnmail.nl

En wij gaan weer beginnen! In september gaan we weer bij elkaar komen op de woensdagen 14 en 28 september. Later kan ook echt niet... IJs en weder dienende is de QSL-manager aanwezig op de 14e zodat je je kaarten kunt inleveren en ophalen. Wat we waarschijnlijk ook meteen gaan doen, is een poging om de antenne op het clubhuis weer te restaureren. Na twee jaar Corona zonder onderhoud is er niet veel meer over van de draadantenne die over het clubhuis liep. Met wat medewerking en enige handige materialen zoals parasolvoeten, mastjes, draad en een snoeischaar, zou de antenne weer opgebouwd moeten kunnen worden. We hopen iedereen weer in goede gezondheid te ontmoeten in ons clubhuis van de Minigolf Zoetermeer in het Vernède sportpark.