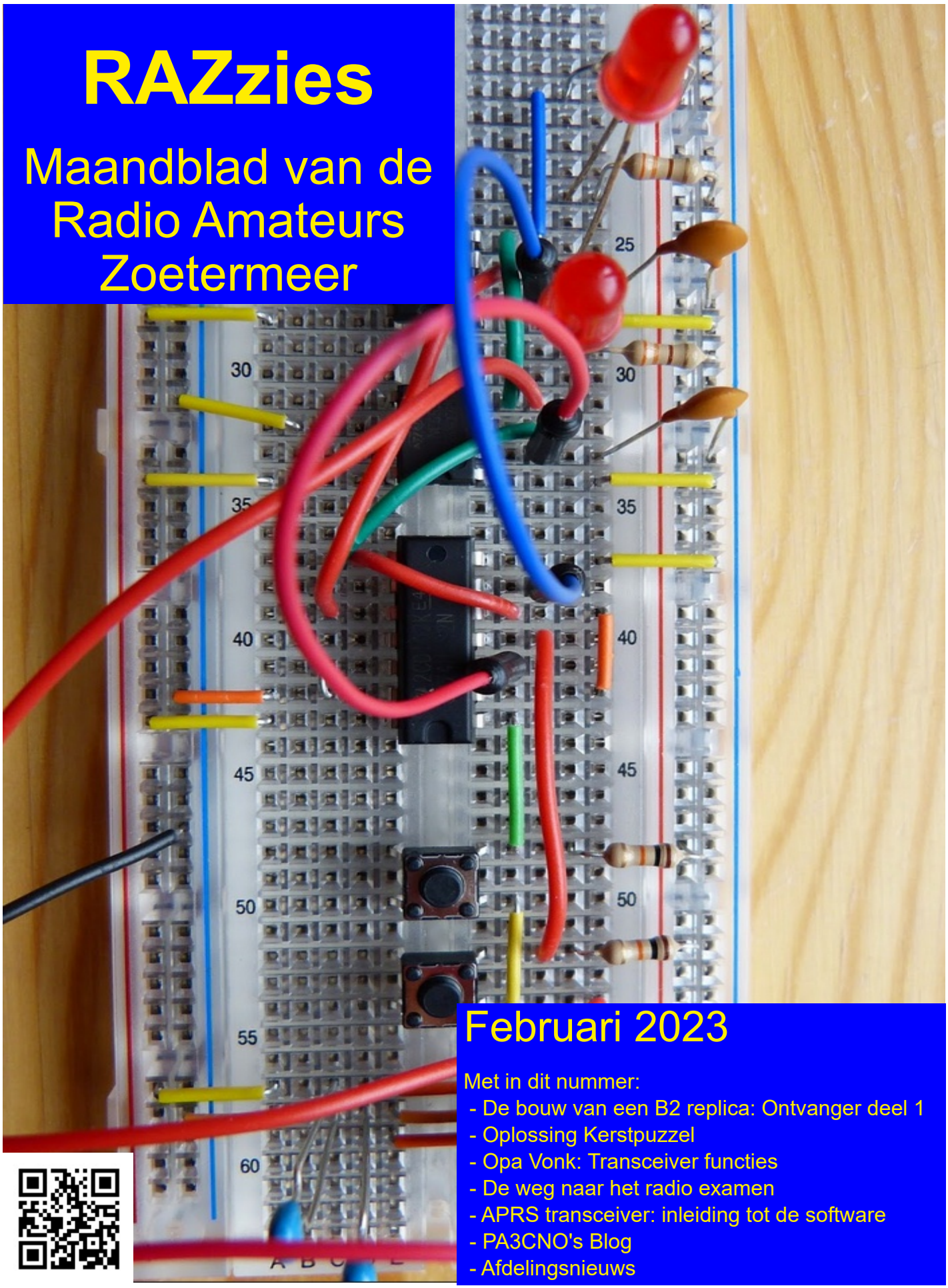
RAZzies

Maandblad van de Radio Amateurs Zoetermeer



Februari 2023

Met in dit nummer:

- De bouw van een B2 replica: Ontvanger deel 1
- Oplossing Kerstpuzzel
- Opa Vonk: Transceiver functies
- De weg naar het radio examen
- APRS transceiver: inleiding tot de software
- 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

Het nieuwe jaar is begonnen en dat betekent ook een nieuw onderkomen voor onze club. Daar zijn we gelukkig qua onderkomen veel beter van geworden en qua financiën niet veel slechter, dus we zijn blij. Deze maand waren er ook veel bijdragen van andere amateurs en dat maakt mij gelukkig: ik hoef dan alleen maar te be- en verwerken, en niet te verzinnen en te schrijven. Ga zo door!

De condities blijven verbazen. Waar we een paar jaar geleden nog geen deuk in een pakje boter sloegen en we al blij moesten zijn als er binnen Europa nog wat te werken was, worden nu op 10m de meest waanzinnige afstanden afgelegd. Amerika is op 10m geen probleem, ook Zuid-Amerika niet, en waar voor mij meestal CW de enige manier was om nog wat DX te werken, gaat het nu ook in phone gewoon goed. Dat wil zeggen: als niet half Europa de lineairs tot het uiterste pusht om er toch nog overheen te komen. Een tip: gebruik je een lineair, monitor dan ook eens je uitgangssignaal. Dat kan al met een eenvoudige HF detector (diode en een C) aangesloten op een oscilloscoop. Zie je afgeplatte toppen op de scoop, dan ben je de zaak aan het oversturen. Niet alleen wordt je signaal onverstaanbaar, maar je stoort ook nog eens andere amateurs door je splatter. Met mijn 100W in een eenvoudige Inverted-V gaat het ook...

De bouw van een B2 replica: Ontvanger deel 1

Nadat de voeding gereed en operationeel is, is het tijd voor de volgende module: de ontvanger. Dat is waarschijnlijk het meest complexe deel van het project. Niet alleen elektronisch, maar ook mechanisch. Terwijl de verf van de voeding aan het drogen was, begon ik met het bestuderen van de ontvanger.

De ontvanger bestaat uit twee delen, feitelijk gebouwd op twee separate chassis, zie de foto van het origineel op de volgende bladzijde.

Het rechter chassis (gezien vanaf de achterzijde) is de feitelijke ontvanger met HF versterking, lokale oscillator, mixer (allemaal gerealiseerd met slechts één buis) en de eerste MF-transformator. Het linker chassis bevat het MF deel met BFO een laagfrequent versterker, gerealiseerd met 3 buizen. Vooral het rechter chassis is complex: deze is opgebouwd in 3 lagen en is heel compact. De afstemschaal zit daar weer achter, aangedreven door een snaar en een knop met vertraging. En dan 3 variabele zelfinducties voor het ontvanger frontend en 3 variabele zelfinducties voor de Lokale Oscillator, plus een dubbele MF-transformator op 470kHz. Dat wordt de grootste uitdaging. Omdat ik al een hele fraaie knop met vertraging had gekocht op Ebay, hoefde ik geen snaar-aangedreven afstemschaal te maken.

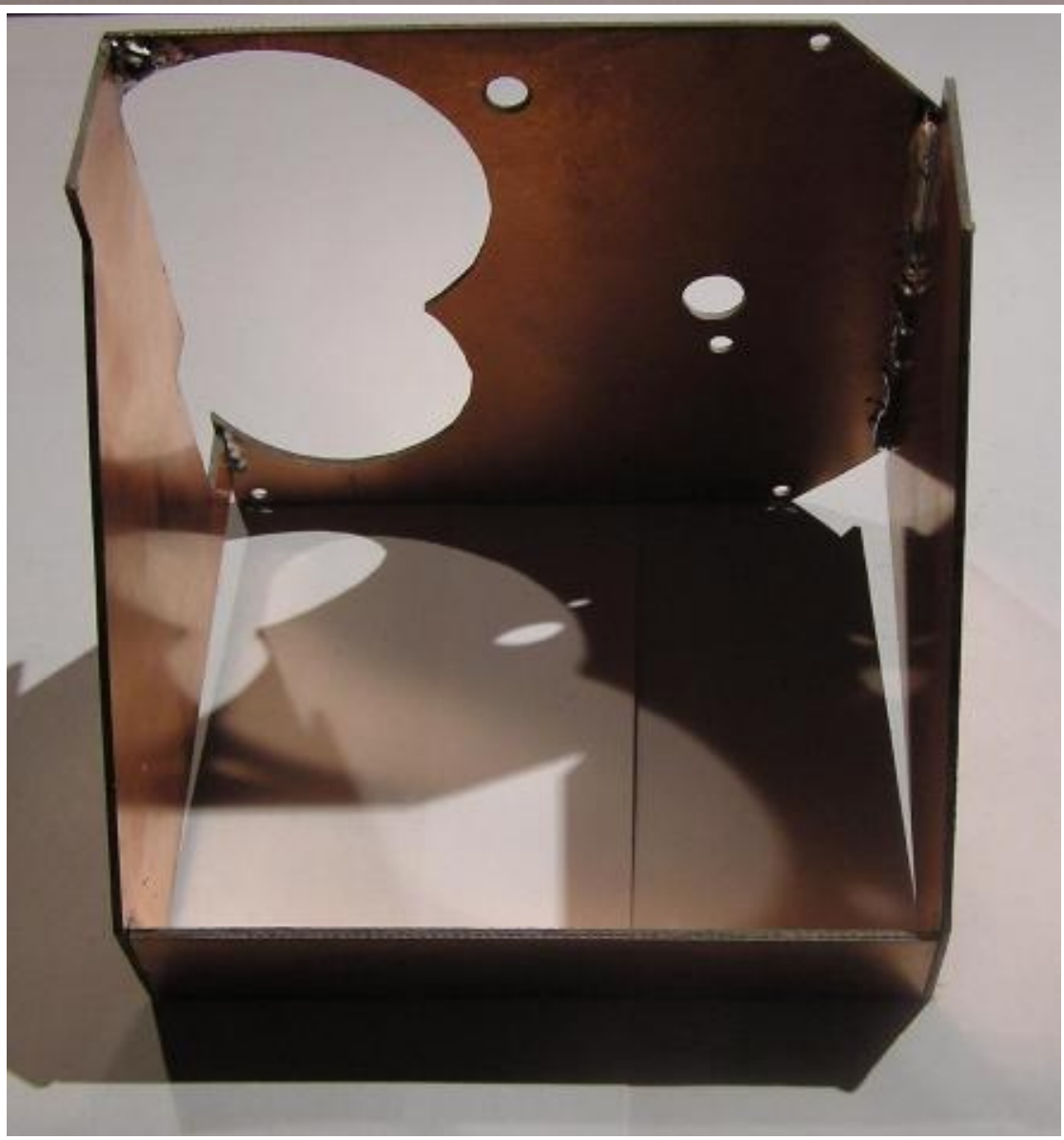
De knop met vertraging kan direct op de as van de afstemcondensator gemonteerd worden. Dan moet nog wel de afstemschaal aangedreven worden. Om dat te realiseren bestelde ik een stel tandwielen bij Conrad. Mans PA2HGJ was zo vriendelijk om op de draaibank wat overtollig materiaal van de tandwielen te verwijderen, en daarmee had ik een mooi tandwiel dat op de as van de afstemknop gemonteerd kon worden. Nadeel: de originele afstemknop is 45mm in diameter en de mijne 72mm... Voordeel: in ruil daarvoor krijg ik een 1:90 vertraging.

Dus, laten we eerst maar eens het mechanische deel voor elkaar krijgen. Dat betekent rekenen, tekenen, controleren of alle montagegaten op de juiste plek zitten en dan kan het boren in het deksel beginnen, te beginnen met de kant van de ontvanger, zie de eerste foto op de volgende bladzijde.

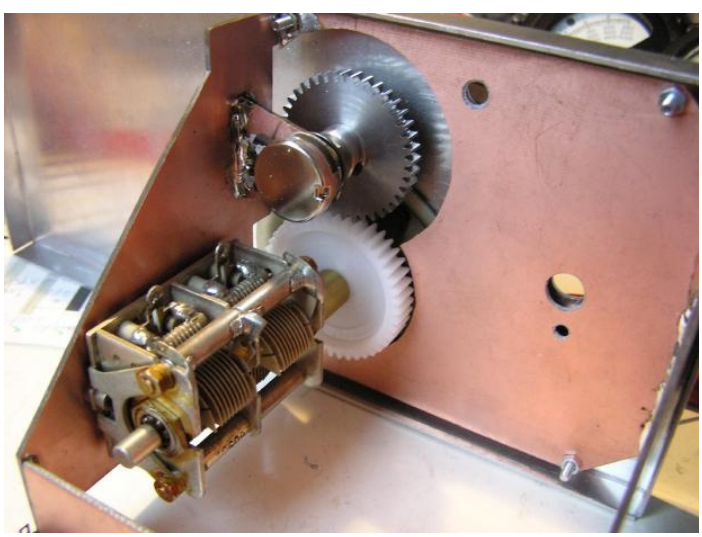
Het gat links boven is voor de antenne aansluiting. Als de ontvanger stand-alone gebruikt wordt, is dat de plek waar de antenne mee verbonden wordt. Wordt de ontvanger in combinatie met de zender gebruikt, dan wordt hier een banaanstekker ingeprikt die met een draadje afkomstig is van de zender.

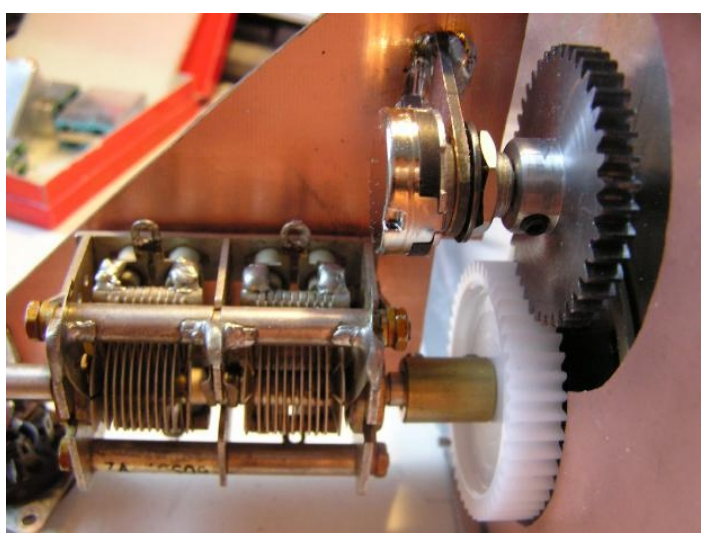
Het grote gat is voor de afstemknop. Links daarvan zit het gat voor de bandschakelaar. De vier M3 gaten in een rechthoek zijn voor het monteren van het sub-chassis. Die wordt bevestigd met 5mm afstandsbusjes waardoor alle bevestigingsmoeren en -bouten achter het deksel komen te liggen. Behalve de antenne connector, die direct op het deksel bevestigd wordt.

Het subchassis bevat diverse uitsparingen voor de tandwlelen, zie de tweede foto op de volgende bladzijde.



Het frame waar de buisvoet op gemonteerd wordt, zit nog niet op zijn plek. Oplettende lezers zal het opgevallen zijn dat als je naar het plaatje van het origineel kijkt, het rechter sub-chassis ongeveer een centimeter minder hoog is dan het linker chassis. Dat is omdat er een strip met trimmer condensatoren onder zit:





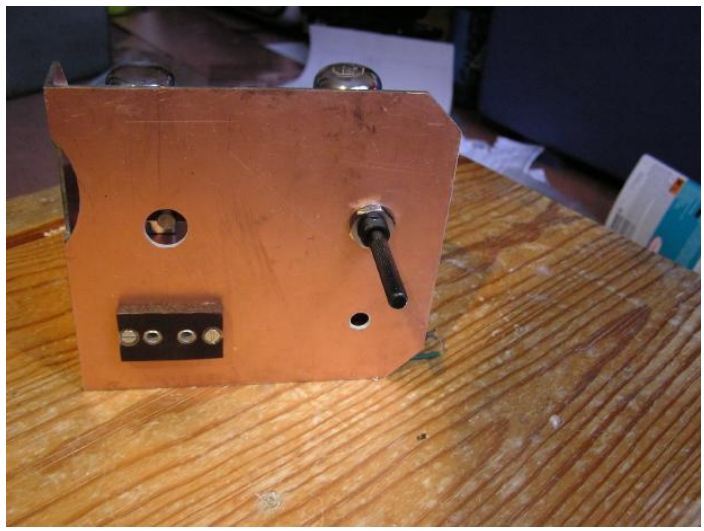
Mijn subchassis zijn gemaakt van dubbelzijdig printplaat. Stevig maar toch makkelijk te bewerken. Voordat de onderdelen gemonteerd worden, ga ik de printplaten grijs verven zodat het een beetje op metaal lijkt. Eerst moet ik tijdelijk de afstemknop monteren, om de knop in lijn te brengen met de afstemcondensator. Dat wordt nog kritisch. En dan kan het tweede subchassis gemaakt worden.

Ik misbruikte een potmeter als lager voor het tandwiel waarop de schaal gemonteerd gaat worden (die wordt op het tandwiel gelijmd). Ik maakte een steun van printplaat en soldeerde die tegen de zijkant van het chassis. Uiteindelijk paste het allemaal perfect, zie de foto hierboven. Het tandwiel dat de schaal moet ondersteunen wordt zo dicht als mogelijk tegen het front gemonteerd.

Daarmee is het mechanische werk voor de ontvanger/Lokale Oscillator gereed. Tijd om het tweede subchassis voor de MF versterker, BFO en LF te bouwen. Er moeten gaten geboord worden voor de buisvoeten, de MF transformatoren, de LF transformator, potmeters en draaddoorvoeren. De BFO-condensator moet vrij van massa opgesteld worden, wat gerealiseerd is met een speciale steun, wederom gemaakt van printplaat. Ik gebruikte een frees om de condensator vrij van massa te maken. Ook hier wordt weer een asverlenger gebruikt, met aan de andere kant van de condensator een plastic as om het handeffect te verminderen. Met de componenten tijdelijk op hun plek gezet ziet het er al leuk uit:

Bovenaanzicht van het tweede subchassis.

Achteraanzicht. Alles past precies.



Vooraanzicht. Aan de rechterkant de volume potmeter en het gat voor de voedingskabel; aan de linkerkant het gat voor de asverlenger en de koptelefoon aansluiting.

De ontvanger is een enkelsuper met een middenfrequent van 470kHz en een frequentiebereik van 3,1 - 15,5 MHz in 3 banden. Van achteren naar voren bouwen is meestal het handigst, omdat je dan deel voor deel kunt testen. In dit geval dus het MF/ BFO/ LF deel. Zie het schema op de volgende bladzijde.

De eerste buis is een gewone 470kHz MF-versterker. Via een dubbele MF-transformator met instelbare koppeling komt het signaal op de tweede buis terecht. En die heeft maar liefst 3 functies. MF-versterker, BFO en mixer. De BFO kan met een ingenieuze truc uitgeschakeld worden: op het uiteinde van het vaantje van de BFO-afstemcondensator zit een bolletje soldeer. Als de BFO-condensator naar één van de uiteinden wordt gedraaid (+ of - 3kHz) maakt het bolletje soldeer kortsluiting tussen de platen van de condensator en de BFO is uitgeschakeld.

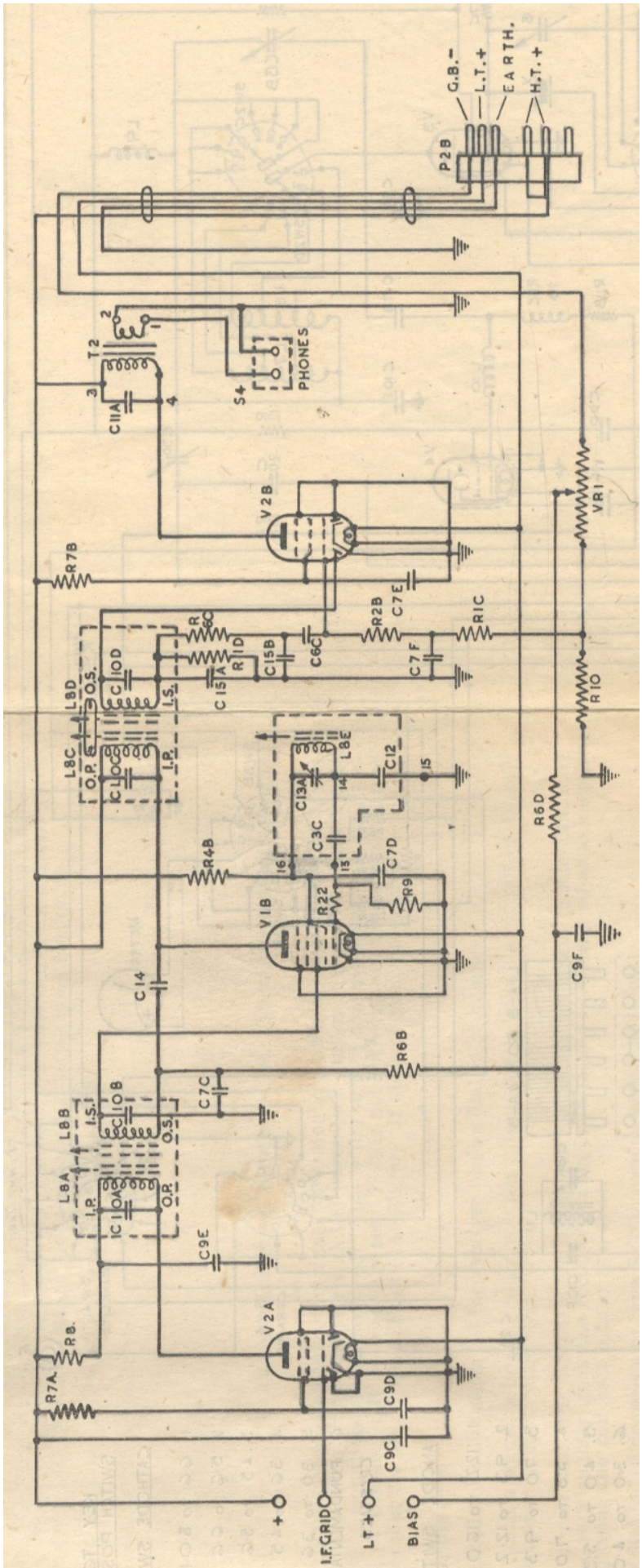
Tevens wordt een deel van het signaal van de tweede buis teruggekoppeld naar de eerste kring. Door deze meekoppeling wordt bij grote versterking de selectiviteit vergroot door ontdemping van de kring - zoals bij super-regeneratieve ontvangers het geval is. De versterking wordt dan ook geregeld door het negatief van de eerste drie buizen te regelen en

niet door het LF te regelen, zoals normaal bij een LF versterker. De laatste buis dient als detector en LF versterker. Als LF transformator gebruikte ik een 220/9V transformatortje uit de dump. Dat levert een impedantietransformatie van bijna 600 op en dat geeft aan de kant van de buis een impedantie van bijna 20k bij het gebruik van een 32 Ohm koptelefoon.

Op een paar 0,1uF condensatoren na zijn alle componenten ex equipment. Ik soldeerde een paar draden met banaanstekkers aan het MF subchassis en verbond mijn HF generator via een 1nF condensator met de MF ingant, na de generator op precies 470kHz gezet te hebben. Ik begon met spanning op de gloeidraden te zetten en wachtte een uurtje voordat ik de 250V aansloot. De buizen zijn tenminste 40 jaar niet meer gebruikt, als ze al gebruikt zijn, en ik wilde ze wat tijd geven om aan hun nieuwe taak te wennen (door de gloeidraden aan te zetten verdwijnen eventuele restjes zuurstof in de buis). Daarna werden negatieve roosterspanning en 250V anodespanning aangesloten en was het tijd voor de smoke test.

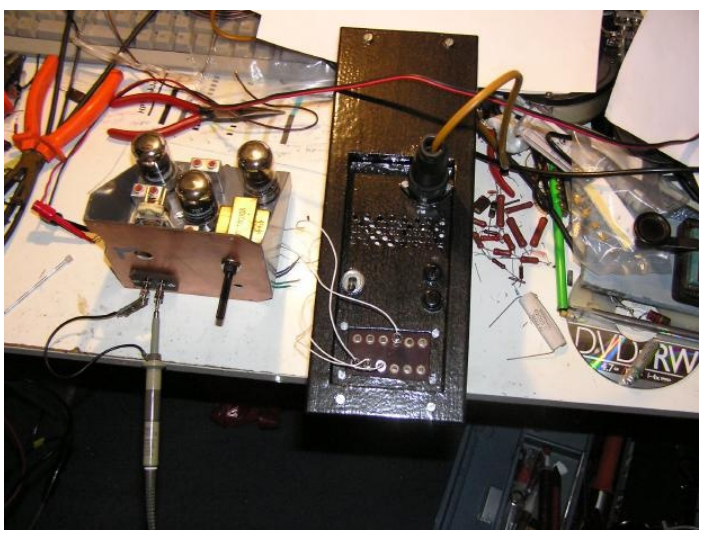
Er explodeerde niets, maar de scope liet rotzooi zien... De reden was de massa aansluiting van de HF generator. Ik stak een stukje draad tussen te platen van de BFO condensator zodat die even stil was en begon met afregelen van de MF transformatoren. Heel voorzichtig, want het afregeltooltje is gemaakt van aluminium en de BFO condensator zit in zijn geheel aan de 250V. En ik voel er niets voor om gefrituurd te worden door het per ongeluk aanraken van de BFO condensator...

Het afregelen gaf geen problemen. Zoals gezegd doet de "volume"-potmeter wat anders dan alleen maar het audio verzwakken: hij regelt het negatief van de eerste drie buizen. De laatste buis is de



enige met een vast negatief. Door het variëren van het negatief verandert ook de versterking. én de terugkoppeling van de tweede buis naar de MF transformator! Ik maakte de 6pF terugkoppelcondensator van een smal reepje dubbelzijdig printplaat door daar stukjes van af te knippen tot de juiste waarde bereikt was. De gevoeligheid lag tussen de 3 en 10 µV bij gebruik van een 30% Amplitude gemoduleerde draaggolf. En dat is nog zonder de eerste trap. Best goed, als je het mij vraagt.

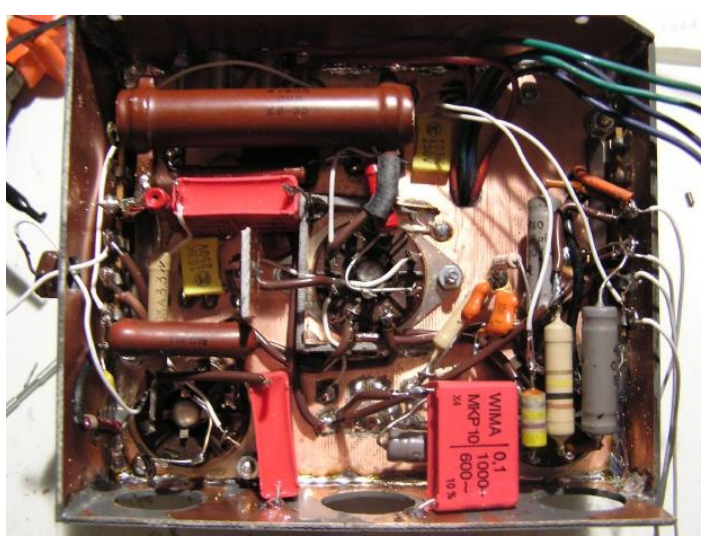
Vervolgens verwijderde ik de draad die de BFO afstemcondensator kortsloot en zette de condensator op de helft van zijn regelbereik, Ik regelde daarna de spoel af op zero beat. Omdat op dit moment de asverlenger nog niet gemonteerd is, kan ik het regelbereik van de BFO niet testen. Er staat 250V op de condensator dus daar kan ik maar beter afblijven... Dus ook de BFO werkt en staat als een huis op frequentie. Een foto impressie:



MF/BFO/LF deel op de testbank

Achteraanzicht van het MF/LF deel

Bovenaanzicht. Let op de afstand tussen de MF trafo en de BFO condensator..



Interieur van het MF/LF deel

Oplossing Kerstpuzzel

Was-ie nou zo moeilijk of is de belangstelling voor puzzels een beetje verdwenen? 14 inzendingen, waarvan 12 goed. Dat is niet veel voor de hoeveelheid lezers van dit blad. Voor degenen die 'm helemaal gemist hebben nog een keer de opgave:

Opa Vonk had een apparaat gekocht wat niet meer werkte omdat er op een print drie onderdelen ontbraken. Opa heeft de beschikking over zes onderdelen die op de print gezeten zouden kunnen hebben: een weerstand, een diode, een condensator, een spoel, een zener en een varistor. Opa probeert een drietal combinaties en de tester geeft de volgende uitslag:

Zener Varistor Condensator (1G, 0V)
Zener Diode Spoel (0G, 1V)
Condensator weerstand zener (0G, 2V)

waarbij een G betekent dat een correct onderdeel ook op de correcte plek zit, en een V betekent dat een onderdeel wel in de schakeling hoort, maar niet op die plek. De oplossing was als volgt:

Uit regel 1 en 2 volgt dat de zener er niet in zit. Want in regel 1 zit er 1 goed, en als dat de zener zou zijn, zou die in regel 2 ook goed moeten zijn. Dus óf de condensator, óf de varistor is goed en zit op de goede plek.

Aangezien de zener niet meedoet, moet volgens regel 3, waar er 2 goede onderdelen op de verkeerde plek zitten, de weerstand en de condensator beiden in de schakeling zitten.

Uit regel 2 kunnen we tevens concluderen dat óf de diode, óf de spoel in de schakeling moeten zitten. Er is er maar één goed en die zit op de verkeerde plek. De zener is immers al geschrapt.

Als Condensator en Weerstand beiden in de schakeling zitten, en daarnaast óf een diode óf een spoel (regel 2), zit er dus ook geen Varistor in. Dat betekent in regel 1 dat de condensator in de schakeling zit, en op de goede plek, dus positie 3.

Als we weten dat op positie 3 de condensator zit, en volgens regel 3 de zener niet meedoet en de weerstand wel, maar op de verkeerde plek zit, moet de weerstand op positie 1 komen. Er zaten immers twee goede onderdelen niet op de goede plek en aangezien plek 3 voor de condensator is en de weerstand niet op plek 2 moet, kan deze alleen maar op plek 1 horen.

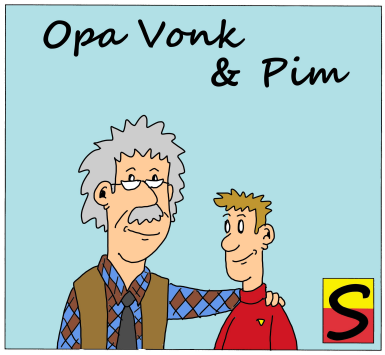
Blijft dat er nog een diode of een spoel in moet. Kijken we naar regel 2, dan zit daar één correct onderdeel op de verkeerde plek. Zou plek 2 inderdaad voor de diode zijn, dan had daar 1 onderdeel goed moeten zitten (1G), maar er staat 1V. Dus de diode is het niet: de spoel doet mee en zit op plek 2.

De oplossing is dus: Weerstand - spoel - condensator.

Na ons trekkingsalgoritme op de goede inzenders losgelaten te hebben zijn de volgende prijswinnaars uit de bus gekomen:

1 = Marc de Hoop
2 = Hans van Beek PA3GFD
3 = Marten Remmers PA3EKM

De prijswinnaars zijn inmiddels geïnformeerd en krijgen hun prijs zo snel mogelijk thuisgestuurd. Iedereen bedankt voor het meedoen en wellicht tot eind dit jaar met nieuwe kansen!



Opa vonk keek geïrriteerd op van zijn soldeerwerk vanwege het gepiep van zijn ontvanger, bediend door zijn kleinzoon Pim. Het gepiep werd dit keer niet veroorzaakt door ontvangen morseseinen, maar door het indrukken van de knoppen van Opa's transceiver waarbij elke druk op de knop een bevestigingspiep oplevert. En Pim drukte er nogal wat in. "Wat ben je in vredesnaam aan het doen?" vroeg Opa. "Ik probeer mijn weg te vinden in al die menu's en mogelijkheden, maar wat een doolhof is zo'n moderne transceiver. Die oude buizenzender van u heeft tenminste nog gewoon knoppen, maar van die menu's word je gek. En wat betekenen al die functies trouwens?" Opa legde zijn soldeerbout neer en liep naar Pim toe. "Ik zal je de meest gangbare functies van een transceiver uitleggen, en ook waar ze voor dienen. Niet alle functies zitten op alle transceivers, maar de meest belangrijke wel. Neem bijvoorbeeld de functies die je nu voor je neus hebt staan. Daar zie je A/B, A=B en SPL. In een pile-up zijn dat veel gebruikte functies. A/B schakelt om tussen VFO-A en VFO-B. Niet dat er twee VFO's in zitten, maar functioneel wel. Het is natuurlijk erg eenvoudig om twee geheugenplaatsen in de software te definiëren en de ene te gebruiken voor VFO-A en de andere voor VFO-B. Het enige wat deze functie doet, is de andere geheugenplaats uitlezen. De A=B functie doet wat de naam zegt: het maakt de niet-geselecteerde VFO gelijk aan de geselecteerde VFO, inclusief de mode. En de functie SPL staat voor Split. Deze drie functies werken nauw met elkaar samen. De praktijk: je luistert naar een DX (of special event) station. Doorgaans is het enorm druk op de frequentie van stations die allemaal door elkaar seinen of roepen en daardoor is het voor het station op die frequentie lastig om signalen uit de brei te halen. Na enige tijd (en soms al meteen) zal het station na het seinen of roepen van zijn call "UP" toevoegen. Daarmee geeft hij aan hoger te gaan luisteren dan dat hij zendt. Het voordeel daarvan is dat zijn frequentie "schoon" blijft en hij dus sneller aanroepende stations kan afwerken. En als hij dan ook nog eens wat meer bandbreedte gebruikt om te luisteren, kan hij de aanroepende stations ook nog eens spreiden. In CW betekent UP doorgaans 1kHz en in SSB 5kHz. Wat is nu jou werkwijze: je drukt op A=B waardoor je twee VFO's aan elkaar gelijk worden. Daarna druk je A/B om naar de andere VFO om te schakelen. Stem die VFO nu 1 (of 5 bij SSB) kHz hoger af, druk nu weer op A/B en vervolgens op SPL. Als je nu gaat zenden, gebruikt de transceiver door de SPL functie de andere VFO om te zenden die dus hoger afgestemd staat. Geeft het tegenstation niet jou maar een ander station antwoord, dan kan je weer op A/B drukken en proberen met je andere VFO de frequentie van het station dat gewerkt wordt te ontdekken. Vooral op lage banden hoor je vaak het tegenstation van het DX station en dat geeft je een indicatie waar hij luistert. Laat je tweede VFO daar op staan en schakel met A/B weer terug. Als hij dan "TU" (CW) of "QRZ" (SSB) geeft, kan jij meteen terugkomen op de frequentie waar hij het laatst luisterde. Dat werkt niet altijd, omdat het DX-station na elke verbinding soms zijn luisterfrequentie een stukje opschuift, juist omdat door deze truc alle stations dan ineens op zijn laatste frequentie terugkomen. Door zijn luisterfrequentie over een groter deel van de band uit te smeren, kan hij makkelijker stations werken dan dat ze precies op één frequentie terugkomen. Overigens gebruik ik de SPLIT ook wel als het tegenstation simplex werkt: ik zet mijn tweede VFO dan 100 tot 150Hz hoger (in CW). Als iedereen dan op precies de frequentie van het tegenstation terugkomt, hoort hij één aanhoudende fluittoon met daar tegenaan een signaal met een iets hogere frequentie: het mijne. En dat stelt me in staat om in een pile-up dan toch gehoord te worden. In SSB werkt dat natuurlijk niet: dan word je juist onverstaanbaar als je naast de frequentie zit. Maar voor SSB heb je weer andere functies: bijvoorbeeld een compressor of limiter. Deze functies geven je SSB signaal gemiddeld meer vermogen.

Hoe dat werkt? Normaal gesproken is het gemiddeld vermogen van een SSB signaal ongeveer 2% van het piekvermogen. Dat gemiddelde vermogen bepaalt de mate waarin het tegenstation je kan verstaan. Echter: het is heel wat anders of je lokaal een praatje zit te maken of dat je een pile-up moet doorbreken. En daar maakt een compressor het verschil. Wat die doet is de pieken van het spraaksignaal afsnijden. Normaal klinkt dat erg vervormd, maar als je dat niet te scherp doet (soft clipping) dan is het nog wel acceptabel. Het gemiddelde vermogen van het signaal neemt daardoor sterk toe, waardoor je makkelijker door een pile-up heen komt. Uiteraard kent iedereen dat trucje en als er dan ook nog gebruik gemaakt wordt van "conditieverbeteraars" van vele kiloWatten dan is het resultaat een splatter die een groot deel van de band in beslag kan nemen. Maar alles om gehoord te worden, nietwaar. Als een station zijn compressor/limiter aan heeft staan, hoor je in de spraakpauzes de klok in de woonkamer gewoon tikken. Dat is voor een lokaal QSO niet te genieten, maar alweer: in een pile-up kan het het verschil zijn tussen wel of niet gehoord worden.

Een ander belangrijk knopje is de RIT of ook wel Clarifier genoemd. RIT staat voor Receiver Incremental Tuning en verstemt de ontvangst-frequentie zonder de zendfrequentie te veranderen. Wanneer gebruik je dat? Als je tegenstation een "schuiftrompet" is. Met de tegenwoordige frequentiestabiele gesynthetiseerde frequentieopwekking hoor je het niet zo heel veel meer, maar vooral amateurs met analoge VFO-gestuurde transceivers (ex-militair of QRP-sets tijdens portable gebruik) willen nog wel eens van de frequentie lopen. Als je dan je afstemming verandert om het tegenstation te volgen en je komt retour, loop je de kans dat hij daarna gaat bijstemmen om jou te volgen en zo loop je na elke doorgang achter elkaar aan. Met de RIT kan je hem volgen zonder dat jou zend-frequentie verandert. Dat geeft je tegenstation houvast om bij jou in de buurt te blijven.

Bij gebrek aan een Split optie op je transceiver kan je daar de RIT voor gebruiken: je stemt je transceiver 1 (CW) of 5 (SSB) kHz hoger af dan het tegenstation, schakelt je RIT in en draait die omlaag tot je tegenstation zero-beat is. In feite werk je dan eveneens Split. Voorwaarde is wel dat je RIT/Clarifier een bereik van 5 kHz heeft anders heb je daar voor SSB niets aan. En sommige transceivers hebben daarvoor de XIT: de Transmitter Incremental Tuning, waarbij de ontvangstfrequentie op zijn plaats blijft en de zender verstemd wordt. Deze manieren van werken geven je dus behalve het volgen van schuiftrompetten de mogelijkheid om Split te werken met slechts één VFO.

Dan hebben sets soms een Digitale Signaal Processor (DSP). Bij de Yaesu 817/847/857/897 serie werkt deze op het laagfrequent. Daarmee kan je storende signalen "over de rand van het filter" duwen waardoor je ze niet meer hoort. Dat wil niet zeggen dat je er dan ook geen last meer van hebt. Stel je voor dat je een transceiver hebt waar een 2700Hz breed kristalfilter in zit, waarmee je naar CW signalen gaat luisteren. 1kHz boven je frequentie zit een sterk station waar je last van hebt. Door je DSP in te schakelen kan je het audio van dat station wegdraaien. Maar door het brede MF filter (voor CW althans) reageert de AGC (Automatic Gain Control ofwel Automatische Versterkings regeling) op het sterke signaal en ondanks dat je het signaal niet meer hoort, heb je er wel last van. De enige manier om dat op te lossen is een smaller MF filter. In Opa's set zit speciaal voor CW een 300Hz Inrad filter. Maar ook voor SSB zijn smallere filters te krijgen! Vooral tijdens drukke contesten helpt dat. Een DSP helpt dus wel, maar is geen ideale oplossing.

Ik noemde 'm al even: de AGC of soms AVR genoemd. Die regelt dus de versterking terug bij sterke signalen. Vaak is de AGC instelbaar of zelfs uitschakelbaar. Instelbaar betekent dan dat je de "hang"-tijd kunt instellen, meestal als Slow en Fast. Beide hebben hun voor- en tegen. Bij Slow wacht de AGC even alvorens de versterking weer op te regelen. Daarmee loop je tijdens het zoeken naar tegenstations het risico

over een zwak station heen te draaien omdat de versterking sterk teruggeregeld is. Aan de andere kant kan een snelle AVG erg onrustig klinken omdat bij een geringe spraakpauze meteen de versterking opregelt.

En waarom zou je 'm uitschakelen? Bij het werken in digitale modes. Waar je dan niet op zit te wachten is dat een of andere Luigi een vette rode streep op je waterval zet die alle andere spoortjes doet vervagen of zelfs verdwijnen. Door je AGC uit te schakelen blijven die in beeld. Is Luigi echt heel hard en gaat de boel vastlopen, regel dan de versterking terug met de RF-gain of de Attenuator (verzwakker).

Wat me brengt bij de volgende belangrijke knoppen of menu's: de verzwakkers. Ontvangen gaat lang niet altijd om versterken, maar minstens zo vaak om verzwakken. Draai bij het luisteren naar een station de RF-gain maar eens zóver terug dat de achtergrondruis begint te verdwijnen. Dat maakt het luisteren een stuk rustiger en je tegenstation veel beter te verstaan of te nemen. Iets soortgelijks zie je bij de Attenuators die soms in stappen van 10 of 20 dB ingeschakeld kunnen worden. Vooral 's-avonds hebben ontvangers het moeilijk en lang niet alleen oudere ontvangers maar ook de Icom 7300 kan het zwaar krijgen. Schakel je dan de verzwakker in, dan verdwijnt ineens een hoop rotzooi die door intermodulatie in niet-lineaire componenten veroorzaakt wordt - en dat zijn niet alleen mixers. Probeer het maar eens: het helpt echt.

Nog een andere functie die je nogal eens ziet is de Noise Blanker. Dat is een filter dat speciaal gericht is op impulsstoringen. Vroeger, toen brommers (maar soms ook auto's) rijdende vonkzenders waren, was zo'n filter inderdaad nuttig. Maar de enige impulsstoring die Opa nog wel eens hoort, is het schrikdraad als er portable tussen de weilanden gewerkt wordt. En met het enthousiasme waarmee onze regering de boerenbedrijven om zeep aan het helpen is, zal die storing binnenkort ook verleden tijd zijn. Dus die functie zal je niet vaak gebruiken.

Een laatste functie waar ik je op wil wijzen is de VOX: de Voice Operated Control Switch. Met de VOX ingeschakeld reageert de transceiver op het microfoonsignaal en gaat automatisch op zenden als de microfoon voldoende signaal oppikt. Dat kan erg handig zijn, bijvoorbeeld bij gebruik van een Bluetooth headset. Maar het kan ook tegen je werken als er veel achtergrond lawaai is zoals XYL's, kinderen, honden en/of andere lawaai producerende inwonenden.

Natuurlijk zijn er nog heel veel meer functies in de transceiver, maar dat zijn veelal Nice-to-haves en niet onmiddellijk noodzakelijk voor het dagelijks gebruik. Hopelijk heb ik je op weg geholpen om de meest gebruikte functies van een transceiver duidelijk te maken", besloot Opa. Pim knikte en zei: "Theorie is één ding, maar die functies blindelings kunnen vinden op het moment dat je ze nodig hebt is een ander verhaal". "Ja, en dat kan je oefenen. Als je een uurtje met zo'n set aan het spelen ben geweest en je hebt de functies die ik je zojuist uitgelegd een paar keer geprobeerd, dan heb je het onder de knie", zei Opa. Pim was het daar helemaal mee eens en ging met de aantekeningen die hij gemaakt had, achter Opa's set zitten om het geleerde in praktijk te brengen.

De weg naar het radio examen

Frank Molenschot PF1SCT

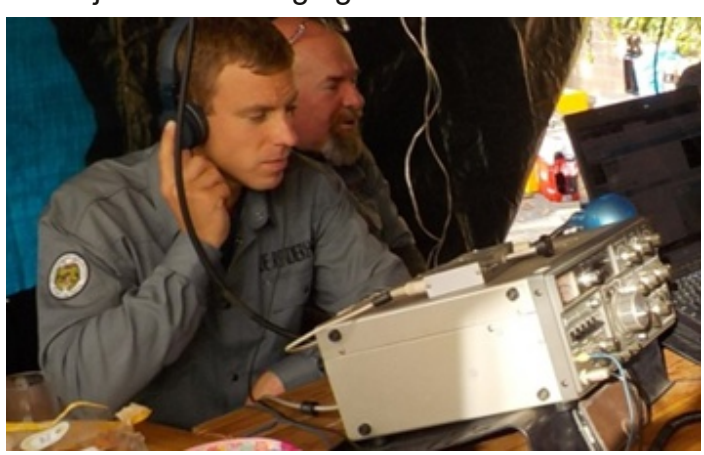
Medio 2021 werd ik (PF1SCT) door goede vriend Gerard benaderd over het behalen van de radiozend-machtiging. Dit artikel gaat over de weg naar het radio-examen, het belang van samenwerken en de rol van scouting.

Ik ben zelf via scouting (mijn vader draaide Jota en mijn leider was ook zendamateur) met de radiohobby in contact gekomen en gebleven. Als jeugdlid en als jeugdleider heb ik vele JOTA's mogen mee maken. Circa 10 jaar geleden werden mijn kinderen ook lid van scouting. Daar werd een JOTA georganiseerd, en al snel was ik weer van de partij. De zendamateur Berrie PD8B was lid van de DLZA (Digitale Leeromgeving Zend Amateurs) en spoorde mij aan om zelf te gaan leren.
Dat heb ik uiteindelijk gedaan en vanaf 2016 mocht ik zelfstandig on the air.

Ik ken Gerard ook via scouting, ik was zijn jeugdleider. Zowel Gerard als ik zijn nog actief in scouting, we beheren een scouting kampeerbos nabij Tilburg (De Rendierhoeve). En we deden de afgelopen 5 jaar met leden van het kampeerbos mee aan de JOTAJOTI.
Het kampeerterrein bestond 70 jaar in 2022, en ik heb toen extra activiteiten georganiseerd rond dat jubileum, waaronder de special call PC70RH. In het voorjaar hadden we een velddag in het kampeerbos en we deden mee aan de JOTAJOTI met een veldstation. Daar was mijn zoon, ondertussen ook jeugdleider, aanwezig met zijn welpen van scouting Berkel-Enschot om deel te nemen aan de JOTAJOTI. De cirkeltjes blijven mooi rond en kennis en enthousiasme wordt doorgegeven.

Vanaf het moment dat Gerard bij mij kwam met het idee om zijn zendmachtiging te halen was ik meteen enthousiast. Gerard heeft net als ik geen elektrotechnische achtergrond, dan moet je er toch even goed voor gaan zitten. Voor mij was het erg leuk om hem te coachen en te proberen het enthousiasme vast te houden. Gelukkig hadden we veel hulp daarbij van andere amateurs, in het bijzonder wil ik noemen PD2RST, bedankt Rob.
Hieronder het interview met Gerard, afgenomen onderweg naar het radio-examen, 18 januari 2023 te Nijkerk.

Waarom ben je begonnen met de cursus tot radio zendamateur?
Goede vraag, zegt Gerard. De oorsprong ligt bij de JOTAJOTI. Meekijken en meeluisteren. Eerst wat terughoudend en toen ik vrienden zag die weer mee deden met de JOTAJOTI een paar jaar geleden, werd ik weer enthousiast.
Ik moest wel even goed nadenken of ik niet te snel van stapel liep. Ik wil het graag goed doen dus de lat ligt bij mij al snel hoog. Je kent dat wel, je bent enthousiast koopt van alles en daarna staat het in een hoek niets te doen. Ik was echter al een paar jaar bezig met het idee om mijn zendmachtiging te behalen.



Meeluisteren JOTAJOTI 2018

Ik vind het leuk om letterlijk de verbinding te maken met andere mensen, on air, zowel in spraak maar ook digitaal zijn er leuke mogelijkheden.

Het praten is best spannend, maar op een gegeven moment heb je er een bepaald gevoel bij.

Het is een veelzijdige hobby op het gebied van communicatie. Te veel om op te noemen. Velddagen, digitaal, morsen, antennes bouwen, solderen, specialcalls, wedstrijden, satelliet, meteorologie, sterrenkunde en natuurlijk de JOTAJOTI of radioscouting in het algemeen.

Ik ben nog redelijk jong (38) voor deze hobby. Het klikt toch nog wat makkelijker met de jeugd zoals bij de JOTAJOTI waar ik dit jaar aan deel heb genomen. Het is mooi om anderen enthousiast te krijgen. Het is een alles omvattende hobby, voor ieders wat wils.

Welke cursusmethode heb je gekozen en hoe beviel dat?
Ik heb eerst het Veron boek voor Novice gekocht en gelezen. Erg interessant maar het was voor mij best ingewikkeld. Ik heb toen een boekje van PA4TON gekocht met uitgewerkte N vragen.
In het boekje staan heel veel vragen en antwoorden met uitwerkingen. Dat is er ook van F. Dat vond ik fijn. Ik heb ook mailcontact met hen gehad. In het volgende artikel staat extra informatie:
https://www.veron.nl/nieuws/examentraining-pa4ton/
Ik ben daarnaast in de DLZA leeromgeving begonnen, www.dlza.nl. Dat is modulair opgebouwd en je moet de stof doornemen en telkens afsluiten met toetsen om verder te komen. Daarnaast zijn de vragen goed uitgewerkt met uitleg.
Ik heb ook de website ham-radio.nl gebruikt daar staan ook oefenexamens in. Dat vond ik fijner om wat later in de studie te doen.

Leer je snel mensen kennen?
Ik heb veel gesproken met zendamateurs die ik al kende en een paar keer op de VERON verenigingsavond, die toevallig ook in het clubgebouw is van scouting The Challenge in Tilburg. Samen naar een verenigingsavond en al uitleg krijgen en geïntroduceerd worden met een aantal mensen is erg fijn.
Ik heb op de JOTAJOTI echt een praktijk weekend gehad. Met het opbouwen en aansluiten kom je de theorie in de praktijk tegen. Een SWR-meter, coax, connectoren, antennetuners en antennes.

JOTAJOTI 2022 Gerard actief met PC70RH

Was het studeren moeilijk?
Het is pittig je moet er veel tijd insteken, maar het zou voor de meeste mensen met de juiste inzet te doen moet zijn. Ik ben er een half jaar mee bezig geweest. Er is veel informatie te vinden en er zijn veel verschillende leermethodes.

Wat vind je tot nu toe leuk aan de hobby?
Je hebt de wereld in je binnenzak. Ik heb veel geleerd over techniek. Ik krijg veel hulp en mensen staan voor je klaar. Er heerst een beetje scouting gevoel, ons kent ons. We helpen elkaar, er zijn veel verschillende mensen binnen de hobby en daar is ook oog voor.

Wat waren de reacties toen je dit ging doen?
Mensen denken dat het suf is. Je hoort al snel mensen zeggen Breakie Breakie, en er een beetje lacherig om doen. Maar als je dan verteld en laat zien wat het is en hoe veelzijdig het is dan komen ze wel los en vinden ze het wel leuk. Ik denk dat we veel meer mensen kunnen betrekken door te laten zien wat we doen.

Wat verwacht je van het examen (vraag gesteld half uur voor het examen)?
Ik heb veel examens gemaakt. Ik zit tussen de 2 en 6 fouten. Ik zit ruim onder de 10 fouten dus ik verwacht wel dat ik slaag.

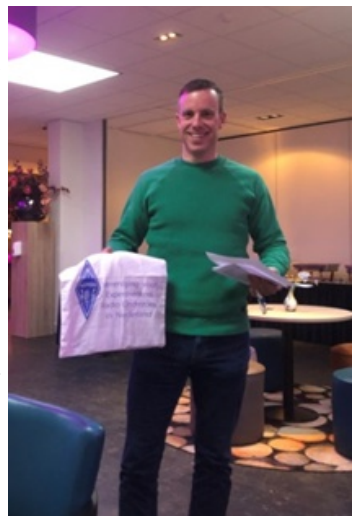
Radio zendamateur Examen

Om 15.15 begon het examen en bleef ik achter in de kantine en had een leuk gesprek met novice amateur Hans PD0HZS die zijn F-examen net had afgesloten. Hij kwam uit Alkmaar, en hij herkende zelfs mijn call (regio Tilburg) en de activiteiten rond scouting en onze QRZ pagina (websites van zendamateurs). Ik maakte nog een praatje met de examen-commissie, die er binnenkort mee stopt als het examen bij het CBR komt. Vanaf deze plek nogmaals hartelijk dank aan de Stichting Radio Examens voor de vele examens die jullie hebben verzorgd.

Geslaagd?
Gerard kwam na ongeveer 25 minuten naar buiten. Opgelucht maar toch nog even afwachten want je mag alleen de antwoorden meenemen. De vragen blijven in het lokaal. We bleven tot het einde hangen, en kregen toen het verlossende antwoord, want de antwoorden komen dan ook online. Geslaagd, met 8 foutjes, net iets meer als verwacht, maar dat zal een zorg zijn, want het papiertje is binnen!
82% van de Novice kandidaten was geslaagd. Het examen Full License dat eerder was gehouden, was blijkbaar moeilijk want daar slaagde slechts 50%

Hoe kijk je, nu je geslaagd bent terug op je cursus en examen?

Ik sprak laatst met mijn vrouw dat ik in het begin erg opkeek tegen de cursus, vooral de techniek. Hoe dichter ik echter bij het examen kwam, merkte ik dat ik ook een persoonlijke ontwikkeling doormaakte. Ik pakte makkelijker kleine klusjes op om iets te repareren en ik heb meer geduld en zoek de dingen uit als ik het niet weet.


Geslaagd! Dat lucht op

Nu zo snel mogelijk on air, want daar heb ik het voor gedaan. De meeste spullen heb ik al staan om op VHF en UHF uit te komen. Na een tijdje te hebben geluisterd kan ik straks ook mee doen.
73 en hopelijk inspireert het mensen om ook hun licentie te gaan halen en anderen om te zorgen voor een stukje coaching.

Het artikel is geschreven vanuit de Onafhankelijke Radioamateurs Brabant, de O-R-B www.o-r-b.nl
Wekelijks verzorgen we sinds 2004 vanuit Tilburg om 20.00 een gezellige radioronde op 145.400.

73 Frank PF1SCT

APRS Transceiver - Inleiding tot de software.

Robert de Kok, PA2RDK

Zoals aangekondigd in de vorige Razzies, ben ik de afgelopen weken bezig geweest met mijn nieuwe APRS VHF transceiver. Allereerst de software.

Omdat mijn kwaliteiten niet voorzien in de zelfstandige ontwikkeling van een userinterface, heb ik mij laten inspireren door de SI4735 radio van Gert PE0MGB en de layout van het scherm van mijn FT991A.

Hierbij kwam ik tot de layout zoals getoond op de volgende bladzijde.

Omdat ik de wensen en eisen nog niet allemaal wil en kan overzien, is een Agile werkwijze voor de hand liggend. Dit betekent dat de software flexibel moet worden ingericht, zodat het

eenvoudig is om er knopjes bij te bouwen en de layout en functionaliteit eenvoudig aan te passen.

Omdat de wens bestaat de radio ook middels internet te kunnen bedienen, is het noodzakelijk de functies zodanig aan te roepen dat dit niet alleen met de knopjes mogelijk is. Het programma moet zeker geen spaghetti worden, want dan loop ik binnen een paar weken hartstikke vast en wordt onderhoud en uitbreiding een drama en is de lol van programmeren snel over. Kortom het programma moet netjes modulair worden opgezet.

Nu wil ik zeker niet beweren dat ik een fantastische programmeur ben en onder druk van tijd en haast ben ook ik regelmatig in de verleiding een (te) korte route te kiezen maar ik ben wel zo arrogant te denken dat ik weet hoe het zou moeten.

Daarom wil ik dit eerste deel van de bouw van de transceiver wijden aan een aantal gebruiken en technieken en mogelijkheden met betrekking tot de ontwikkeling van de (Arduino) software in het algemeen, zodat deze onderhoudbaar, leesbaar en uit te breiden of aan te passen is voor derden. De tekst is niet bedoeld als beginnerscursus programmeren, ik ga er van uit dat je al de nodige ervaring hebt.

Voor velen zal het gesneden koek zijn, maar wellicht pik je toch er iets van op!

Voor diegenen die de transceiver willen bouwen en gewoon gebruik willen maken van het programma en helemaal geen aspiratie hebben zelf met software aan de gang te gaan, moeten nog even geduld hebben, volgende maand word ik concreter.

De bouw is nog volop in ontwikkeling maar mocht je erg nieuwgierig zijn of haast hebben, het programma en het schema in de huidige staat zijn beschikbaar op Github.

Het modulair bouwen van software vraagt eigenlijk om het object georiënteerd (=object oriented =OO) bouwen. Deze route heb ik om verschillende redenen niet gekozen, maar de principes van OO ondersteun ik wel maximaal.

De belangrijkste kenmerken zijn dat een module zelfstandig functioneert, de functie waarvoor de module is gebouwd altijd wordt uitgevoerd door deze module, er wordt dus geen code gekopieerd. Aanvullend geldt dat een module zich met één losstaande taak bezighoudt en niet afhankelijk is van andere delen van het programma.

Een goed voorbeeld is het tekenen van het scherm. Deze tekent alleen het scherm en doet geen berekeningen aan bijvoorbeeld frequenties. Een module kan wel uit sub modules bestaan, bijvoorbeeld het tekenen van het scherm bestaat onder andere uit modules om knoppen, de meter en de frequentie informatie te tekenen. Deze sub modules moeten ook zelfstandig te gebruiken zijn.

Waarom doen we dit zo: er zijn verschillende redenen om de frequentie opnieuw te tekenen, als deze handmatig wordt veranderd, als de PTT wordt ingedrukt, als er iets met APRS gebeurt etc. Als we dan steeds opnieuw de code voor het tekenen van de frequentie herhalen en we besluiten bijvoorbeeld een ander lettertype te gaan gebruiken dan dienen we dit op de verschillende plekken te doen. Het risico er een te vergeten is groot met als gevolg dat we bijvoorbeeld bij het indrukken van de PTT

opeens tegen een ander lettertype aan kijken. Dat ziet er niet uit natuurlijk!

Om deze reden kan een functie met maar één regel heel functioneel blijken: Stel dat het tekenen van de frequentie er als volgt uit ziet:

```
tft.drawString(rxFrequency, 220,90,1);
```

(Hierin is rxFrequency de te tonen frequentie, 220 de positie op x-as, 90 de positie op de y-as en 1 het fontnummer.)

Dan is het heel aantrekkelijk deze regel te kopiëren, maar als de frequentie een beetje verplaatst moet worden dan mogen we dit op verschillende plekken aanpassen.

Als we er deze functie van maken, hoeven we alleen die aan te roepen en kunnen deze op een plaats aanpassen als de frequentie een beetje verplaatst moet worden:

```
void function drawFrequency(){
  tft.drawString(rxFrequency, 220,90,1);
}
```

In de functie drawScreen kunnen we de functie drawFrequency aanroepen:

```
void function drawScreen(){
  drawFrequency();
}
```

Maar natuurlijk kunnen we drawFrequency op andere plekken ook aanroepen.

Laten we deze functie een beetje uitbreiden, stel we hebben een globale variabele (globaal wil zeggen dat de variabele overal in het programma beschikbaar is, hierover later meer.) ‘isTransmitting’. Dit is een boolean, true als er wordt gezonden en false als er wordt ontvangen. Als er wordt gezonden moet de frequentie in het rood worden getoond, bij ontvangst in het geel. Dit zou bijvoorbeeld zo kunnen:

```
void function drawFrequency(){
  If (isTransmitting) {
    tft.setTextColor(TFT_BLACK, TFT_RED);
  } else {
    tft.setTextColor(TFT_BLACK, TFT_YELLOW);
  }
  tft.drawString(rxFrequency, 220,90,1);
}
```

Dit zal gewoon werken, maar de functie (module) is nu afhankelijk van het bestaan van de variabele ‘isTransmitting’ en kan dus niet meer zelfstandig bestaan. Dit moet je daarom niet willen. Geloof mij, programmeer een poosje door en ik krijg vanzelf ergens een keer gelijk van je.

```
void function drawFrequency(bool setTXColor){
  If (setTXColor) {
    tft.setTextColor(TFT_BLACK, TFT_RED);
  } else {
    tft.setTextColor(TFT_BLACK, TFT_YELLOW);
  }
  tft.drawString(rxFrequency, 220,90,1);
}
```

Dit is een nettere oplossing, we gebruiken nu een lokale variabele ‘setTXColor’ (een lokale variabele is alleen bekend binnen de functie). De waarde van de variabele geven we mee met de aanroep van de functie, dus als volgt:

```
void function drawScreen(){
  drawFrequency(isTransmitting);
}
```

De aanroep drawFrequency(); werkt nu niet meer, we moeten nu een boolean parameter meegeven, dus drawFrequency(isTransmitting);

Technisch is dit een prima oplossing, maar het kan flexibeler. Als we de gewenste kleur meegeven in plaats van de variabele ‘isTransmitting’ kunnen we ook een derde of vierde kleur gebruiken. De functie ziet er dan als volgt uit:

```
void function drawFrequency(uint16_t btnColor){
  tft.setTextColor(TFT_BLACK, btnColor);
  tft.drawString(rxFrequency, 220,90,1);
}
```

En de aanroep van de functie kan dan de volgende zijn:

```
void function drawScreen(){
  if (isTransmitting){
    drawFrequency(TFT_RED);
  } else {
  drawFrequency(TFT_YELLOW);
  }
}
```

Even tussendoor: Nu ben ik niet per definitie een voorstander van het proberen zoveel mogelijk functionaliteit op een regel te proppen, ik weet het, het is een sport en er worden hele wedstrijden rondom georganiseerd, maar bovenstaand voorbeeld is mij toch te dol. De hele functie kan verkort worden naar:

```
void function drawScreen() {
  drawFrequency(isTransmitting?TFT_RED:TFT_YELLOW);
}
```

Veel talen ondersteunen deze schrijfwijze, het is een (dubbel) verkorte if/else.
Eigenlijk staat er if (isTransmitting==true) then TFT_RED else TFT_YELLOW.

IsTransmitting is een boolean, dus isTransmitting is al isTransmitting en hoeft dus niet te worden uitgeschreven met ==true. Andersom kan je ook op deze manier op false checken:

```
 drawFrequency(!isTransmitting?TFT_YELLOW:TFT_RED);
```

Dit doet hetzelfde en betekent:
if (isTransmitting==false) then TFT_YELLOW else TFT_RED. Het uitroepteken voor isTransmitting betekent 'not', dus staat er eigenlijk 'if not isTransmitting then TFT_YELLOW else TFT_RED'. Het vraagteken vervangt de 'if', en de : vervangt de 'else'.

Deze wijze van een if/else opschrijven kan je nesten, dus in elkaar vlechten. Stel we hebben nog een globale variabele 'openSquelch', deze is true als de squelch open staat en er dus geluid uit de speaker komt. We willen dat de frequentie groen is als de squelch open is, rood als er wordt gezonden en anders geel. Dit lukt op deze wijze:

```
drawFrequency(isTransmitting?TFT_RED:openSquelch?
TFT_GREEN:TFT_YELLOW);
```

De eerste else is nu vervangen door een tweede if/else paartje.

Verder met onze functie: Welke en het aantal parameters dat je wilt meegeven aan een functie is oneindig. Stel dat je ook het fontnummer extern wilt kunnen bepalen, dan gaat de functie er zo uitzien:

```
void function drawFrequency(uint16_t btnColor, int fontNummer){
  tft.setTextColor(TFT_BLACK, btnColor);
  tft.drawString(rxFrequency, 220,90,fontNummer);
}
```

Bij de aanroep van de functie is het nu wel verplicht het fontnummer mee te geven, laatstgenoemde voorbeeld gaat er dan als volgt uitzien:

```
drawFrequency(isTransmitting?TFT_RED:openSquelch?
TFT_GREEN:TFT_YELLOW, 1);
```

Duidelijk? Mooi.

Wel vervelend dat we nu altijd de twee parameters moeten meegeven, het gebruikte fontnummer is vrijwel altijd 1 en maar op 1 plek in het programma afwijkend. Daarvoor zijn er overloads bedacht, een functie wordt overloaded door dezelfde functie maar met minder of afwijkende parameters:

```
void function drawFrequency(uint16_t btnColor){
  drawFrequency(btnColor,1)
}
```

Deze functie kan bestaan naast onze eerste functie. De functie heeft maar 1 parameter en roept dezelfde functie met 2 parameters aan, waarbij de tweede parameter (het fontnummer altijd een 1 is. Je kunt hierbij nog verder gaan:

```
void function drawFrequency(){
  drawFrequency(TFT_YELLOW,1)
}
```

Dit roept ook onze eerste functie aan, altijd met de kleur TFT_YELLOW en font 1.
Maar ook dit kan:

```
void function drawFrequency(bool isTransmitting){
  drawFrequency(isTransmitting?TFT_RED:TFT_YELLOW,1)
}
```

We geven nu een boolean mee in plaats van een kleur en fontnummer, is die true dan wordt de frequentie rood, anders geel. Het fontnummer is altijd 1.

De compiler weet welke versie (welke overload) van een functie gebruikt moet worden. Ik maak veelvuldig gebruik van deze mogelijkheid, kijk maar eens naar de functie ‘drawButton’ in het transceiver programma.

Nog even tussendoor: over TFT_YELLOW en de andere kleuren, dit is een handig grapje in c++ (Arduino is eigenlijk gewoon C++ met een schilletje eromheen om het eenvoudiger te maken). Dit is een globale constante. Een constante wordt in Arduino gedefinieerd met #define TFT_YELLOW 0xFFE0. Hierbij is 0xFFE0 de HEX-waarde van de kleur geel. Als we nu iets geel willen kleuren kunnen we TFT_YELLOW gebruiken in plaats van 0xFFE0. Een stuk beter te lezen en onthouden toch?

Dit soort defines of constantes worden veelvuldig gebruikt en maken het lezen van het programma een stuk makkelijker. Het is een goede gewoonte om alle defines bovenin het programma te zetten.
#define kan worden gebruikt voor hex waardes, maar ook voor getallen of strings.

In plaats van #define wordt ook regelmatig const gebruikt, #DEFINE aprsFreq 144800 gedraagt zich hetzelfde als const uint32_t aprsFreq = 144800;

Technisch gaat de compiler anders om met de twee verschillende methodes, maar het resultaat is hetzelfde. In beide gevallen hebben we in het programma een globale variabele aprsFreq beschikbaar met de waarde 144800. De inhoud van de variabele kan niet worden gewijzigd, is constant.

We hebben nu variabelen aan een functie meegegeven, andersom kan een functie ook gegevens teruggeven. In bovenstaande voorbeelden beginnen alle functies met ‘void’, dit betekent leeg, er komt geen waarde terug van de functie. Een functie een waarde laten teruggeven kan op deze wijze:

```
int myIntFunction(){
  return 3;
}

String myStringFunction(){
  return “Text”;
}

bool myBoolFunction(){
  return true;
}
```

In plaats van void begint de functiedefinitie met het type variabele dat wordt teruggegeven. Dus in bovenstaande voorbeelden een integer, een string en een boolean. De functie MOET een waarde teruggeven, anders wil het programma niet compileren.
Dit gaat dus fout:

```
bool myBoolFunction(){
  Bool test = true;
  If (test) return true;
}
```

Wist je trouwens dat if (test) hetzelfde betekent als if (test==true)?

If (!test) betekent if (test==false)!
Er wordt alleen een waarde teruggegeven als test true is. De compiler gaat hierop onderuit.
Ondanks dat test hard op true is gezet. Op deze manier werkt het wel:

```
bool myBoolFunction(){
  Bool returnValue = false;
  Bool test = true;
  If (test) returnValue = true;
  return returnValue;
}
```

Spreekt voor zich denk ik. Het woord ‘return’ wordt altijd gevolgd door de returnwaarde en breekt de functie af. Het mag dus ook zo:

```
bool myBoolFunction(){
  Bool test = true;
  If (test) return true;
  return false;
}
```

Doet precies hetzelfde maar is (vind ik) slechter leesbaar. Ik ben er een voorstander van de functie altijd op de laatste regel te laten eindigen met een return waarvan de waarde ergens in de functie wordt gezet. Maar dit is persoonlijk.

Een voorbeeld: stel we willen het bepalen van de kleur van de frequentie in een functie stoppen:

```
uint16_t getFrequencyColor(bool isTransmitting, bool openSquelch) {
  uint16_t returnColor = TFT_WHITE;
       returnColor = isTransmitting?TFT_RED:openSquelch?TFT_GREEN:TFT_YELLOW;
  return returnColor;
}
```

De aanroep wordt dan:

```
drawFrequency(getFrequencyColor(isTransmitting, openSquelch));
```

Vaak wordt een returnvalue gebruikt om aan te geven of een functie gelukt is:

```
bool setFrequency(){
  returnValue = false;
  set de frequentie;
  if (frequentie klopt) returnValue = true;
  return returnValue;
}
```

Waarom doen we dit allemaal zo vraag je je wellicht af. Er zijn verschillende redenen voor, allereerst het geheugen in een ESP32 of willekeurig welke microprocessor is niet oneindig. We dienen hier met zorg mee om te gaan. Door een variabele alleen in een functie te definiëren en gebruiken, bestaat deze variabele alleen in de functie. Idealiter zitten er in het geheugen alleen de variabelen die we nodig hebben. We kunnen het geheugen dus voor verschillende variabelen gebruiken. Ten tweede, de code wordt een stuk beter leesbaar en zoals aan het begin van dit verhaal al aangehaald, de functies kunnen zelfstandig en autonoom functioneren. Dit op zich hoeft geen doel te zijn, maar het maakt een functie wel veel beter herbruikbaar.

We hebben het continu over variabelen en natuurlijk weten we allemaal dat er verschillende soorten variabelen bestaan. Grofweg zijn er nummers, booleans en strings. Maar er valt een hoop meer over te vertellen.

Allereerst de numerieke variabelen, Arduino kent onder andere int, int8_t, uint8_t, int16_t, int32_t en float. Er zijn er nog meer. Maar wat betekent het:

int8_t is een integer (een geheel getal) van maximaal 8 bits. Met 8 bits kan je 255 waardes definiëren, dus kan een int8_t een getal tussen -127 en +127 bevatten. Je snapt het al, een int16_t bestaat uit 16 bits en kan dus een getal tussen -32767 en +32767 bevatten. Als je geen negatieve getallen nodig hebt kun je overwegen de uint8_t of uint16_t types gebruiken. De u staat voor unsigned. Een 8bits uint kan daarom een getal tussen 0 en 255 bevatten en een 16 bits tussen 0 en 65535. Arduino is heel eenvoudig in de afhandeling van integers

hetgeen best leuke bugs kan opleveren. Als je stelt uint8_t x = 256, dan vindt de compiler dat x=0, na 255 krijg je weer 0, 257 is dus 1, 258 is 2 en zo verder, tot 511, dat is 255 en 512 is weer 0!

Een int in de Arduino omgeving is per definitie een int16_t, dus 2 bytes groot. Als het nummer in de variabele nooit groter wordt dan 127, gebruikt de variabele dus 2 keer zoveel geheugen dan noodzakelijk, zonde, gebruik dan bij voorkeur een int8_t!

Als je geen negatieve waardes nodig hebt, gebruik dan een uint. Technisch maakt het meestal niet uit, maar het komt de leesbaarheid ten goede.

Een float bestaat uit 4 bytes en kan een waarde tussen -3.4028235E+38 en 3.4028235E+38 bevatten.

Een boolean kan alleen ja/nee of true/false of 1/0 bevatten en is dus maar 1 bit groot. Als je aan een boolean genoeg hebt, gebruik dan ook een boolean want dat is het meest economisch.

Elke integer kan ook als boolean worden gebruikt in de Arduino omgeving, maar realiseer je dat je tenminste 7 bits geheugen weggooit.

Een string bevat een tekst met een willekeurige lengte. Hier kom ik nog op terug.

We komen ook de byte en char tegen. Technisch zijn het, net als de int8_t en uint8_t beiden variabelen

We kunnen ook een eigen soort variabele maken. Dit is een ‘struct’. Een struct is een vorm van een variabele waarin we meerdere eigenschappen van die variabele vastleggen. In de transceiver maak ik hier veelvuldig gebruik van.
Een voorbeeld hiervan is de struct ‘Button’:

```
typedef struct // Buttons
{
   const char *name;     // Buttonname
   const char *caption;  // Buttoncaption
   char waarde[12];      // Buttontext
   uint16_t pageNo;
   uint16_t xPos;
   uint16_t yPos;
   uint16_t width;
   uint16_t height;
   uint16_t btnColor;
   uint16_t bckColor;
} Button;
```

Een variabele van het type ‘Button’ bestaat dus uit 10 variabelen (name, caption, waarde, pageNo etc..). Een ander voorbeeld is de struct ‘Sfreq’:

```
typedef struct  // Frequency parts
{
  int fMHZ;
  int fKHz;
} SFreq;
```

Zoals we hierboven hebben gezien, een functie kan maar een variabele teruggeven. Maar die variabele kan prima een struct zijn. Op deze manier kunnen we dus meerdere variabelen teruggeven vanuit een functie.

In het transceiver programma maak ik voor het rekenwerk aan de frequentie gebruik van kanalen (channels) in plaats van de frequentie. Channel 0 komt overeen met 144.000MHz, channel 160 is 146.000MHz, channel 22880 is 430.000MHz, 23680 is 440.000MHz. Inderdaad, een 12.5 KHz raster, tellend vanaf 144.000MHz.

In het programma heb ik een functie om het kanaal om te rekenen naar een frequentie, maar voor het tekenen van de frequentie is het makkelijk om de MHz’en en KHz’en als losse variabelen terug te krijgen. Dat doe ik met de volgende functie:

```
SFreq getFreq(int channel){
  int fMHz = floor(channel/80) + 144;
  int fKHz = (channel - (floor(channel/80)*80))*125;
  SFreq sFreq = {fMHz,fKHz};
  return sFreq;
}
```

Weer even tussendoor: Wat gebeurt er hier: er zitten 80 stappen van 12.5 KHz in een megaherz. Dus in fMHz stoppen we channel/80 zonder het deel achter de komma en tellen hier 144 bij op.

*In fKHZ stoppen we channel – (fMHz*80) en vermenigvuldigen dit met 125.*
Bijvoorbeeld channel 100: 100/80 = 1 en rest 20. fMHz wordt dus 1+144 = 145. fKHz wordt 100 – (1 80) = 20, 20 *125 = 2500, dus fKHz = 2500.*

fKHZ is eigenlijk in stappen van 0,1 kHz, dus 2500 betekent 250,0

Als we de transceiver willen ombouwen naar een 5KHz raster, hoeven we dus alleen hier een aanpassing te maken, Dat is nog eens efficiënt programmeren!

In deze functie komt alles samen, de ingaande parameter is het kanaal dat we willen omrekenen, we retourneren een zelf gedefinieerde variabele van het type SFreq. SFreq bestaat weer uit 2 variabelen fMHz en fKHz.

Een voorbeeld van de werking:

```
SFreq sFreq = getFreq(140);
Serial.printf("%01d.%04d",sFreq.fMHz,sFreq.fKHz);
```

Er wordt een variabele sFreq van het type (struct) SFreq gemaakt en gevuld met de waardes van channel ‘rxChannel’. In sFreq zijn 2 variabelen beschikbaar, sFreq.fMHz en sFreq.fKHz.
Dit resulteert in de tekst 145.7500.

Weer een stukje tussendoor: De functie printf is een heel handige om variabelen in een tekst te kunnen zetten. In bijvoorbeeld Serial.printf(“De frequentie is %01dMHz en %04dKHz, dit is in de %s band",sFreq.fMHz,sFreq.fKHz,”twee meter”);

Op de plaats ‘%01d’ komt de integer fMHz te staan, de 01 wil zeggen dat er minimaal 1 positie wordt gevuld en de d wil zeggen een getal. ‘%04d’ is een getal met minimaal 4 posities. 25 wordt zodoende 0025. Hier komt fKHz te staan. ‘%s’ wordt vervangen door de string ‘twee meter’.

Deze werkwijze kan ook worden gebruikt om een buffer te vullen:

Sprintf((buf, “De frequentie is %01dMHz en %04dKHz, dit is in de %s band", sFreq.fMHz, sFreq.fKHz,”twee meter”);
Serial.print(buf);

De buffer ‘buf’ wordt gevuld en op de tweede regel geprint.

Ik wil het hebben over array’s. Een array is een lijst van variabelen. Een array kan worden gemaakt van alle variabele types, dus ook van structs.

Voorbeelden van arrays:

```
int myInts[6];
int myInts[] = {2, 4, 8, 3, 6};
int myInts[5] = {2, 4, -8, 3, 2};
char message[6] = "hello";
SFreq sFreqs[4];
```

De eerste is een array van 6 int’s waarvan de waardes niet zijn vastgelegd.
De tweede en derde zijn allebei array’s met 5 vastgelegde int’s. De schrijfwijze van de derde is overdreven, technisch mag het wel maar het is verwarrend.
De vierde is een array van 6 chars (bytes die worden geïnterpreteerd als ASCII karakters).
De vijfde is een array van 4 SFreq’s, deze zijn alle vier nog leeg.
Een element van een array kan worden opgevraagd of gezet met zijn index: van de 2e en 3e array is myInts[0]=2, myInts[1]=4 etc. Elementen

in een array tellen dus vanaf 0, zijn zerobased.

```
int myInts[4] = {2, 4, -8, 3, 2};
int myInts[6] = {2, 4, -8, 3, 2};
```

De eerste regel gaat fout met de melding:
error: too many initializers for 'int [4]', een array met 5 waardes en er zijn maar 4 plekken gereserveerd
De tweede regel is spannend, de compiler vindt het goed, maar zijn er nu 5 of 6 plekken?

char message[6] = "hello"; heeft maar 5 letters maar toch is de array vol. Een array met characters (bytes die een teken vertegenwoordigen) wordt altijd afgesloten met een character null. Dit is geen 0, wat dat is ASCII 48. Kijk maar eens hier. Ik kom hier later nog op terug.

In het transceiver programma maak ik veel gebruik van array's. Een belangrijke is de array met buttons, dit is een array van structs van het type Button. Maar er is ook een array met repeaters en een array met CTCSS codes in gebruik.

In arrays wil je over het algemeen zoeken naar een element of door de verschillende elementen lopen. Een goed voorbeeld hiervan is de functie drawButtons(), deze tekent (onder voorwaarden) alle knoppen uit de array met buttons. Met een for loop kun je door alle items in de array met buttons lopen:

```
for (int i=0; i<aantal buttons; i++) {//etc..}
```

Het venijn zit in 'aantal buttons', dit moet je wel kunnen achterhalen. Nu is er een functie sizeof() beschikbaar. Het had leuk geweest als sizeof() het aantal elementen in het array had verteld, maar dat doet het niet. sizeof() vertelt de grootte van de variabele in bytes. Met sizeof(buttons) blijkt dus de grootte van de hele array buttons. Maar er is een leuke oplossing: sizeof(button[0]) geeft de grootte aan van de variabele button[0] in de array buttons. Dit mag natuurlijk ook button[1] of button[12] zijn, tenminste als die bestaan in de array.

Sizeof(buttons)/sizeof(button[0]) geeft dus het aantal elementen aan in de array buttons.
De for loop ziet er dus zo uit:

```
for (int i=0; i<sizeof(buttons)/sizeof(buttons[0]); i++) {//etc…}
```

Het zoeken in een array gebeurt niet anders, loop door de array totdat je het juiste item vindt:

```
Button findButtonByName(String name){
 for (int i=0; i<sizeof(buttons)/sizeof(buttons[0]); i++) {
                    if (String(buttons[i].name)==name) return
findButtonInfo(buttons[i]);
 }
 return buttons[0];
}
```

Spreekt voor zich hoop ik. Hierbij blijkt meteen hoe fijn het is dat een functie een struct als returnwaarde kan teruggeven. Met de aanroep:

```
Button mijnKnop = findButtonByName("Scan");
```

hebben we alle gegevens van de button "Scan" te pakken in de variabele mijnKnop gestopt.. Een echte functie om te zoeken in een array is er niet, maar kan je natuurlijk wel zelf maken.

Ik had het er net al over: char message[6] = "hello"; Dit is een array van characters (lettertekens). String message = "hello"; lijkt en is ook vrijwel hetzelfde. Maar niet helemaal. In de Arduino omgeving kan gebruik gemaakt worden van strings, maar C++ kan dit standaard niet. In C++ is een string altijd een array van characters afgesloten met een char(null). Het werken met char arrays is een beetje lastiger dan het werken met strings, maar ik wil er toch voor pleiten dat je probeert eraan te wennen. Daarvoor heb ik een goede reden, De compiler gaat jou proberen te helpen met het gebruik van strings, maar deze zijn voor de compiler niet ideaal. Vooral niet omdat de lengte niet is gedefinieerd en de char(null) ontbreekt waarmee heel mooi het einde van een string bepaald kan worden. Het voorbeeld is technisch niet helemaal correct maar geeft een beeld van wat er gebeurt in het geheugen van

een processor bij gebruik van een string. Stel drie variabelen:

```
Int i=5;
String s=”Oliebol”;
Char c=’X’.
```

In het geheugens staat nu 5OliebolX.
We veranderen nu de string in “Appelflap”. Dit past in het geheugen niet op de plek van “Oliebol”, hiervoor zijn 2 extra posities noodzakelijk. Het geheugen gaat er dan zo uitzien:5_____XAppelflap. Voor string s is een nieuw plekje gevonden en in het geheugen is een gat ontstaan. Het opvullen van dit gat met nieuwe variabelen is voor de processor lastig.
Hadden we dit gedaan:

```
Int i=5;
Char s[20]=”Oliebol”;
Char c=’X’.
```

Dan had het geheugen er zo uitgezien: 5Oliebol0_____X. En na het veranderen 5Appelflap0_____X. Het geheugen hoeft dan niet te groeien en er hoeven geen variabelen te worden verplaatst. Neem dit serieus als je wilt dat jouw programma 24/7 blijft doordraaien. Het manipuleren van strings leidt vrijwel altijd tot regelmatig (1 keer per week is ook regelmatig) crashen, omdat het geheugen vol zit door losse stukjes string.

Als er belangstelling voor blijkt wil ik nog wel eens verder uitweiden over deze materie.

Als laatste deze maand wil ik het hebben over de naamgeving van variabelen. Je doet jezelf een lol als je hiermee probeert consequent te zijn, dit komt de leesbaarheid van de code ten goede. Bezuinig niet op de lengte, dat kost echt geen geheugen.
De variabelen rf en tf zeggen niemand iets, rxFrequency en txFrequency zijn meteen duidelijk. De tijd die het extra typewerk kost, verdien je ruimschoots terug!

Om dezelfde reden, herkenbaarheid van de code, is de schrijfwijze ook belangrijk: we zien vaak camelCase, PascalCase en Snake_Case voorbij komen.

camelCase, dus beginnen met een kleine letter, alles aan elkaar en elk nieuw woord met een hoofdletter, gebruik je voor variabelen. Een variabele begint dus altijd met een kleine letter.

PascalCase, dus beginnen met een hoofdletter, alles aan elkaar en elk nieuw woord met een hoofdletter, gebruik je voor functies en structs. Tenminste zou het moet en zijn vind ik. In Arduino gaat dit al mis omdat de functies loop() en setup() beiden met een kleine letter beginnen.

CAPITALS, dus alles hoofdletters, gebruik je voor defines.

Snake_Case is uit de mode, teveel onzinnige underscores.

Soms kom je nog wel eens de Hungarian notation tegen, daarbij geeft de eerste letter of letters aan wat voor soort variabele het betreft, dus strName voor string, iName voor integer, bName voor boolean, fName voor functie. Een beetje achterhaald, maar soms wel lekker duidelijk.

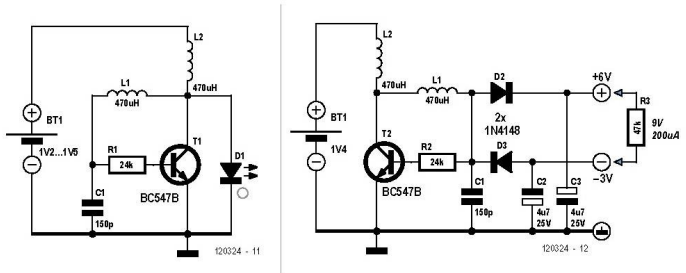
Ik weet het, ik ben absoluut te betrappen op fouten wat betreft naamgeving, maar ik doe mijn best.

Volgende maand gaan we wat dieper in op het programma zelf. Dan wordt het vast interessanter.

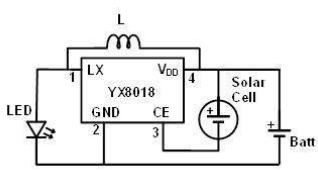
PA3CNO's Blog

Afgelopen maand hoorde ik tijdens het vaste avondrondje Henny PA3HK bezig met de reparatie van een oude versterker waarvan de transistoren moeilijk te verkrijgen waren. Ik heb dat probleem zelf ook wel eens gehad: een apparaat dat onderdelen bevat die defect zijn gegaan maar die bij de reguliere onderdelenleveranciers op internet niet meer te krijgen zijn. Iedereen kent natuurlijk Conrad, maar die is zijn assortiment aan het beperken, richt zich steeds meer op SMD componenten en aan HF doen ze sowieso niet veel. In Duitsland hebben we Reichelt, die vaak wat goedkoper is en ook meer spullen heeft voor de experimenterende HF amateur. Ga je meer de professionele kant op, dan is daar Mouser die heel veel heeft, maar een stuk duurder is en ook nogal wat verzendkosten rekent. Wat minder bekend is Farnell, die nog meer op de professionele markt gericht is. Maar een bedrijf wat bijna niemand kent, is Donberg.ie. Deze site levert de meest onmogelijke onderdelen. Als ze het niet hebben, is het nooit gemaakt. Ze zijn stervensduur, maar als je geen keus hebt is het wel de enige mogelijkheid om aan historische onderdelen te komen. Misschien een tip. Zet 'm in je bookmarks voor het geval je een keer tegen kapotte onderdelen aanloopt die je nergens meer kunt vinden.

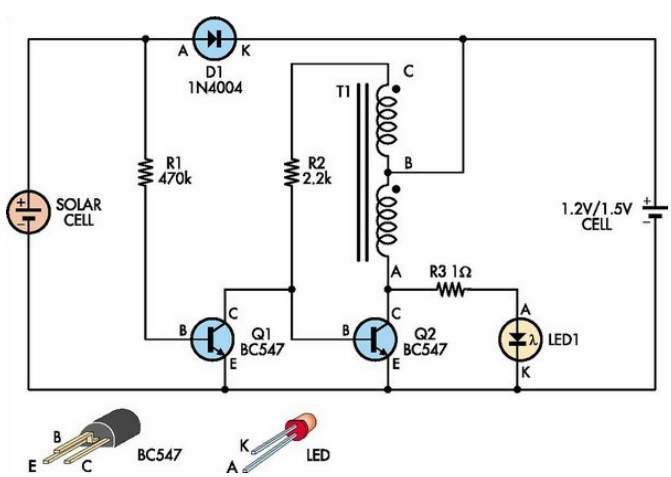
In de Elektuur nieuwsbrief stond een stukje over een één-transistor spanningsomvormer zoals die toegepast wordt in van die tuinverlichting met zonnepaneeltje. Daar zit vaak maar één AA oplaadbare batterij in en die zijn nominaal 1,2V: een spanning die te laag is om zelfs maar een rode LED (die de laagste doorlaat spanning heeft) op te laten branden. Zo'n omvormertje zorgt er dan voor dat de LED blijft branden tot de batterijspanning tot zo'n 0,8V gedaald is. De auteur van het artikel beschrijft hoe hij de schakeling ombouwde zodat er 9V bij 200uA uit te halen is. Geen idee wat je daar mee moet, maar misschien heb je er een toepassing voor.



Wat me intrigeerde, is hoe die zonnecel dan zodanig aangesloten wordt dat de batterij geladen wordt en het geheel automatisch inschakelt als het donker wordt. Enig speurwerk op internet liet zien dat in de meeste gevallen gewoon een chip toegepast wordt. Dat heb ik ook gezien bij de reparatie van mijn tuinlampjes, want hoe goed ze die dingen ook zeggen te maken, uiteindelijk lopen ze vol water, rotten de LED's van de print en geven de historische NiCd batterijen het op. Uiteindelijk vond ik een schema met transistoren, maar niet met één transistor, zie de schakeling hieronder.



Als de zonnecel spanning levert, is Q1 geopend en die neemt de basisspanning van Q2 weg. De accu wordt dan geladen via D1. Maar dan brandt de LED toch via T1? Nee, want daar is de accuspanning te laag voor zoals ik al schreef. Wordt het donker, dan gaat Q1 dicht, Q2 kan daardoor oscilleren en er wordt voldoende spanning opgewekt om de LED te laten branden. Ingenieus niet?



Heb je recent nog op aprs.fi gekeken? Is je wat opgevallen? De site aprs.fi is afgestapt van het gebruik van Google Maps. Al in 2018 sloeg de sitebeheerder alarm omdat met het gehanteerde billing beleid de rekening voor het gebruik op zou kunnen lopen tot €4000-5000 per maand. En nu is de bom kennelijk gebarsten: aprs.fi is overgestapt op het gebruik van OpenStreetMap. Sommige dingen zullen dus anders zijn, maar het belangrijkste werkt nog: kunnen zien waar stations zich bevinden. Mooi toch?

Afdelingsnieuws

Verslagen zijn we door het bericht dat op 24 januari j.l. Anneke van Strien, PD4EJP, XYL van Paul PA3DFR, overleden is. We wisten dat Anneke al een tijd ziek was, maar haar overlijden kwam toch nog onverwacht snel. We wensen Paul en zijn familie veel sterkte toe met het verwerken van dit verlies.

Op woensdag 11 januari hadden we dan onze eerste bijeenkomst van het nieuwe jaar en ook de eerste die in ons nieuwe onderkomen gehouden werd: buurthuis 't Span aan de Sullivanlijn 31 in Zoetermeer. De ruimte is een stuk luxer dan ons oude onderkomen van de Minigolfclub: er is een bar waar we niet zelf meer achter hoeven te staan en die voldoende aanbod heeft voor onze wensen. We zijn er dus blij mee, zie de foto rechtsonder.

Voor de maand februari zijn de clubavonden op de woensdagen 8 en 22, waarbij op de 8e de QSL-manager weer aanwezig zal zijn voor het uitwisselen van de kaarten. Vanaf 20:00 kan iedereen dan weer bij ons terecht op de nieuwe locatie: Sullivanlijn 31, 2728BR Zoetermeer. Het buurthuis ligt enigszins van de weg, aan het Aldo van Eyckpark. Voor nieuwkomers is het soms even zoeken, maar als je langs de (momenteel in verbouwing zijnde) snackbar loopt, dan ligt het aan je linkerhand.

Nog even over het weerstation project: dat is nog in ontwikkeling. Het blijkt dat de problemen waar nabouwers tegenaan lopen dusdanig groot zijn, dat we het over een andere boeg gaan gooien. Hardwarematig is het allemaal heel eenvoudig, maar de software er op de juiste manier inkrijgen is een hersenkraker. Daarom is onze softwaredokter nu bezig om het weerstation te voorzien van een webpagina waarop je de configuratie kunt aanpassen. Als dat werkt, kunnen we de ESP32's geprogrammeerd uitleveren en de problemen beperken. Ook de functionaliteit is uitgebreid: je ziet nu ook de bandcondities en de MUF zoals deze door de ionosonde in Dourbes gerapporteerd wordt.