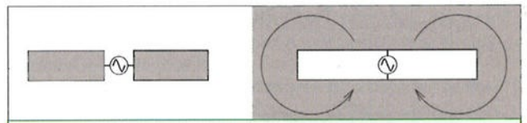
publiceerd worden [3]. Het verband is als volgt. Een antenne die bestaat uit gaten in een metalen plaat, heeft precies hetzelfde stralingsdiagram als eentje die bestaat juist uit metaal op de plekken van die gaten en vice versa, de z.g. complementaire antenne. Daarbij moet dan wel het volgende in acht worden genomen:
(i) Het elektrisch en het magnetisch veld worden verwisseld; praktisch gezien betekent dit dat de polarisatie 90 graden draait.
(ii) Het voedingspunt wordt 90 graden gedraaid; dat moet wel, want als je metaal door lucht vervangt en vice versa, zou je voedingslijn anders in de lucht eindigen.
(iii) De impedantie wordt omgekeerd: als je begint met een antenne met lage impedantie, dan heeft de complementaire antenne juist een hoge impedantie, en omgekeerd. Net als punt (i) heeft dit ermee te maken dat spanning en stroom a.h.w. verwisseld worden.

Als voorbeeld toont figuur 6 links een gewone halvegolf-dipoolantenne, bestaande uit een metalen strip die halverwege is onderbroken om de zender aan te sluiten. Rechts staat z'n complement: een rechthoekig gat in een oneindig grote metalen plaat, met halverwege de zender aangesloten over de breedte van de sleuf. Punt (ii) is meteen duidelijk: de aansluiting voor de zender is inderdaad 90 graden gedraaid.



Figuur 6 | Halvegolfdipool en z'n complement, een halve golf lange sleuf

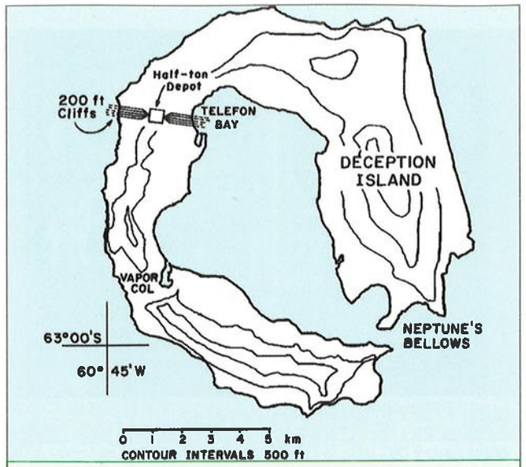
Punt (i) betekent dat terwijl de dipool horizontaal gepolariseerd is, de horizontale sleuf verticaal gepolariseerd werkt. Dat is niet zo gek: de spanning van de zender wordt nu immers verticaal aangeboden. De verwisseling van de rol van het elektrische en het magnetische veld kunnen we ook op andere wijze inzien. In de dipool beweegt zich elektrische lading heen en weer; de linkerhelft is negatief en de rechterhelft positief geladen, en een halve periode later omgekeerd. In de sleufantenne lopen stromen door het metaal, zoals aangegeven door de pijlen: links tegen de klok in en rechts met de klok mee (en een halve periode later natuurlijk omgekeerd). Zo krijgen we links een magnetische noordpool naar ons toe wijzend, en rechts een magnetische zuidpool: de tegenhangers van positieve en negatieve lading op de dipool.

Punt (iii) ten slotte betekent dat terwijl de halvegolf-dipool een lage impedantie heeft, bijv. 67 ohm als de strip 1/200 λ breed is, de sleufstraler juist een vrij hoge impedantie heeft, nl. 530 ohm bij deze zelfde afmetingen. Het product van deze getallen is (afgezien van afronding) 35476 Ω^2. Volgens Booker's berekening is dat altijd zo als je de impedanties van een antenne en z'n complement vermenigvuldigt, ongeacht de vorm en afmetingen van de antenne. Wil je een sleufstraler met een lage impedantie, dan moet ie ongeveer een hele golf lang zijn: 0,925 λ lang en 0,066 λ breed resulteert in precies 50 ohm (getallen uit [4]).

Eiland als antenne

Niet alleen amateurs verzinnen soms gekke dingen, professionals kunnen er ook wat van. Zo is in 1960 prof. Millett Morgan (die overigens ook radioamateur W1HDA was), met het voorstel gekomen om een heel eiland als zendantenne te gebruiken om propagatie van VLF-signalen te onderzoeken [5]. Hij had daarvoor Deception Island, vlakbij Antarctica, op het oog.

Figuur 7 toont een kaart van Deception Island, dat bestaat uit een ringvormig gebergte met in het midden een lagune, die op één plek verbinding heeft met de zee. Op de kaart linksboven is het voedingspunt: vanaf een plek midden op het land worden draden getrokken naar het water aan de 'buitenkant' van het eiland, en naar Telefon Bay in de lagune. Vervolgens zou de stroom door het zeewater links en rechts om het eiland moeten lopen.



Figuur 7 | Deception Island (uit [6])

Aangezien het zeewater een geleidend vlak vormt en het eiland daarin een isolator is, zou dit een natuurlijke sleufantenne vormen. Gezien Booker's theorie zou het stralingsdiagram vergelijkbaar moeten zijn met dat van een rondgebogen horizontale dipool, maar dan verticaal gepolariseerd.

In 1961 zijn daadwerkelijk impedantiemetingen op Deception Island gedaan [6], maar de meetresultaten klopten niet met de verwachtingen. Er zijn meer experimenten gedaan met landtongen in zee als antenne [7]. De conclusie lijkt te zijn dat land simpelweg niet voldoende isoleert, zodat de stroom gewoon onder de voedingslijn terugloopt in plaats van de gewenste omweg via het zeewater te nemen.

Overigens is de naam Telefon Bay in deze context natuurlijk wel opvallend. Deze baai blijkt te zijn genoemd naar een Noors schip dat daar begin vorige eeuw gerepareerd is. Dat schip had ook nog een zusterschip genaamd 'Telegraf'. Met zulke namen zou je wellicht denken dat het kabellegschepen waren, maar nee, ze werden ingezet bij de walvisvaart.

Referenties

[1] https://w140.com/tekwiki/wiki/TU-75B
[2] http://abelian.org/vlf/amateur-radio
[3] H.G. Booker: Slot aerials and their relation to complementary wire aerials (Babinet's principle). *Journal of the IEE* part IIIA, 1946.
[4] J.D. Kraus: Antennas. 1950
[5] M.G. Morgan: An Island as a Natural Very-Low-Frequency Transmitting Antenna. *IRE Tr. on antennas and propagation*, sept. 1960.
[6] M.G. Morgan: Impedance measurements of Deception Island as a natural very-low-frequency antenna. *Radio Science*, feb. 1979.
[7] N.K. Uzunoglu, S.J. Kouridakis: Radiation of Very Low and Extremely Low Frequencies (VLF & ELF) by a Natural Antenna Based on an Island or a Peninsula Structure. *Radio Science Bulletin*, maart 2004.
[8] Markus Vester DF6NM via VLF@groups.io, 1 sept. 2021. https://groups.io/g/VLF/topic/sounding/85056084
[9] Technische notities van PA3FWM, *Electron* 5/2018. ■

Marcel Opbroek PA3MRO
pa3mro@veron.nl

WSPR van A tot Z

De technologie van foutcorrectie bij WSPR-transmissies

Inleiding

In 2015 zag ik op de dag van de radioamateur in Apeldoorn een presentatie over WSPR. WSPR staat voor Weak Signal Propagation Reporter. Daar werd uitgelegd wat WSPR was en hoe je met weinig vermogen (maximaal enkele Watts.) enorme afstanden kon afleggen. Dit alles met een zelfbouwpakket. Het pakket besteld en gebouwd, en experimenteren maar. Binnen de kortste keren werd ik met twee watt gehoord in Australië. Fantastisch! Dat heeft mij nieuwsgierig gemaakt naar de werking.

Na wat zoeken op internet kwam ik diverse artikelen tegen, waarin wordt beschreven hoe je een WSPR-signaal maakt. Duidelijke verhalen, maar er ontbrak nog iets, namelijk: wat gebeurt er aan de ontvangende kant?

In dit artikel leg ik uit wat er allemaal komt kijken bij het coderen en decoderen van WSPR-signalen. Ik heb veel artikelen bekeken en ben in de WSJT-X software gedoken om te kijken wat er allemaal bij een WSPR-transmissie komt kijken. Delen van de algoritmes heb ik nagebouwd in python om te onderzoeken wat er precies gebeurt. In dit artikel neem ik jullie mee van begin tot eind van een WSPR-transmissie.

Om het uit te leggen herhaal ik een aantal belangrijke onderwerpen uit de communicatietheorie.

Algemeen communicatiemodel

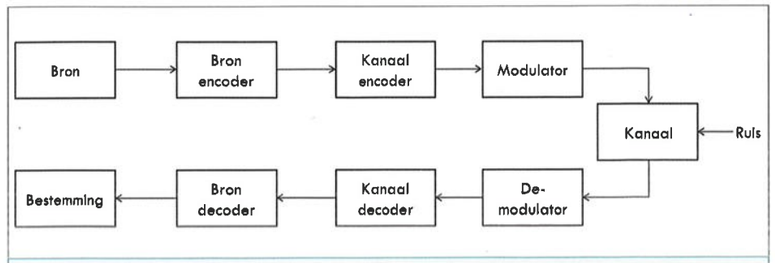
In figuur 1 is een algemeen communicatiemodel weergegeven. Aan de hand van dit model leg ik WSPR uit. De bedoeling is informatie van de bron naar de bestemming over te brengen met zó weinig fouten dat het bericht betekenis heeft voor de ontvanger. Het model begint bij de bron; deze bevat de informatie die verstuurd moet worden.

De broninformatie wordt eerst gecodeerd in de bronencoder. Hier wordt alle informatie zo compact mogelijk vertaald naar 0'en en 1'en. Bij WSPR is die informatie het callsign, de locator en het uitgezonden vermogen (in dBm). Het nettobericht (of payload) van het WSPR-bericht is 50 bits.

De kanaalencoder zet de broninformatie om naar een kanaalcode. De kanaalcode zorgt ervoor dat de informatie aan de ontvangende kant foutloos (binnen grenzen) kan worden gedecodeerd. Dit wordt 'forward error correction' (FEC) genoemd. Hier zijn verschillende technieken voor. Bij WSPR wordt een convolutiecode gebruikt, JT65 gebruikt Reed Solomon codes, en FT8 en FT4 gebruiken Low Density Parity Codes (LDPC-codes). De kanaalencoder zet 50 bits WSPR payload om naar een 162-bits convolutiecode.

Om te zorgen dat de datastroom bestand is tegen burstfouten (meerdere fouten achter elkaar door bijvoorbeeld fading of storing (QRM/QRN)) worden de te versturen bits in een willekeurig patroon gezet; deze techniek heet interleaving. Er wordt als laatste een synchronisatievector toegevoegd.

De modulator zet de bits om naar een radiosignaal. Bij WSPR worden er vier frequenties verstuurd (4-MFSK). Deze frequenties liggen 1,46 Hz uit elkaar, wat een totale bandbreedte van het WSPR-signaal geeft van ongeveer 6 Hz. De totale bitlengte is 0,683 s, wat met 162 bits zorgt voor een totale transmissietijd van 110,6 s.
Tijdens de propagatie van het signaal wordt het signaal beïnvloed door ruis, fading, overlappende transmissies en QRM. De zender en ontvanger kunnen frequentiedrift van het signaal veroorzaken.
In de ontvanger wordt het ontvangen signaal gedemoduleerd. De bits worden in de oorspronkelijke volgorde gezet (deinterleave), en daarna wordt foutcorrectie toegepast. Na deze stap wordt de payload gedecodeerd om call, locator en grid zichtbaar te maken.



Figuur 1 | Algemeen communicatiemodel

Broncodering

Omzetting van call, locator en power naar 50 bits

Er zijn drie typen WSPR-berichten:
- Type 1, bestaande uit callsign + 4-digit locator + dBm transmit power. Voorbeeld: PA3MRO JO22 33 (33 dBm is 2 W);
- Type 2, bestaande uit prefix/callsign of callsign/suffix + dBm transmit power. Voorbeeld: D/PA3MRO 33, als ik mijn transmissie uit Duitsland zou doen;
- Type 3, bestaande uit een hashed callsign + 6-digit locator + power. Dit type bestaat uit twee transmission cycles. Te herkennen aan <PA3MRO> JO22oi 33. De zes karakters lange locator is te lang om als type 1-bericht in 50 bits te worden gecodeerd. Er zijn daarom twee uitzendingen nodig. Om te zorgen dat het tweede deel van de uitzending gekoppeld kan worden aan het eerste, wordt gebruik gemaakt van een aparte code voor het callsign. Dit wordt een hash genoemd. Na het tweede deel van de uitzending wordt de bijbehorende call opgezocht via de hashwaarde. WSJT-X slaat alle hashes met callsign en locator op in het bestand 'hashtable.txt' (te vinden in de WSJT-X map). Als er nog geen hashwaarde van het callsign bekend is, bijvoorbeeld doordat alleen het tweede deel van de uitzending is ontvangen, dan zie je <...>. Zodra de eerste uitzending is geweest wordt de hash van de call toegevoegd aan de hashtabel.

Type 1 messages worden het meeste gebruikt en die licht ik toe. De payload bestaat uit in totaal 50 bits met de volgende indeling:
- 28 bits voor het callsign;
- 15 bits voor de locator;
- 7 bits voor het power level.

Coderen van het callsign naar 28 bits
De afspraak in het WSPR-protocol is dat op de derde plek in de call altijd een getal staat. Als de prefix voor het getal één karakter bevat, dan wordt op de eerste plaats een spatie gezet. Hetzelfde geldt als de suffix van de call minder dan drie karakters heeft, ook hier worden spaties toegevoegd. Er zijn 37 karakters mogelijk. De getallen 0 t/m 9 worden gecodeerd als waarden 0 t/m 9 en de letters A t/m Z worden gecodeerd met waarden van 10 t/m 35. Een spatie wordt gecodeerd als waarde 36.

Codering van de call wordt gedaan met onderstaande formule:
N1 = [Ch 1] Het eerste karakter kan bestaan uit een van alle 37 karakters inclusief de spatie;

N2 = N1 × 36 + [Ch 2] het tweede karakter kan alles zijn behalve een spatie;

N3 = N2 × 10 + [Ch 3] het derde karakter moet altijd een getal zijn, 10 mogelijkheden;

N4 = N3 × 27 + [Ch 4] – 10 het vierde karakter is een letter, 27 mogelijkheden;

N5 = N4 × 27 + [Ch 5] – 10 het vijfde idem;

N6 = N5 × 27 + [Ch 6] – 10 het zesde idem.
N6 bevat het eindresultaat, en als je dat omzet naar binair levert dat de eerste 28 bits van de payload op.

Voorbeeld: callsign PA3MRO
'P'= 25, 'A'= 10, '3'= 3, 'M'= 22, 'R'= 27, 'O'= 24

N1=	25	P
N2=	25 × 36 + 10 = 910	A
N3=	910 × 10 + 3 = 9103	3
N4=	9103 × 27 + 22 – 10 = 245793	M
N5=	245793 × 27 + 27 – 10 = 6636428	R
N6=	6636428 × 27 + 24 – 10 = 179183570	O

N6 omzetten naar binair levert:
10101010111000011111111010010
(28bits)

Coderen van de locator
De codering van de locator gebeurt met deze formule:
M1= (179 – 10 × [Loc 1] – [Loc 3]) × 180 + 10 × [Loc 2] + [Loc 4]
Letters in de locator lopen van A t/m R en krijgen de waarden 0 t/m 17. De getallen worden gecodeerd met de numerieke waarden 0 t/m 9.

Voorbeeld: mijn locator JO22.
Loc1= 'J'= 9, Loc2= 'O'= 14,
Loc3= '2'= 2, Loc4= '2'= 2

M1= (179 – 10 × 9 – 2) × 180 + 10 × 14 + 2 = 15 660 + 142 = 15 802

Coderen van uitgezonden vermogen
De laatste stap is het coderen van het uitgezonden vermogen. Dat gaat als volgt:
M= M1 × 128 + [Pwr] + 64

M1 is de waarde die is uitgerekend bij de locator, 15 802 in het voorbeeld.

Mijn uitgezonden vermogen is 33 dBm (2 W):
M= 15802 × 128 + 33 + 64 = 2 022 753

M omgezet naar binair levert:
011110110111010110000 1 (22= 15+7 bits)

De 50-bits payload wordt aangevuld met 0'en, in totaal 31 (voor de 'zero tailing', daar kom ik later op terug), waarmee de totale lengte van de payload 81 bits wordt.

De binaire payload voor PA3MRO JO22 33 wordt dus:
101010101110000111111101001001111101101110101100001000000000000000000000000000000000000

Kanaalcodering
Voordat het bericht verzonden kan worden moeten er nog een aantal zaken gebeuren:
- Forward error correction toevoegen. Hiermee worden extra informatiebits toegevoegd waarmee de ontvanger bitfouten kan corrigeren. Dit kunnen fouten zijn als gevolg van fading of andere verstoringen.
- Interleaving van de bits. Met interleaving worden de bits in een willekeurig volgorde geplaatst. Hiermee wordt voorkomen dat opeenvolgende bits wegvallen waardoor het decoderen niet meer lukt. Denk ook hier weer aan fading. Na deinterleaving heeft fading random bitfouten veroorzaakt in plaats van opeenvolgende fouten.
- Toevoegen van de synchronisatievector. Hiermee kan de ontvanger de tijdvertraging van het binnenkomende signaal meten. De tijdvertraging wordt veroorzaakt door een niet goed synchroon lopende klok bij de zender en/of de ontvanger.

Forward error correction (FEC)
Forward error correction voegt extra bits toe waarmee fouten aan de ontvangende kant kunnen worden hersteld. Een maat voor het herstellend vermogen van een code is de 'Hammingafstand' van de code. De Hammingafstand (d) is het aantal bitposities waarin twee verschillende codewoorden van elkaar verschillen. Hoe groter de Hammingafstand, hoe meer fouten kunnen worden hersteld. Het aantal bits dat je kunt corrigeren is

$$\#bits = \left\lfloor \frac{d-1}{2} \right\rfloor \quad (d \geq 1)$$

De rechte haken betekenen: afronden naar beneden naar een geheel getal. Een code met Hammingafstand d=5 kan per codewoord maximaal twee bits corrigeren.

RTTY is een voorbeeld van een code waarmee geen fouten kunnen worden hersteld. Het Baudotalfabet bestaat uit codewoorden van vijf bits, en daarmee kunnen $2^5 = 32$ codewoorden worden gemaakt. Een foutief bit in het ontvangen codewoord levert direct een fout karakter op. De Hammingafstand van de Baudotcode is d=1. Wat je vaak ziet bij RTTY is dat berichten twee keer verstuurd worden. Dit is ook een vorm van forward error correction. Beter nog is het om het bericht drie keer te sturen; als je twee dezelfde decodes hebt is dat het meest waarschijnlijke verzonden bericht.

Convolutiecodes
Bij WSPR is ervoor gekozen de brongecodeerde bits om te zetten naar een foutcorrigerende convolutiecode. Convolutiecodes zijn bedacht door Peter Elias in 1955. Convolutiecodes zijn geschikt voor codering van continue bitstromen, zoals video of audio. Convolutiecodes worden veel toegepast; ze werden onder meer gebruikt in de satellieten van de Pioneer- en Voyagerprogramma's die in de jaren '70 zijn gelanceerd. Ook in het Mars Pathfinder-programma zijn convolutiecodes gebruikt. Het foutcorrigerende vermogen van de convolutiecode maakt het mogelijk signalen van satellieten uit de diepe ruimte te kunnen decoderen.

Er is een theoretische limiet aan de capaciteit van een transmissiemedium, de 'Shannonlimiet' (Shannon, 1948). Door een lage bitrate te kiezen (onder de Shannonlimiet) en door steeds betere foutcorrigerende codes te ontwikkelen, met een hoge coding gain, is het gelukt dicht in de buurt van de Shannonlimiet te komen. Door gebruik te maken van een foutcorrigerende codering profiteren we van een aantal dB's 'versterking', de 'coding gain', en dat betekent dat je minder zendvermogen of antenneversterking nodig hebt om het bericht foutloos te kunnen versturen. De coding gain van de convolutiecode van het Pioneerprogramma is 7 dB vergeleken met ongecodeerde BPSK-signalen.

Een convolutiecode heeft een 'rate' (R), die aangeeft hoeveel outputbits de encoder genereert per inputbit. Een R=1/2 betekent dat voor iedere inputbit er twee outputbits worden gegenereerd. De 'constraint length' (K) is de lengte waarover foutcorrigerende informatie wordt gegenereerd. Daarnaast kan een code 'systematisch' of 'non-systematisch' zijn. WSPR gebruikt een rate R=1/2, constraint lenght (K) van 32, non-systematic convolutiecode. Het totaal aantal bits van de convolutiecode is 162 bits (81 ingangsbits ×2).

Een convolutie-encoder wordt opgebouwd met een schuifregister met een lengte van K–1, en een aantal exclusive OR-poorten afhankelijk van de rate van de code (R). De constraint length K van een convolutiecode bepaalt hoeveel toestanden de encoder heeft. Het aantal encodertoestanden is 2^{K-1}. Hoe langer de constraint length (K) en hoe lager de code rate van de encoder, hoe beter meestal de foutcorrigerende eigenschappen zijn. Een voorbeeld van de opbouw van een convolutie-encoder voor R=1/2 en K=3 is te zien in figuur 2.

De uitgangsbits van de encoder worden gevormd door de exclusive OR te berekenen van een combinatie van de inhoud van de verschillende schuifregisters. Bij een R=1/2 code wordt dat twee keer met verschillende combinaties gedaan. Die combinaties worden de 'generators' van de encoder genoemd. Een R=1/2 non-systematic code heeft twee generators. De generators bepalen de structuur en de foutcorrigerende eigenschappen van de convolutiecode. Om de code te kunnen decoderen moet de ontvanger de generators en de constraint length kennen. Generators worden meestal in octale vorm weergegeven.

Het verschil tussen een systematic en een non-systematic code is dat bij een systematic code het oorspronkelijke codewoord wordt verwerkt in de gecodeerde datastroom. Bij een non-systematic code worden alleen de parity bits gebruikt. Een non-systematic convolutiecode heeft betere foutcorrigerende eigenschappen dan een systematic convolutiecode, en dat willen we graag.

Convolutiecodes kunnen worden omgezet naar een code met een vaste lengte door de payload aan te vullen met. Dit wordt 'zero tailing' genoemd. Hiermee eindigt de encoder weer netjes in de 0-toestand.

Tussen twee convolutie-codewoorden wordt gesproken over de vrije Hammingafstand (d_{free}). De vrije Hammingafstand is een indicatie voor de foutcorrigerende eigenschappen van de convolutiecode. Bij blokcodes wordt de Hammingafstand (d) gebruikt, en bij convolutiecodes spreekt men over d_{free}. D_{free} is de minimale Hammingafstand tussen de codewoorden. Met eerdergenoemde formule kan worden bepaald hoeveel fouten er tenminste kunnen worden gecorrigeerd. Omdat de parity bits worden berekend over de lengte van het schuifregister, is dat het window waarover fouten kunnen worden gecorrigeerd. Doordat het 81 bit lange codewoord door de encoder wordt geklokt is er binnen een blok van 64 outputbits foutcorrectie mogelijk. Dit is de lengte van het schuifregister, vermenigvuldigd met het aantal outputbits dat gegenereerd wordt per inputbit. Over dit window van 64 bits zijn tenminste d_{free} fouten te corrigeren, maar vaak nog een paar extra. Foutcorrectie van convolutiecodes is het meest effectief als de fouten willekeurig voorkomen.

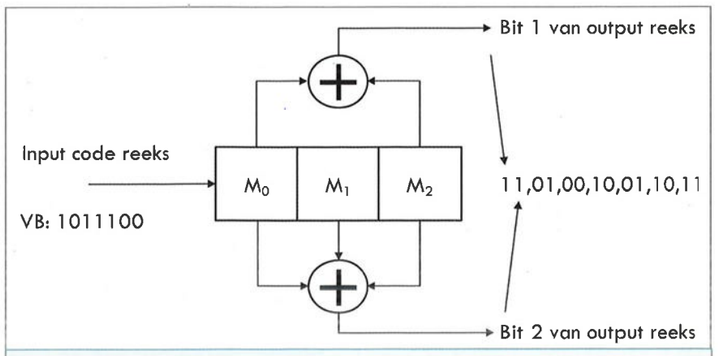
Een voorbeeld: in onderstaand voorbeeld is een R=1/2, constraint length K=3 non-systematic convolutiecode getoond.

Het aantal toestanden van deze encoder is $2^{K-1} = 2^{3-1} = 4$. Deze worden gevormd door M_1 en M_2. M_0 wordt niet meegerekend in het aantal toestanden, omdat de encoder bij een eerste bit nog geen gebruik maakt van geheugen. G_1= 5 en G_2= 7 zijn de generators van deze code.

De inputbits van de te coderen datastroom worden een voor een het schuifregister ingeklokt. Het proces start met een volledig met nullen gevuld schuifregister. Vervolgens wordt er twee keer een combinatie van de drie registers in een exclusive OR gestopt. Deze vormen samen de output van de decoder op basis van de bit-inputs. In figuur 2 wordt de combinatie van de registers aangegeven met G_1 (101(5)= M_0 XOR M_2en G_2(111(7)= M_0 XOR M_1 XOR M_2). In onderstaande figuur 3 is een voorbeeld te zien van het coderen van de inputstroom (1011100).

De outputstroom van de encoder is (11010010011011). Je ziet dat de encoder het aantal bits heeft verdubbeld. Van deze code is bekend dat d_{free}= 5,

Figuur 2 | Voorbeeld van een R=1/2, K=3 convolution encoder



Figuur 3 | Voorbeeld van het coderen van een inputstroom (begint bij linker ingangsbit)

waarmee twee bits te corrigeren zijn. Verderop in dit artikel leg ik uit hoe je de d_{free} van deze code kunt bepalen.

WSPR-convolutiecode

Voor het genereren van de convolutiecode voor WSPR is gekozen voor een R=1/2, K=32 non systematic code. Het schuifregister voor deze code heeft een lengte van 31. In totaal levert dit 2^{31} ofwel ruim twee miljard toestanden op.
Van de WSPR-code zijn de generators: G_1: 42545013236_{oct} en G_2: 70436206116_{oct} (in de source code van WSJT-X kom je G_1:$f2d05351_{hex}$ en G_2:$e4613c47_{hex}$ tegen. Dit zijn G_1 en G_2 omgekeerd, vanwege de bitvolgorde in het schuifregister in de C-programmeertaal). Wetenschappers Layland en Lushbaugh hebben deze code ontdekt.

De vrije Hammingafstand van de WSPR-convolutiecode heb ik in de literatuur kunnen terugvinden. Hij is d_{free}= 28, waarmee tenminste 13 bitfouten gecorrigeerd kunnen worden. Dit is het aantal fouten dat per 64 bits ongeveer gecorrigeerd kan worden. Voor het hele WSPR-bericht van 162 bits kunnen naar verwachting ongeveer 2,5×13= 33 bits gecorrigeerd worden.

Leuk om te weten is dat voor deze R=1/2, K=32 er waarschijnlijk een code bestaat met d_{free}= 36 (te bepalen met de 'Griesmer Bound'). De generators zijn alleen nog niet gevonden.

Bit interleaving

Burstfouten kunnen onder andere door fading optreden. Het signaal kan onderweg veel signaalniveau verliezen waardoor gedurende een bepaalde tijd detectiefouten ontstaan, of er wordt niets meer gedetecteerd. Bit interleaving wordt toegepast om de effecten van burstfouten te minimaliseren. Door de bits voor transmissie goed te verspreiden raakt een burstfout het verzonden bericht op willekeurige plaatsen in het hele codewoord, in plaats van in een reeks opeenvolgende bits. Aan de ontvangende kant moet het interleavepatroon bekend zijn, zodat de bits weer in de oorspronkelijke volgorde kunnen worden gezet. Dit heet deinterleaving.

Syncvector

De syncvector wordt opgeteld bij de interleaved bitstroom. De syncvector is een z.g. pseudo random binary sequence. Een pseudo random binary sequence heeft goede autocorrelatie-eigenschappen, waardoor deze goed is te detecteren aan de ontvangende kant. Het toevoegen van een dergelijke sequence maakt het mogelijk aan de ontvangende kant vertragingen te meten. Het signaal dat de modulator in gaat wordt samengesteld door:

$$symbol[n] = syncvector[n] + 2 * data[n]$$

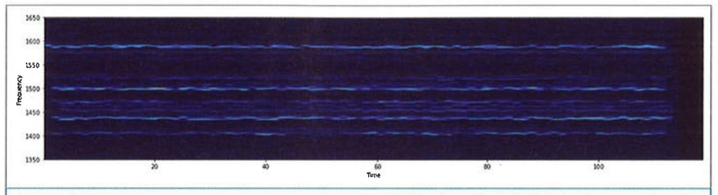
Hierin kan 'symbol' de waarden 0,1,2,3 hebben. Je kunt het beschouwen als een tweede kanaal dat in de data verstuurd wordt. De syncvector is bij de ontvanger bekend. In de zender worden deze symbolen uitgezonden op vier frequenties die ongeveer 1,46 Hz van elkaar liggen.

Het kanaal

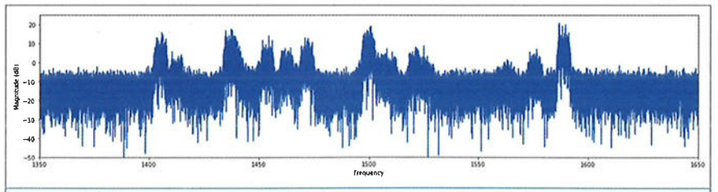
In figuur 4 is het spectrogram van een volledige opname op 40m te zien van de WSPR-signalen. In een spectrogram wordt met kleur de signaalsterkte per frequentie als functie van de tijd weergeven. Het is een 'waterval op zijn kant'. Merk op dat het startmoment van de verschillende zenders redelijk gelijk is. Een uitzondering is te zien op ongeveer 1460 Hz: hier beginnen de uitzendingen iets later, en gaan ook iets langer door. Er zijn sterke en zwakke signalen, en ook effecten van fading zijn zichtbaar (signaal rond 1400 Hz). Dit was een uitzonderlijk drukke uitzending; WSJT-X decodeerde maar liefst 22 signalen. Als we inzoomen op een paar signalen dan ziet dat eruit als in figuur 6.
In deze figuur zijn de tonen van het WSPR-signaal te zien. De kleur geeft aan hoe hoog het signaalniveau in de ontvanger is geweest. Hoe helderder (roder), hoe harder het signaal is ontvangen. Hier is ook te zien dat de signaalsterkte gedurende de ontvangst varieert (vanaf 80 s). In figuur 7 is een voorbeeld van een zwak signaal te zien. Gezien de kleur komt het nét boven de ruis uit.

Demoduleren van de bits

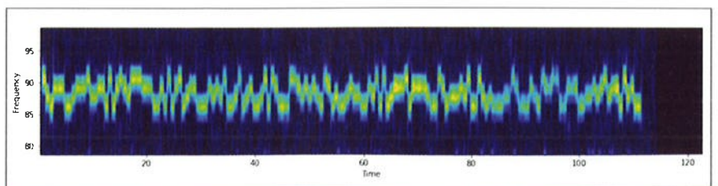
WSJT-X maakt een opname met een sample rate van 12 000 samples per seconde. De WSPR-signalen vallen in een 200 Hz breed audiogebied, van



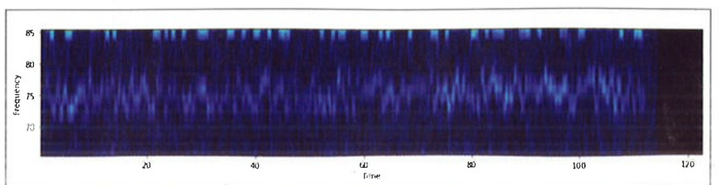
Figuur 4 | Spectrogram van een WSPR-recording op 40m



Figuur 5 | Spectrum van een 40m WSPR-sample



Figuur 6 | Uitvergroting van een WSPR-signaal met fading



Figuur 7 | Signaal in de ruis met fading

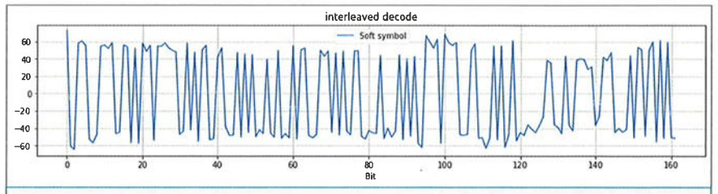
1400 t/m 1600 Hz als je de ontvanger precies op de standaard WSPR-frequentie zet. Met een bittijd van 0,683 sec zijn er 8196 samples per bit.

Het eerste wat er gebeurt in WSJT-X is dat het signaal wordt geresampled met een factor 32 en verschoven. Resamplen wordt toegepast om het aantal samples terug te brengen om het aantal berekeningen te beperken. Let wel dat WSJT-X binnen enkele seconden álle decoderingen toont. Er wordt gebruik gemaakt van een truc met complexe Fouriertransformaties om dit snel uit te voeren. Complex betekent hier: met behulp van complexe getallen (een uitbreiding op de reële getallen), waarmee amplitude- en faseinformatie wordt verwerkt. Na deze transformatie wordt er verder gerekend met de complexe 'I- en Q-signalen' (In phase en Quadrature). Het gevolg daarvan is dat er in de figuren hierna zo nu en dan negatieve frequenties te zien zijn. Fysisch heeft dit geen betekenis, maar wiskundig wel.

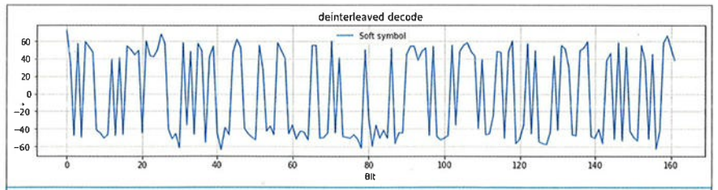
WSJT-X bepaalt vervolgens de pieken in het spectrum van de recording. Iedere piek in het signaal wordt onderzocht.

Om de bits te herleiden uit Figuur 6 wordt per bit (van 0,683 sec lang) een correlatie uitgevoerd met een viertal sinussen die 1,46 Hz uit elkaar liggen. Hierbij wordt de correlatie met de meegezonden syncvector bepaald. Dit levert per bit een correlatiegetal op, en voor alle bits samen een totaal synchronisatiegetal. Hoe groter dit getal, hoe beter. Door nu de tijdshift, de frequentie en de drift te variëren wordt de maximale synchronisatie bepaald. De correcte shift, frequentie en drift zijn essentieel voor een correcte decodering.

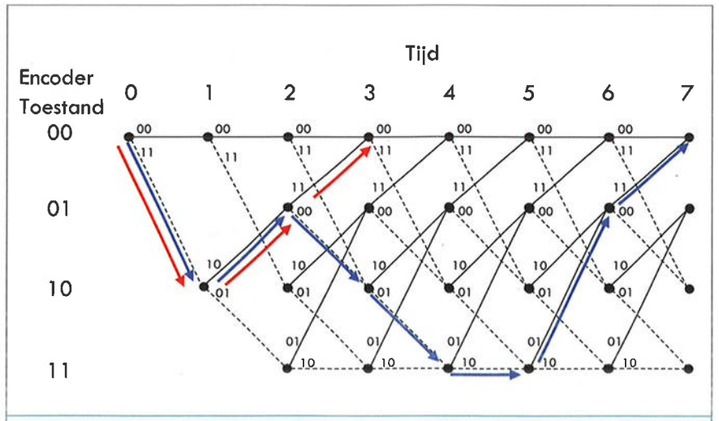
In figuur 8 beginnen al bits te verschijnen, maar het zijn geen 'harde' 0'en en 1'en. Afhankelijk van de sterkte van het signaal kunnen de waarden tussen −128 en 128 liggen. Wat opvalt is dat er variatie in de bits zit. We zouden er harde 0'en en 1'en van kunnen maken door van alle waarden boven nul een



Figuur 8 | Bits herleid uit het audiosignaal (nog interleaved)



Figuur 9 | Deinterleaved bits van Figuur 8



Figuur 10 | Trellis voor de R=1/2 G_1=5 G_2=7 non systematic convolution code

'1'te maken, en van alles onder de nul een '0'. Dit worden hard symbols genoemd. Daarmee hebben we een bitreeks die we verder kunnen behandelen. Maar hiermee gaat informatie verloren. Beter is het, de waarden uit Figuur 8 rechtstreeks te gebruiken, de soft symbols. Het voordeel daarvan is dat een waarde nét boven nul aangeeft dat het *misschien* een '1' is geweest. Een waarde die vér boven nul ligt is *hoogstwaarschijnlijk* een '1'. Door deze waarschijnlijkheid mee te nemen in het decodeerproces is winst te behalen: volgens de theorie levert dit zo'n 2 dB extra coding gain op. Dit wordt soft decision decoding genoemd. De variant waarbij de bits hard op 0 of 1 worden gezet wordt hard decision decoding genoemd.

Kanaaldecoder

Deinterleaven

De ontvanger kent het interleavepatroon van de zender, en kan daarmee de gedecodeerde soft symbols weer in de goede volgorde zetten. In figuur 9 zijn de soft symbols uit Figuur 8 in de goede volgorde gezet.

Decoderen van convolutiecodes

Voor het decoderen van een convolutiecode zijn diverse methoden beschikbaar. Bij het decoderen van een ontvangen bitstroom en het gebruiken van de foutcorrigerende eigenschappen van de convolutiecode wordt onderzocht wat de meest waarschijnlijke input-datastroom van de encoder is geweest die tot de ontvangen bitstroom heeft geleid.

Omdat de convolutiecode foutcorrigerende eigenschappen heeft, is het mogelijk aan de ontvangende kant fouten te herstellen. De ontvanger maakt gebruik van de structuur van de code die is verstuurd. Zoals eerder genoemd beschikt de ontvanger over de generatorpolynomen en de constraint length van de encoder, en kan daarmee het oorspronkelijke bericht herleiden.

Het bekendste algoritme voor het decoderen van een convolutiecode is de Viterbidecoder (Viterbi, 1970). Dit is de optimale decoder. Het is een 'maximum likelihood'-decoder. Hierbij maakt de decoder gebruik van de hele ontvangen code om de beste match te vinden. De Viterbidecoder maakt daarbij gebruik van een 'trellis' van de code. Een trellis is een grafische weergave van de code (zie het voorbeeld in figuur 10). Een gestippelde lijn is een één, en een ononderbroken lijn is een nul. Bij de lijn staat de encoderoutput. Merk op dat de structuur zich herhaalt.

De input-bitstroom uit figuur 3 (1,0,1,1,0,0) levert als outputstroom (11,01,00,10,01,10,11). Dit pad is weergegeven met blauwe pijlen in figuur 10.

Bij foutloze ontvangst van de bitstroom kan het pad teruggevonden worden, en daarmee de input-bitstroom. Bij fouten in de ontvangen bitstroom kan het meest waarschijnlijke inputpatroon worden teruggevonden door per twee bits te kijken naar de Hammingafstand tussen de ontvangen bits en de meest waarschijnlijke output.

Voor dit voorbeeld is het nog te doen, maar het is onpraktisch om deze methode toe te passen op convolutiecodes met grote constraint lengths. Met K=32 van de WSPR-code wordt dit een gigantische grote trellis. Na wat zoeken ben ik terechtgekomen op een implementatie van een Viterbidecoder voor K=24.

Met de trellis is ook d_{free} van de code te bepalen, door het pad in de trellis te zoeken dat met het kleinste gewicht (aantal enen in de output) terugkeert in de '00'-toestand (niet het nulpad). In figuur 10 is dit het pad (met rode pijlen) met inputreeks 100; dit levert een outputstroom (11,10,11) op. Deze reeks heeft vijf enen, waarmee d_{free}=5.

Een andere methode om convolutiecodes te decoderen is met een sequentiële decoder. Sequentiële decoders hebben geen beperking voor codes met grote constraint lengths, maar zijn iets 'minder optimaal' dan het Viterbi-algoritme. Sequentiële decoders gebruiken wel de codestructuur, maar maken geen gebruik van een trellis, en zijn daarmee geschikt om codes met grote constraint lengths te decoderen. Voorbeelden van sequentiële decoders zijn het stackalgoritme (Jelinek, 1970) en het Fano-algoritme (Fano, 1970).

WSJT-X gebruikt standaard het Fano-decoderalgoritme. Bij onderzoek van de sourcecode blijkt ook de stackdecoder van Jelinek erin te zitten. Deze is vanaf de command line te gebruiken, echter niet rechtstreeks uit WSJT-X.

Stackalgoritme
Voor mijn implementatie in python heb ik een stackdecoder geïmplementeerd. Een stack is letterlijk een stapel. De werking van het stackalgoritme is in grote lijnen:

- Stap 1: Zet de root node op de stack (dit is de 0-state van het schuifregister);
- Stap 2: Bereken de toestanden die vanuit de toestand van het bovenste element in de stack bereikt kunnen worden als een 0 wordt ingeklokt in de encoder; idem voor een 1;
- Stap 3: Bereken de metrics (score), en verwijder het bovenste element van de stack (startpunt van stap 2). Plaats +0 en +1 codewoord, toestand en metric op de stack en sorteer deze op aflopende metric;
- Stap 4: Is het einde van de code bereikt (bij WSPR 162 bits diep), dan is het bovenste codewoord in de stack het gedecodeerde en foutgecorrigeerde codewoord. Als het einde van de inkomende code reeks nog niét is bereikt, ga dan door met stap 2.

De metric kent een score toe aan de kwaliteit van het decoderen van steeds twee ontvangen bits. Door het verkennen van de te bereiken toestanden met een 0 en met een 1, weet de decoder ook wat volgens de structuur van de code de outputbits zouden moeten zijn. Door nu de ontvangen bits te vergelijken met de verwachte bits en daar een score aan te verbinden, ontstaat een totaalscore voor de meest waarschijnlijke inputbit. Hoe slechter de ontvangen bits overeenkomen met de verwachte bits, hoe lager de score. Bij een match wordt een kleine positieve score toegekend, bij een halve of geen match wordt een negatieve score toegekend. De stack wordt gesorteerd op metric van hoog naar laag. Op de bovenste plaats op de stack staat de meest waarschijnlijke gedecodeerde bitreeks. De minder waarschijnlijke codewoorden verdwijnen steeds dieper op de stapel. De decoder neemt alle informatie van alle bits mee in het bepalen van het meest waarschijnlijke verzonden bitpatroon en het bijbehorende input-bitpatroon (waar onze info in zit).

Het bovenstaande laat zich het beste uitleggen met een voorbeeld. In dit voorbeeld neem ik de simpele convolutiecode (K=3, R=1/2, G_1=5, G_2=7) uit figuur 2. In de tabel (verderop) is de werking van het stackalgoritme te zien. In de eerste kolom wordt een foutloze bitstroom aan de decoder toegevoerd. In de tweede kolom is een bitfout geïntroduceerd (rood). Onder 'code' staat de inputreeks, 'reconstructed' is het gereconstrueerde ontvangen signaal, en 'metric' is de score die de hele ontvangen bitreeks heeft gekregen.

In voorbeeld heeft de decoder het foutieve bit kunnen corrigeren. Stap 1 en 2 zijn gelijk. Bij stap 3 is er een afwijking van de verwachte twee outputbits en de twee ontvangen bits, waardoor de metric niet maximaal is. De ontvangen bits die hierna volgen leveren pluspunten op waardoor de gereconstrueerde code boven in de stack komt, en uiteindelijk leidt dit tot de correcte decodering van het ontvangen signaal. De items onder op de stack komen eventueel weer naar boven als de metric toeneemt. In bovenstaand voorbeeld is te zien dat de decoder 1 extra stap nodig heeft gehad.

Brondecoder

Decoderen van de payload
Als de decoder klaar is en de fouten zijn gecorrigeerd, beschikken we over de 81 bits die bij de zender zijn gegenereerd. Van deze 81 bits zijn alleen de eerste 50 relevant. Deze worden uitgepakt zodat call, locator en vermogen bekend zijn. Hiervoor worden de bits opgeknipt. De eerste 28 bits bevatten het callsign, dan volgen er 15 bits voor de locator en daarna 7 bits voor het vermogen. Door modulorekenen en restbepaling kan hieruit de informatie worden bepaald.

Stel dat we, na foutcorrectie, als eerste 28 bits deze bitstring hebben ontvangen:
1010101011100001111111010010

De decimale waarde van deze bitreeks is 179183570. Hier zitten de symbolen van het callsign in verwerkt. De karakters kunnen teruggewonnen worden door de volgende berekening (hierin is % de 'modulo'-bewerking, waarmee je de rest van een deling bepaalt):

- 179183570 % 27 = 14 (+10 levert het karakter 'O' op)
- ((179183570–14) / 27) % 27 = 17 (+10 levert het karakter 'R' op)
- ((6636428–17) / 27) % 27 = 12 (+10 levert het karakter 'M' op)
- ((245793–12) / 27) % 10 = 3 (levert het cijfer '3' op)
- ((9103–3) / 10) % 37 = 10 (levert het karakter 'A' op)
- ((910–22) / 37 % 37 = 25 (levert het karakter 'P' op)

Daarmee is het callsign (PA3MRO) gedecodeerd. Het decoderen van de grid locator en het vermogen gaat op vergelijkbare wijze.

Hiermee is de hele WSPR-codeer en -decodeercyclus gereed, maar er is meer...

Verborgen signalen

Het komt vaak voor dat WSPR-stations op dezelfde frequentie uitzenden, en dat het WSJT-X tóch lukt beide signalen te decoderen. Dat werkt als volgt. Na het succesvol decoderen van het sterkste signaal zijn call, locator en vermogen bekend, maar ook de tijdvertraging en drift. Met deze info wordt een reconstructie gemaakt van het signaal zoals dat is verstuurd.

Received bits: 1101001001			
1	code	reconstructed	metric
	1	11	1.0
	0	00	-9.0
2	code	reconstructed	metric
	10	1101	2.0
	11	1110	-8.0
	0	00	-9.0
3	code	reconstructed	metric
	101	110100	3.0
	100	110111	-7.0
	11	1110	-8.0
	0	00	-9.0
4	code	reconstructed	metric
	1011	11010010	4.0
	1010	11010001	-6.0
	100	110111	-7.0
	11	1110	-8.0
5	code	reconstructed	metric
	10111	1101001001	5.0
	10110	1101001010	-5.0
	1010	11010001	-6.0
	100	110111	-7.0
rcvd:		1101001001	
reconstructed:		1101001001	
Decoded		10111	
Total Steps		5	
metric:		5.0	

Received bits: 1101101001			
1	code	reconstructed	metric
	1	11	1.0
	0	00	-9.0
2	code	reconstructed	metric
	10	1101	2.0
	11	1110	-8.0
	0	00	-9.0
3	code	reconstructed	metric
	100	110111	-2.0
	101	110100	-2.0
	11	1110	-8.0
	0	00	-9.0
4	code	reconstructed	metric
	101	110100	-2.0
	1000	11011100	-6.0
	1001	11011111	-6.0
	11	1110	-8.0
5	code	reconstructed	metric
	1011	11010010	-1.0
	1000	11011100	-6.0
	1001	11011111	-6.0
	11	1110	-8.0
6	code	reconstructed	metric
	10111	1101001001	0.0
	1000	11011100	-6.0
	1001	11011111	-6.0
	11	1110	-8.0
rcvd:		1101101001	
reconstructed:		1101001001	
Decoded		10111	
Total Steps		6	
metric:		0.0	

Bij de ontvanger heb je een signaal met alle effecten van het transmissiepad. Gedurende de transmissie is het signaal verzwakt, vanwege fading. Door het ontvangen signaal te delen door de gereconstrueerde versie zoals die blijkbaar bij de zender moet zijn verstuurd, kun je de channel gain uitrekenen. Als je het gereconstrueerde signaal vermenigvuldigt met de channel gain krijg je het ontvangen signaal zoals je dat zou hebben ontvangen zonder het onderliggende signaal. Door nu dat signaal af te trekken van de opname wordt het eerst gedecodeerde signaal volledig verwijderd. Nu worden eventueel onderliggende uitzendingen zichtbaar, die weer gedecodeerd kunnen worden. Dit levert een aantal extra decoderingen op. Voorbeelden volgen.

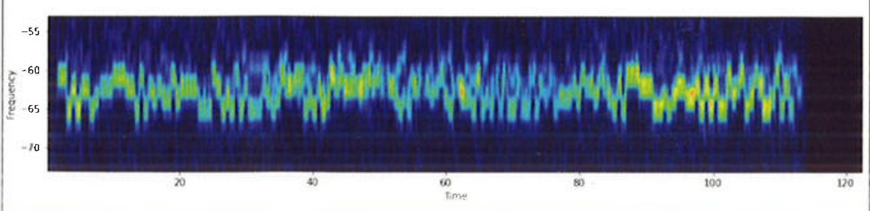
WSPR-decoder in actie

Voldoende uitgelegd; tijd om de decoder in actie zien. In de voorbeelden heb ik opnames gebruikt van WSJT-X. WSJT-X slaat alle audio-opnames op (als je dat aanzet). Steeds wordt de output van het algoritme getoond. De eerste regel bevat de mate van synchronisatie, dan de tijdsverschuiving, de frequentie en signaalsterkte. Vervolgens is de ontvangen ongecorrigeerde deinterleaved bitstroom ('rcvd',162 bits) getoond. Daarna volgt de gecorrigeerde bitstroom ('reconstructed'). 'Decoded' zijn de informatiebits (81 bits) waarin callsign, locator en power zijn gecodeerd. Vervolgens informatie over het stack-decoderproces, het aantal stappen dat de decoder heeft uitgevoerd, en de metric. Dan het aantal fouten dat is gerepareerd, en als laatste de decodering van callsign, locator en grid. Ten slotte wordt aangegeven wat het type WSPR-bericht is.

Het eerste voorbeeld is een redelijk hard signaal zonder al te veel fadingeffecten (figuur 11).

De decoder vindt de volgende informatie:
sync: 0.494443 shift 656 -62.99 Hz
drift 0 -12.87 dB

rcvd: *11101111100011010010100100101111100110110011111001010001010001010011000111001101111100000111111101101010101010111001010011101100111110000100010011000101101000 0011*



Figuur 11 | Ontvangen signaal van G4CAO IO91 27

reconstructed: *11101111100011010010100100101111100110110011111001010001010001010011000111001101111110000011111110110101010101011100101001110110011111100000100010011000101101000 0011*

decoded 111101100101101010110000110001111111010101101101100000000000000000000000000000
Decoder steps 81
metric: 690
Errors corrected 0
Decoded payload: G4CAO IO91 27
Type 1: 6 digit call, grid, power

De decodering is uitgevoerd zonder dat er foutcorrectie nodig is. De decoder heeft 81 stappen uitgevoerd. Als er geen fouten zijn is dit het minimumaantal stappen. Zodra er fouten worden gevonden, zie je dat terug in het aantal stappen dat de decoder doorloopt. Ook de metric is te zien (de score). Hoe hoger hoe beter.

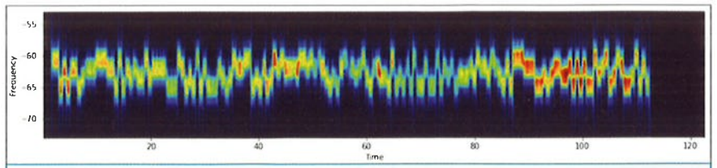
Om te onderzoeken of er nog een signaal onder G4CAO zit wordt het verzonden signaal gereconstrueerd, inclusief de channel gain. In figuur 12 en 13 is dat gereconstrueerde signaal te zien. Hierin zijn drift, tijdvertraging en channel gain verwerkt.

Het gereconstrueerde signaal van G4CAO trekken we af van de oorspronkelijke opname, en er blijkt een signaal onder G4CAO verborgen te zitten.

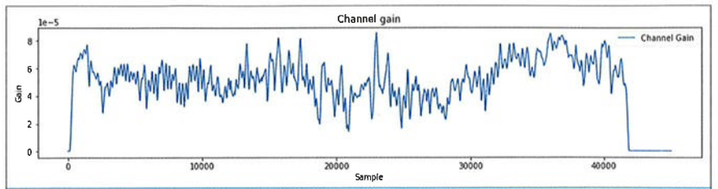
Het verborgen signaal in figuur 14 varieert in sterkte en is zwakker dan G4CAO.

Dit resultaat gaat opnieuw de decoder in.
sync: 0.352417 shift 1060 -60.79 Hz
drift 0 -22.56 dB

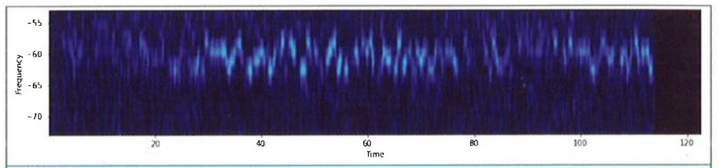
rcvd: *110110011111101001100110001110111010011111100010101101110101100001100110100100011101011000011111000000111001100011000010010110100101110101000011011010000101001111*
reconstructed: *11011001111100100010011000011010101001111110001010000111010111000110011010010001111100011000011111000000111001100*



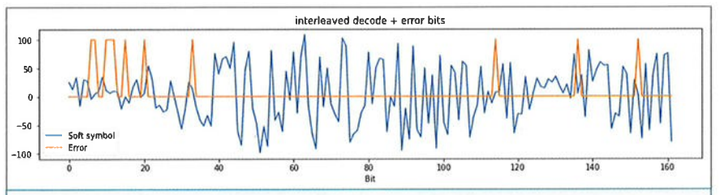
Figuur 12 | Gereconstrueerd signaal van G4CAO



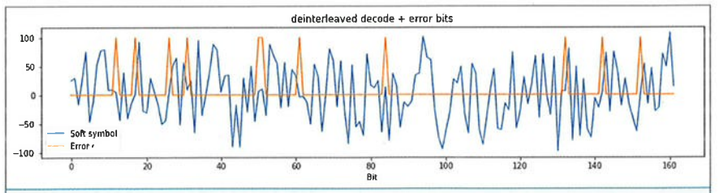
Figuur 13 | Channel gain van G4CAO



Figuur 14 | Verborgen signaal zichtbaar na signaalsubstractie



Figuur 15 | Interleaved bits met bitposities waar fouten zijn gecorrigeerd (oranje)



Figuur 16 | Deinterleaved bits en errors

01100001001011010010101010000
0101101000110100111

decoded *10100101011011111000100*
1
11101101101011101111001010000
0
00000000000000000000000000
Decoder steps 82
metric: 446
Errors corrected 11
Decoded payload: OH3HTI KP21 37
Type 1: 6 digit call, grid, power

OH3HTI blijkt onder het signaal van G4CAO te zitten. De decoder heeft 11 bits kunnen herstellen. In figuur 15 zijn de foutieve bitposities weergegeven in oranje (de pieken geven alleen aan waar er een fout is gecorrigeerd, niet wat de waarde had moeten zijn). Goed te zien is dat vooral aan het begin, waar het signaal (zie figuur 14) het meest verzwakt is, de meeste fouten zijn gecorrigeerd.

In figuur 16 is te zien dat na deinterleaving de bit fouten redelijk verspreid verdeeld zijn, wat het foutcorrectieproces van de stackdecoder helpt.

Reconstrueren van OH3HTI

Het signaal van OH3HTI wordt opnieuw gereconstrueerd om het uit de opname te verwijderen. In figuur 17 is de berekende kanaalverzwakking te zien. Deze komt overeen met de verzwakkingen zichtbaar in het ontvangen signaal van figuur 14.

Dit gereconstrueerde signaal van OH3HTI trekken we af van het signaal in figuur 14, waarna te zien is dat er geen signaal meer onder zit.

In figuur 19 zijn G4CAO en OH3HTI verdwenen uit de opname.

Nog een voorbeeld uit een andere opname (figuur 20, 21, 22, 23):
sync: 0.572146 shift 416 -60.79 Hz
drift 0 -2.09 dB

rcvd: *00110110101010110101010000*
10011011111100000101000111011
11110001101001011011101001101
00111101011011000011101010010
01010011100001101110100100001
0
1100110001101001111
reconstructed: *00110110101010110*
10
01010001001101111110000010100
01
11011111100011010000110111010
01
10100111111011011000011101001
00
1001010011100001101110100100
01
0110011000110100111

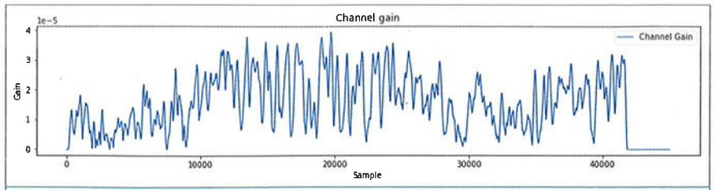
decoded *01011011101001001001101*
1
01100111100010011101100100000
00000000000000000000000000
Decoder steps 81
metric: 556
Errors corrected 2
Decoded payload: DK2DB JN48 37
Type 1: 6 digit call, grid, power

Na substractie wordt een verborgen signaal zichtbaar (figuur 24). Opnieuw decoderen levert:
sync: 0.359051 shift 672 -63.72 Hz
drift 0 -13.74 dB

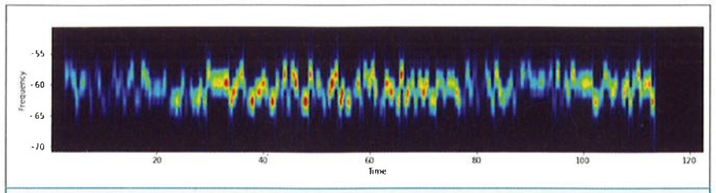
rcvd: *11101111100011010010010*
01001100001101100011110010100
01000001010011000111001101111
10000011111110110101010001011
00101001110010001111000010000
0
01100000110100000011
reconstructed: *11101111000110100*
10100100101111100110011011111
00
10100010100010100110001110011
01
11110000011111110110101010101
11001010011101100111110000100
01
00110001011010000011

decoded *01011011101001001001101*
1
01100111100010011101100100000
00000000000000000000000000
Decoder steps 84
metric: 518
Errors corrected 11
Decoded payload: G4CAO IO91 27
Type 1: 6 digit call, grid, power

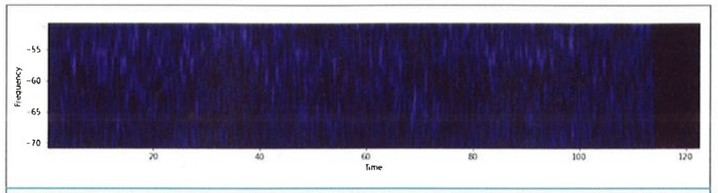
De decoder heeft het iets moeilijker en kan 11 bits corrigeren, zoals te zien in figuren 25, 26 en 27.



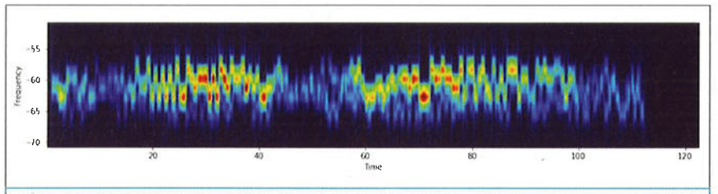
Figuur 17 | Channel gain van OH3HTI



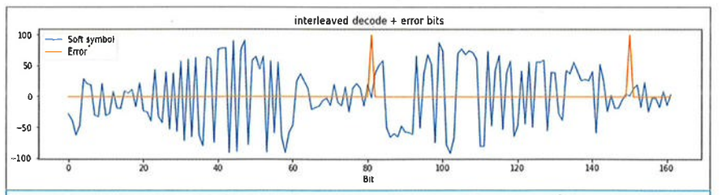
Figuur 18 | Reconstructie van OH3HTI



Figuur 19 | Na substractie van OH3HTI uit de opname



Figuur 20 | Signaal van DK2DB JN48 37



Figuur 21 | Bitfouten(2) DK2DB

WSPR-voorbeelden
Voorbeeld waarbij veel fouten zijn gecorrigeerd (figuur 28, 29):
sync: 0.202371 shift 416 -98.14 Hz
drift 0 -31.95 dB

rcvd: *10110010000000010011011001*
11100010011100100111111011111
11100100011000000010110101010
00110100100100001011101010110
00110100001011100110000001000
01101110010001100 11
reconstructed: *00110110101010110*
11
10110011110001000111100110111
0
11111111101000110000110101110
11
00101010101110100101011110100
1
10001100001010110110010000000
01110110001010110011

decoded *01011011110011010010111*
1
00100111101000001111010111000
00
00000000000000000000000000
Decoder steps 229
metric: -26
Errors corrected 31
Decoded payload: DL0PBS JO33 23
Type 1: 6 digit call, grid, power

Te zien is dat de decoder flink meer stappen (229) nodig had om goed te decoderen. In totaal zijn er 31 bits gecorrigeerd.

Voorbeeld VK3MO, figuur 30:
sync: 0.298288 shift 320 79.10 Hz
drift 0.00 -13.90 dB
rcvd:
11100010110010111111111011011
01000100000011010010110000101
1
00101001010011010000011111110
1
11011111010101100110001101100
01110011101001000111111010100
10001101001111
reconstructed: *11100010110010111*
1
11101011101010001000000110100
1
01100001011001010010101011010000
01111110011101101001010110011100
10110100011100111010010011101010
10101011000110100111

decoded *11010101010011100011100*
0
10010001100001010001100101000000
00000000000000000000000000
Decoder steps 83

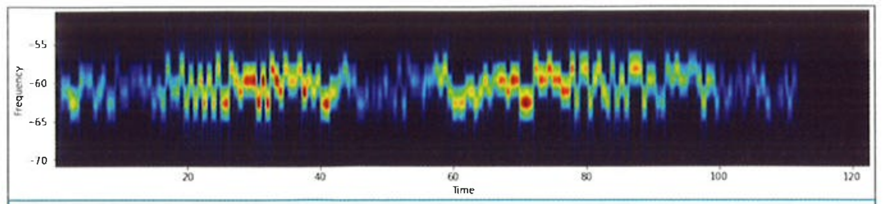
Schematheek wordt digitaal

Ik heb afgelopen tijd diverse dubbele papieren manuals in Electron geplaatst, en er zijn maar vier amateurs op deze boeken afgekomen.

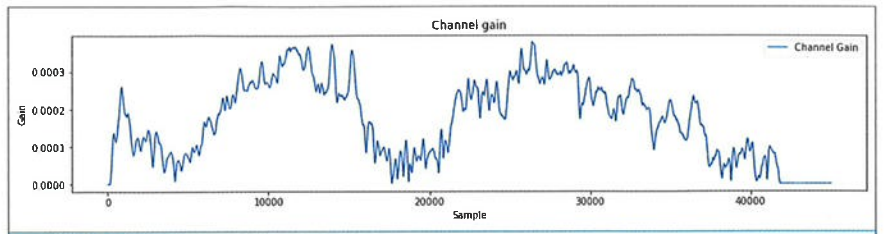
Dus, voor de laatste maal: doe er nu je voordeel mee, want na de beurs in Rosmalen is het afgelopen met dubbelen. En wat er dan mee gaat gebeuren weet ik nog niet.

Zelf ben ik ook niet meer de jongste, dus moet ik helaas drastische maatregelen nemen. Na de beurs heb ik dus enkel nog digitale schema's.

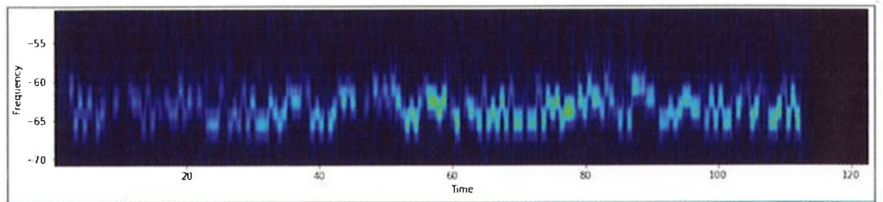
73' van Toine PD0MHS



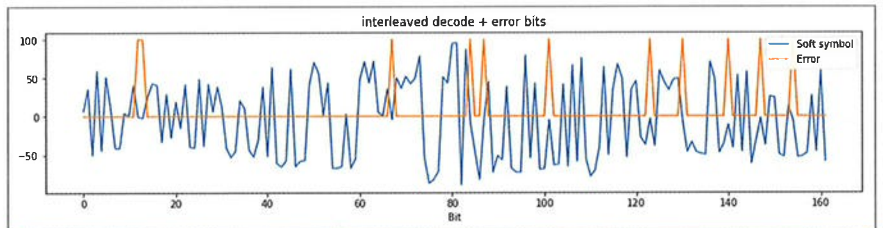
Figuur 22 | Gereconstrueerd signaal van DK2DB



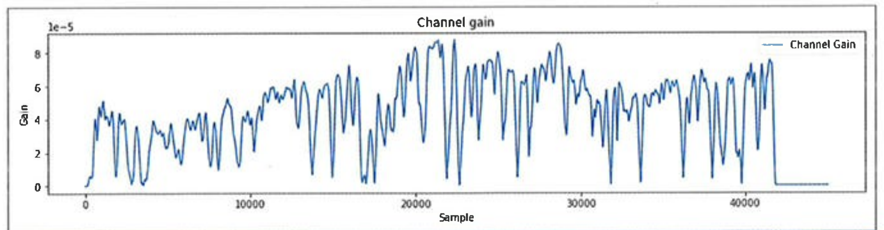
Figuur 23 | Channel gain DK2DB



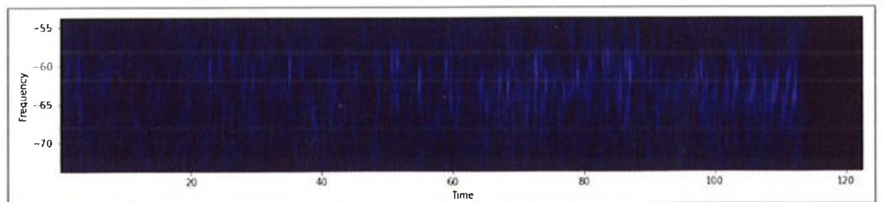
Figuur 24 | Verstopt signaal onder DK2DB is G4CAO



Figuur 25 | Bitfouten (11) in de ontvangst van G4CAO



Figuur 26 | Channel gain van G4CAO



Figuur 27 | Na substractie; het algoritme heeft niet alles kunnen verwijderen

metric: 332
Errors corrected 12
Decoded payload: VK3MO QF22 37
Type 1: 6 digit call, grid, power

De decoder heeft 12 bits kunnen herstellen.

Voorbeeld HS0AJ, figuur 31:
sync: 0.193842 shift 384 -5.13 Hz drift 0.00 -20.97 dB

rcvd: *1110101111101111010100010*
1011100010011100101010000111101
0001110101101011100110001111000
0111110111101010010000100011O
0011010011111111000011110001O1
100011110011101110O
reconstructed: *001110111110110101*
0100000101110001001110010101000
0111010000000100101011100110001
1100001111101111010100101001000
1100011010011101101001011001100
10100111101111001101

decoded *011110000001000101100000*
0110100110111101001110111100000
000000000011011110101001111
Decoder steps 150
metric: 155
Errors corrected 19
Decoded payload: HS0AJ OK03 30
Type 1: 6 digit call, grid, power

De decoder heeft 19 bits kunnen herstellen.

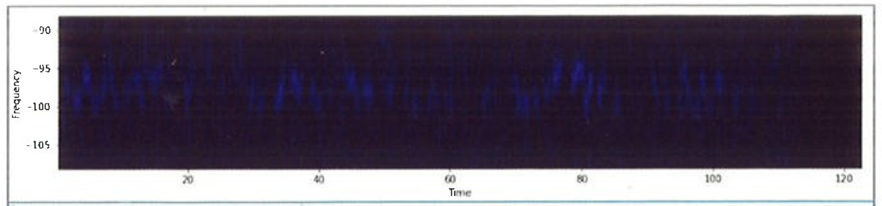
Conclusie

In dit artikel heb ik beschreven hoe de cyclus van codering tot decodering van een WSPR-signaal werkt. Dat levert hopelijk veel inzicht over de achtergrond en werking van WSPR, en natuurlijk mooie plaatjes. Er zit een hele wereld achter deze technologie. De foutcorrigerende eigenschappen van de convolutiecode en de lage datarate die WSPR gebruikt maken het mogelijk met minder vermogen toch grote afstanden af te leggen.

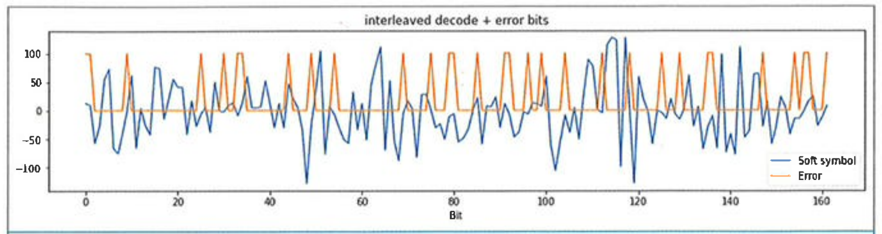
WSJT-X en JTDX zijn programma's om o.a. WSPR te decoderen. Deze software zijn technische hoogstandjes op het gebied. Joe Taylor (K1TJ) en Steve Franke (K9AN) hebben daarmee fantastische software ontwikkeld voor de zendamateur, waarmee de grenzen van de communicatietechnologie wordt opgezocht.

Is uw roepnaam gewijzigd?

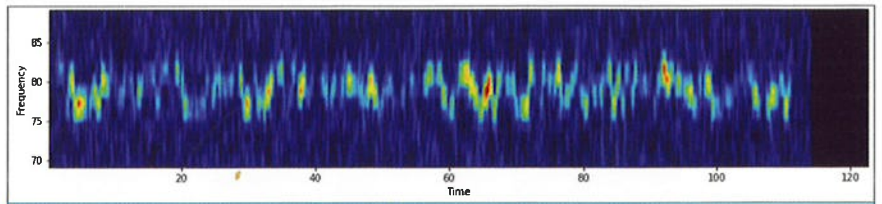
Heeft u een nieuwe roepnaam aangevraagd? Geef het door aan de ledenadministratie via
centraalbureau@veron.nl, veroncb@veron.nl



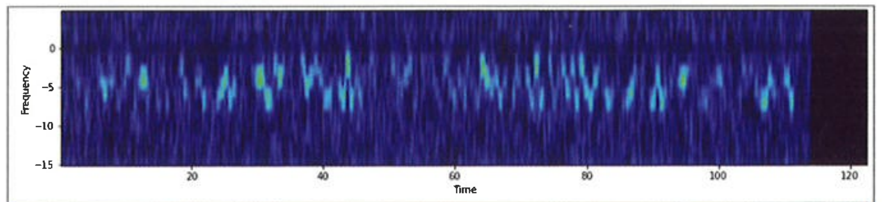
Figuur 28 | Ontvangen signaal van DL0PBS JO33 23



Figuur 29 | Gecorrigeerde bits (31) van DL0PBS



Figuur 30 | Op 20m ontvangen signaal van VK3MO QF22 37



Figuur 31 | Op 20m ontvangen signaal van HS0AJ OK03 30

Geraadpleegde bronnen:

[1] https://swharden.com/software/FSKview/wspr/
[2] 'The FT4 and FT8 communication protocols', Steve Franke, Bill Somerville, Frank Taylor, July/August 2020, *QEX magazine*, ARRL
[3] *Error Control Coding (2nd Edition)*, Shu Lin, Daniel J. Costello, 2004
[4] *Fundamentals of Convolutional Coding*, Rolf Johannesson, Kamil Sh. Zigangirov,1999
[5] *Error correcting coding*, Todd K. Moon, 2005.

De links zijn direct aanklikbaar op www.veron.nl/electronlinks

Dick Fijlstra PA0DFN Carole Perry Educator of the Year 2022

Dick Fijlstra PA0DFN ontvangt de Carole Perry Educator of the Year Award. Dat maakt de organisatie van de Orlando HamCation bekend. De award erkent een uitstekende bijdrage aan het onderwijs en bevordering van de jeugd in amateurradio.

De prijs werd voor het eerst uitgereikt in 2018 aan naamgever Carole Perry WB2MGP. Orlando HamCation en ARRL sponsoren de HamCation, de ARRL National Convention van 10 tot 13 februari 2022.

Special event rond honderdjarig BBC

Leden van de BBC-radioclub, The London BBC Radio Group, hebben een speciale roepnaam gekregen om in 2022 het hele jaar door het honderdjarig bestaan van de Britse omroep BBC te vieren.

Ofcom (het Britse Agentschap Telecom) heeft de roepnaam GB100BBC toegekend om het hele jaar door op de amateurbanden te kunnen werken. Het begint allemaal om middernacht op nieuwjaarsdag, vanaf het BBC-hoofdkantoor in Broadcasting House in Londen.

Operationele slots worden vervolgens toegewezen voor gebruik door individuele leden en lokale groepen operators. Zij werken zowel vanuit hun eigen QTH- of BBC-locatie in het Verenigd Koninkrijk.

Repeaterkaart met meer dan 4000 repeaters wereldwijd

De nieuwe repeaterkaart geeft via een webbrowser toegang tot informatie over meer dan 4000 repeaters wereldwijd. Ze staan als groene stippen op de kaart weergegeven. Door op zo'n stip te klikken, verschijnen de gegevens van het betreffende relaisstation (call, in- en uitgangsfrequentie en transmissiemode).

De verzameling relaisstations kun je ook vooraf filteren op basis van frequentie en/of transmissiemode, waardoor uitsluitend die stations op de kaart verschijnen waarin je echt geïnteresseerd bent. De maker van Repeatermap is Martin DK3ML.

Het is een handig stuk gereedschap om op een snelle manier te achterhalen welke repeaters zich in een bepaalde omgeving bevinden. De kaart beslaat veel landen van de wereld en je kunt in- en uitzoomen.
Zie de site: www.repeatermap.de