Een Open Source Transceiver op basis van de DRA818V

Robert de Kok, PA2RDK

Hoe het begon

Op een regenachtige middag kwam ik ergens een verhaal tegen over een DRA818V. De Dorji DRA818 is een complete 1Watt VHF FM transceiver voor 134-174 MHz. De module is voorzien van CTCSS, de mogelijkheid om te schakelen tussen een ½ Watt en 1 Watt output, squelch en scanmogelijkheden. De DRA kost bij diverse bronnen op EBay rond de € 12,50 en is maar een klein beetje groter dan een standaard 40pens IC. De module is weliswaar SMD maar kan met een gewone soldeerbout worden gesoldeerd en wordt bestuurd middels RS232 commando's op TTL niveau. Dit wil zeggen dat de signalen schakelen tussen de 0 en 5 volt. Standaard RS232 signalen schakelen tussen -12 en +12 volt. Het is dus niet verstandig de module rechtstreeks op een PC aan te sluiten. Dat overleeft die vast niet! Er is ook een 70cm versie beschikbaar, de DRA818U.

De specificaties van de DRA818x:

- **Frequency Range VHF: 134~174MHz**
- **Frequency Range UHF: 430~470MHz**
- **Tx/Rx frequency independant**
- **Channel space: 12.5/25KHz**
- **Configurable multi-channels**
- **Sensitivity: -122dBm**
- **Output power: +27/30dBm**
- **CTCSS / CDCSS codes**
- **8 volume levels**
- **8 squelch levels**
- **UART interface**
- **Temperature: -20°C ~+70°C**
- **TX current: 450/750mA**
- **Supply voltage: 3.3~4.5V**

Met een minimaal aantal componenten rond de DRA is het mogelijk om een volledige VHF (of UHF) transceiver te bouwen. Een 5 Volt voeding, een electretmicrofoon, een audioversterker (LM386) met speaker en een besturing in willekeurige vorm voldoen om met de DRA een transceiver te bouwen. Met deze wetenschap ben ik op het alwetend internet eens gaan zoeken wat er allemaal met zo'n leuke module mogelijk is!

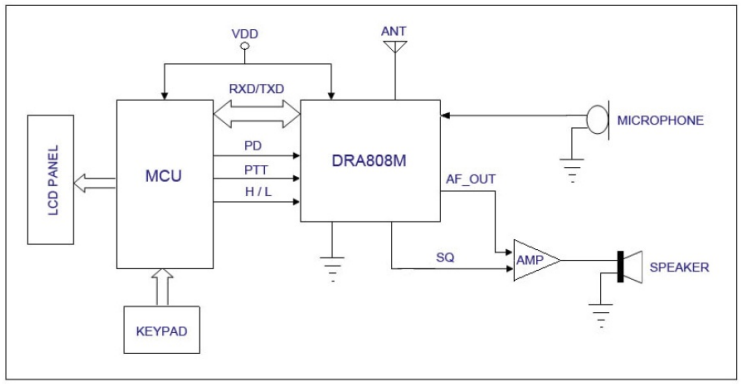
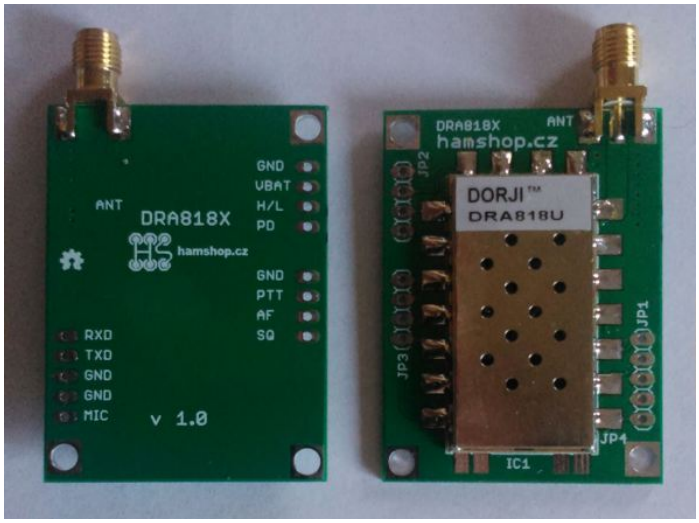


Figure 3: DRA818U Application Circuit

Al snel bleek dat ik niet de eerste ben die met de DRA in de weer zou gaan. Over de hele wereld, dus ook in Nederland, is er door zendamateurs van alles gebouwd met deze module, waarbij de besturing uiteraard het belangrijkste verschil maakt. Er bestaan de nodige ontwerpen rondom een Arduino, maar ook met een Raspberry PI als besturing. Volgens mij een beetje 'overkill' want een Raspberry staat zich echt dood te vervelen als die alleen een DRA en een LCD scherm moet aansturen.

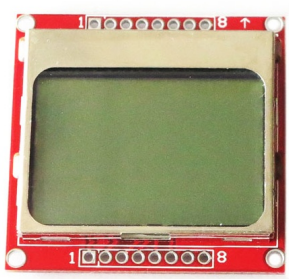
Wat alle door mij gevonden projecten gemeenschappelijk hadden is het feit dat de software vaak niet af was en alleen maar in een stadium, net voldoende om de module te testen.

Dit vind ik natuurlijk niet erg, want er moet wel wat te ontwikkelen over blijven. Gewoon in elkaar zetten wat een ander bedacht heeft vind ik net iets te makkelijk. Kortom, ik ging op zoek naar een basis waarop ik mijzelf helemaal kon uitleven.

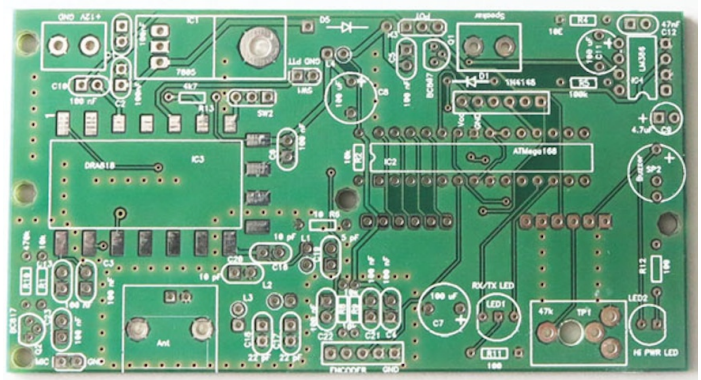


Op hamshop.cz[1] kwam ik een mooi printje tegen voor omgerekend nog geen € 4,- waar de DRA op gesoldeerd kan worden, zodat deze handelbaar wordt en bijvoorbeeld op een breadboard geprikt kan worden. Dit biedt de mogelijkheid om de module eenvoudig aan te sluiten op bijvoorbeeld een Arduino om hiermee te gaan 'prototypen'. Deze Tsjechische firma levert ook de module[2] voor omgerekend € 12,50. Vervolgens raakte ik aan het twijfelen of ik met een Arduino of een PIC in de weer zou gaan. Prototypen met een Arduino is leuk, maar ik ben er geen liefhebber van om een Arduino als basis van een product te maken. Liever gebruik ik een kale processor (bij voorkeur een PIC) en bouw het geheel op een stukje veroboard.

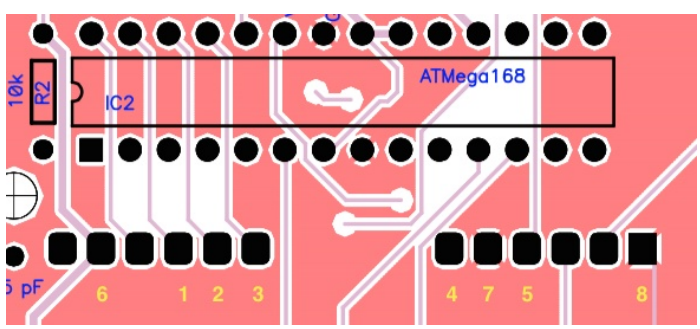
Toen kwam ik het blog tegen van Jurij Mikeln, een Sloveense zendamateur. Onder de naam 'A weekend Project' heeft Jurij een print en software ontwikkeld voor de DRA818 en een Atmega328 processor. Dit is (bijna) dezelfde processor als ook op onder andere de Arduino Uno wordt gebruikt. Dit biedt theoretisch de mogelijkheid om de processor te programmeren met de Arduino IDE. Jurij had er echter voor gekozen de software te ontwikkelen in Bascom, een soort Basic (hebben we allemaal toch geleerd op school?) maar dan voor Atmel processoren. Het eerste ontwerp van Jurij was vrij basaal met een 2x16 karakter LCD scherm



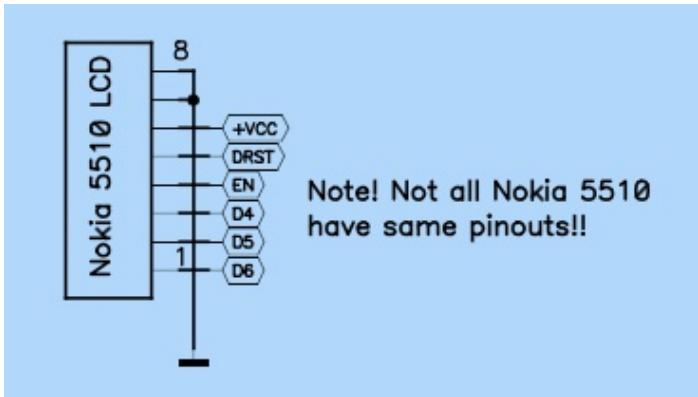
maar ook hij had meer dan een weekend nodig en na het eerste prototype volgde er een versie met een Nokia3310 grafisch LCD scherm. Hier had ik nog nooit mee gewerkt, dus dat was leuk. Jurij heeft ook een webshop en ik besloot om bij hem een print+module[3] en een LCD[4] te bestellen. Onderhand heeft Jurij in zijn webshop een complete kit[5], inclusief alle onderdelen, beschikbaar, dus het wordt steeds makkelijker. Al na een paar dagen kwamen de spullen binnen en kon het knutselen beginnen.



De print bleek van een uitstekende kwaliteit en op het masker van de print staan alle componenten genoemd. Met het schema voor de details er bij, was de print in een uurtje van alle componenten voorzien. Iets lastiger bleek het om uit te zoeken hoe het display bedoeld was om aangesloten te worden, de print is ontwikkeld voor een standaard 2x16 karakter LCD display en op een deel van deze aansluitingen moest de Nokia LCD worden aangesloten. Het display heeft 8 aansluitingen, deze aansluitingen kunnen verschillen per leverancier, maar het display welke Jurij levert heeft onderstaande aansluitingen en dient aangesloten te worden zoals genummerd op onderstaande uitsnede van de print.

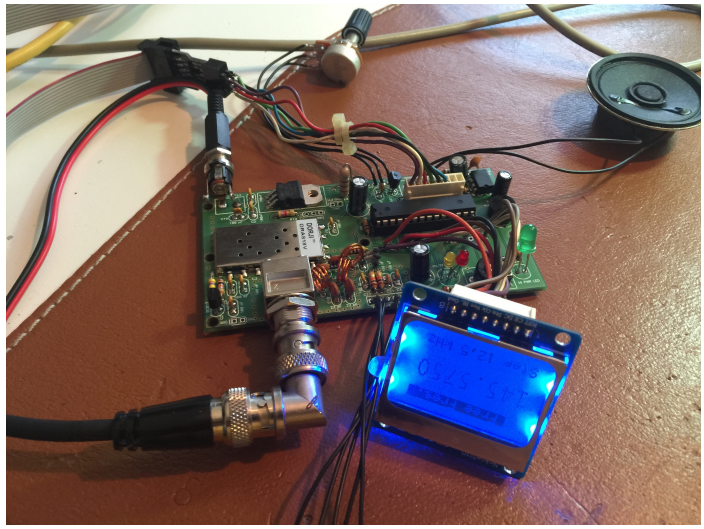


Nummering van de Nokia aansluitingen. Deze komen overeen met de nummering zoals getoond op de volgende bladzijde.

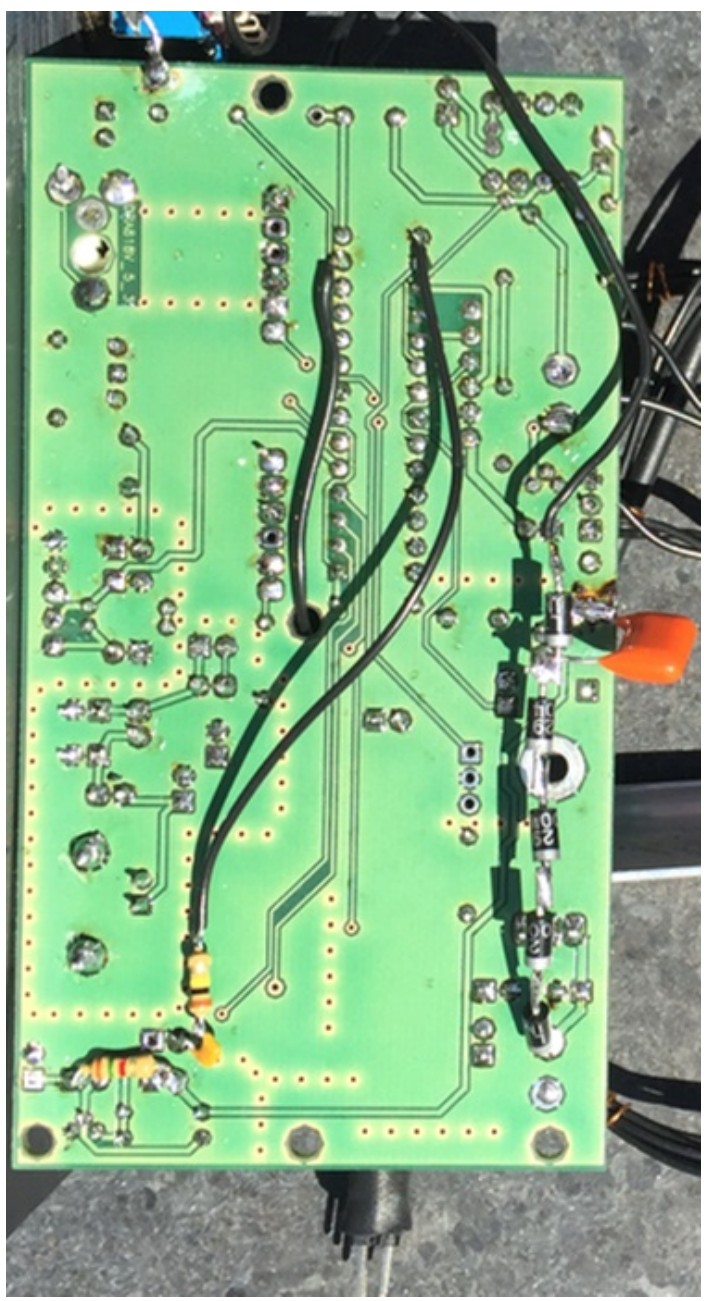


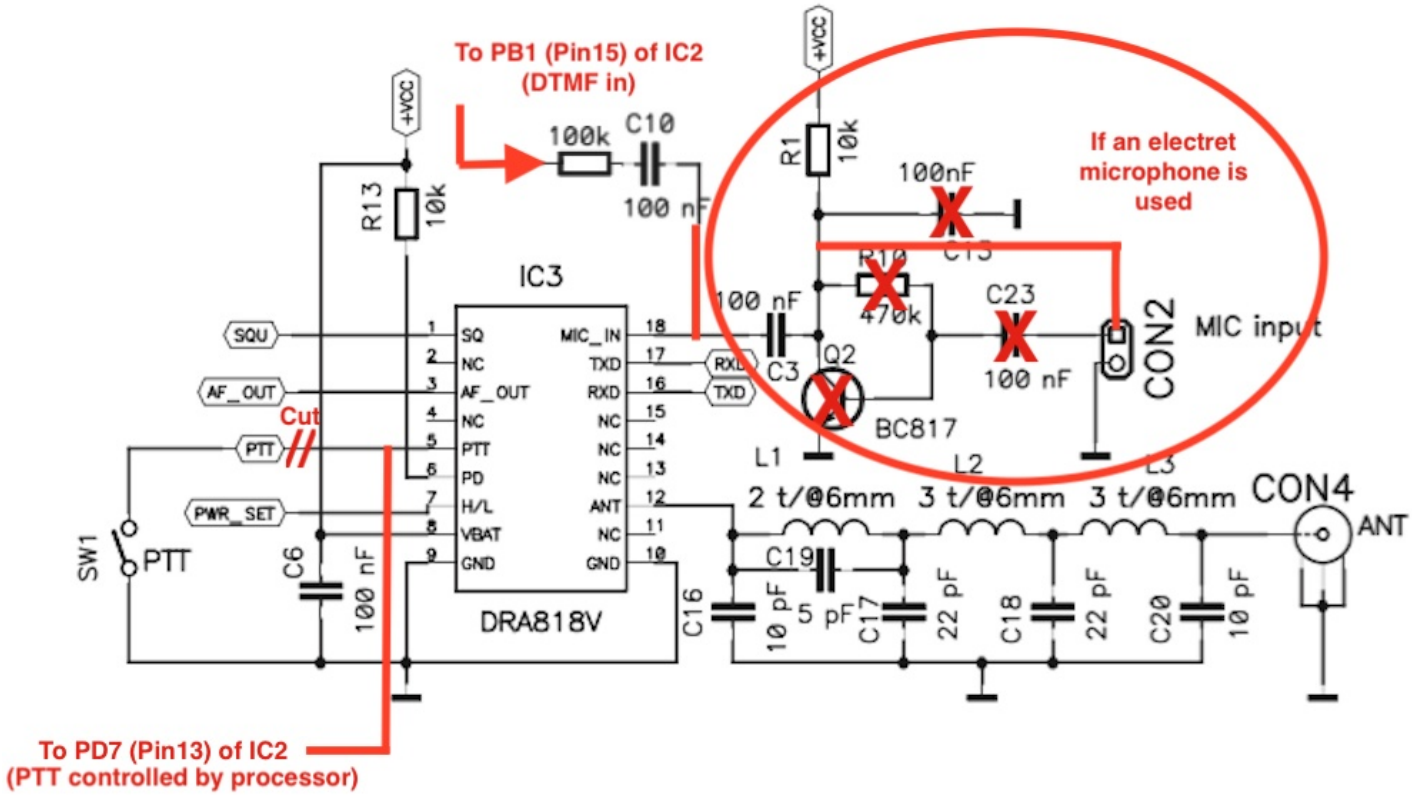
Aansluitingen van het Nokia display

Omdat ik een complete microfoon met een elektret element had liggen, heb ik R10, Q2, C13 en C23 niet gemonteerd. Het microfoon-signaal wordt dan rechtstreeks aangeboden op het punt tussen R1 en C3. Pas op: sluit hier geen gewoon microfoonelement op aan want via R1 staat er DC op dit punt. De 7805 heb ik niet op de print neergelegd. Omdat de module bij het zenden al bijna 1 ampere nodig heeft bij 5 volt en de aangeboden spanning uit een standaard voeding 12 Volt is, wordt de dissipatie in de 7805 fors. Daarom is het beter deze te voorzien van een koelplaat. Uiteindelijk vond ik bij langdurig zenden de 7805 nog steeds flink warm worden, dus heb ik in de 12 volt aanvoer 4 1N4004 diodes in serie gezet. Hierdoor zakt de spanning 3 volt, hetgeen het te dissiperen vermogen in de 7805 dusdanig verminderde dat deze nu alleen nog maar lekker warm wordt. Een andere oplossing is natuurlijk om in plaats van 12 volt een 9 volt adapter te gebruiken. Denk er wel om dat die minstens 2 ampère moet kunnen leveren. Omwille van later beschreven software aanpassingen heb ik een aantal modificaties aan de print gedaan. In het orginele ontwerp wordt de PTT van de DRA rechtstreeks door de PTT schakelaar van de microfoon bediend. Dit vond niet ideaal omdat ik dan softwarematig geen controle over de zendstatus van de transceiver zou hebben. Dus ik heb de PTT van de DRA aangesloten op pin 13 van de processor. De tweede aanpassing is de moge-lijkheid om DTMF tonen te kunnen versturen. Waarom en hoe staat verder beschreven, maar de hardware aanpassing is het verbinden van pin 15 van de processor via een weerstand van

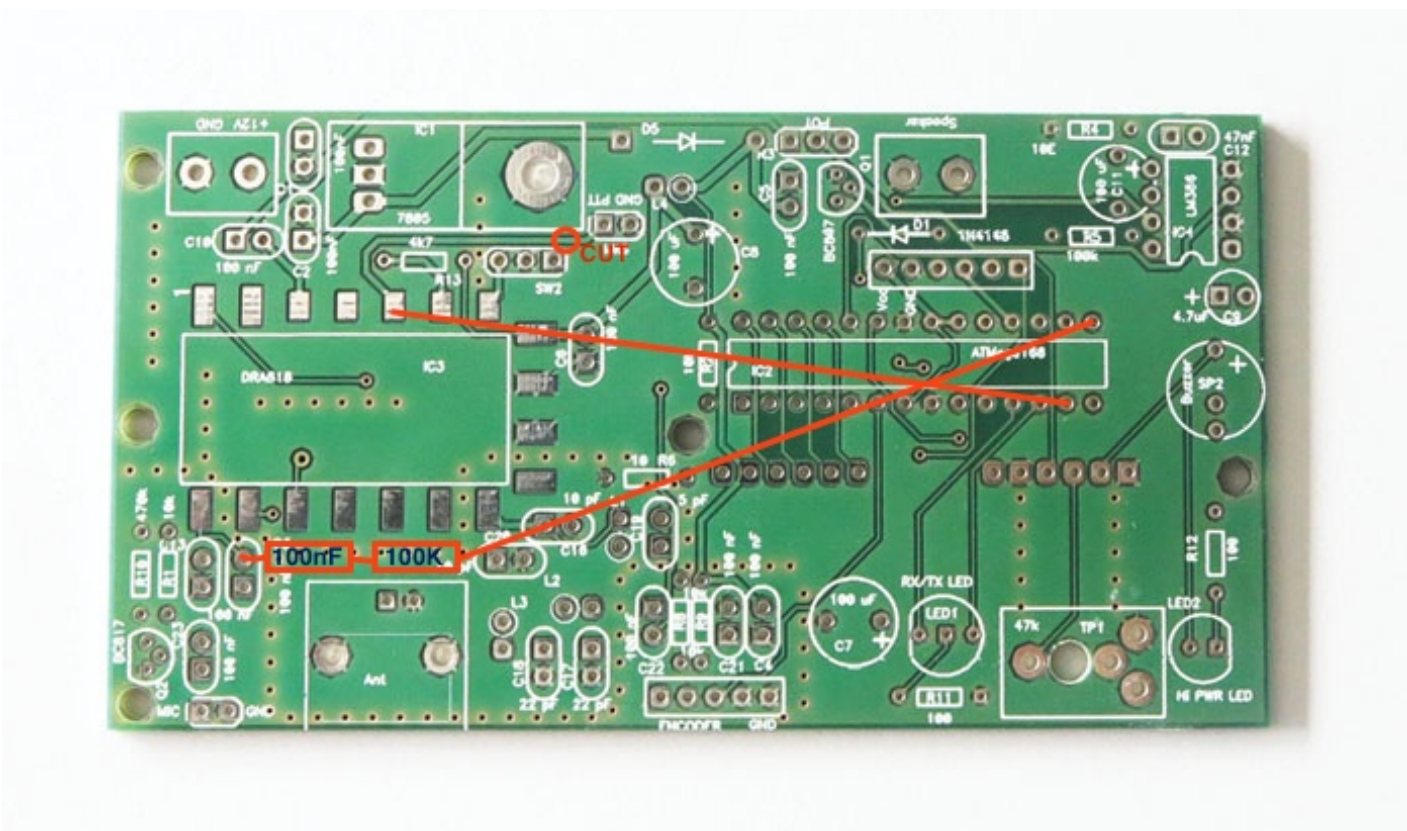


100K in serie met een C van 100nF met de microfooningang van de DRA.

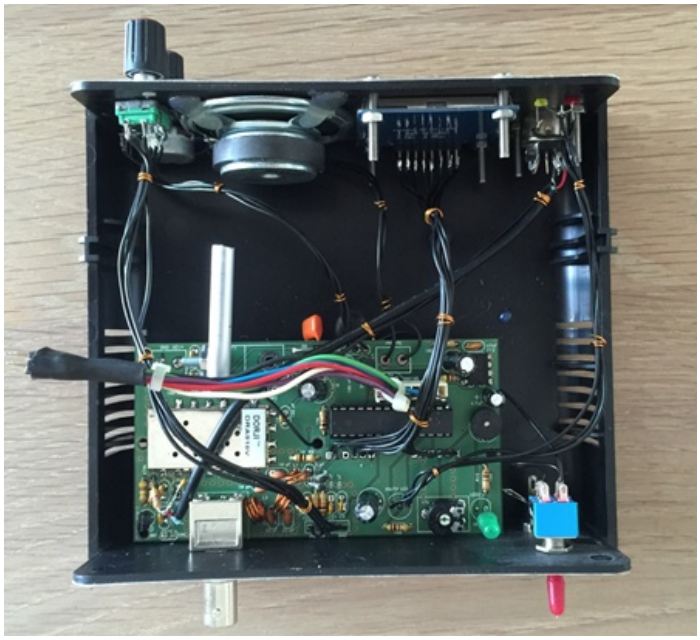




Aanpassingen aan het microfooncircuit, inclusief CTCSS injectie



Aanpassingen op de print.

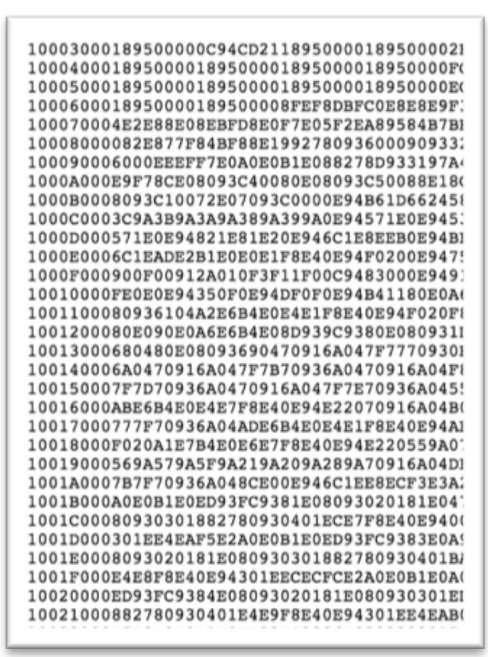


De transceiver in het kastje

De software

Alvorens de diepte in te gaan eerst iets over de ontwikkeling van software. Computers, dus ook microprocessoren, want een computer is niets anders dan een (hele dikke) microprocessor met de nodige randapparatuur, werken met enen en nullen. Dit heten bits. Deze bits worden geclusterd in blokken van 8, dit is een byte. In de grotere processoren worden deze bytes weer geclusterd in groepen van 2, 4 of 8 bytes (de laatste is een 64 bits processor). Om een processor te laten doen wat je wilt dat hij doet is het daarom noodzakelijk om de processor te programmeren met een verzameling bytes.

Omdat een byte bestaat uit 8 bits kunnen er in een byte dus maximaal 2 tot de 8ste =256 (= 0 tot 255) verschillende waardes worden opgeslagen. In ons dagelijks 10 tallig stelsel is dat geen handig aantal, vandaar dat men besloten heeft deze bytes uit te schijven in een 16 tallig stelsel: dit noemen we het hexadecimaal stelsel. Hierbij tellen we 0-9 en vervolgens A-F, dat zijn er samen 16 (0 – 15). Het hexadecimaal getal 1A betekent dus 1x16+10 = 26 in het tientallig stelsel. Het hexadecimaal getal FF is dus decimaal 255 (15x16 +15). Een byte kan op deze manier worden opgeschreven met 2 hexadecimale cijfers, waarbij er decimaal 3 cijfers nodig zijn. Een bijkomend technisch voordeel is dat een hexadecimaal cijfer een halve byte (4 bits) beschrijft, we noemen dit een nibble. Een programma dat we in een processor willen programmeren ziet er uit als een hele verzameling van hexadecimale getallen achter elkaar. Voor een normaal mens is het onmogelijk een programma te schrijven in deze hexadecimale waardes, vandaar dat er compilers zijn uitgevonden. Een compiler is een vertaler welke min of meer leesbare tekst converteert naar zo'n stroom hexadecimale waardes, wij noemen dat een HEX file. Deze leesbare tekst is natuurlijk geen spannend verhaal maar een heel gestructureerde verzameling instructies die de processor moet uitvoeren. We noemen dit een programmeertaal. Een programmeertaal is dus niet echt een programmeertaal maar een taal die als interface dient tussen de programmeur en de compiler. Afhankelijk van de processor die je wilt programmeren, het soort programma dat je wilt maken en de ervaring en voorkeur van de programmeur, bestaan er vele programmeertalen. De meest bekende daarvan is wellicht Basic, maar ook Pascal, Fortran, C en C# zijn bekende namen. De echte diehards programmeren in assembler, maar dat is niet echt iets voor de beginners.



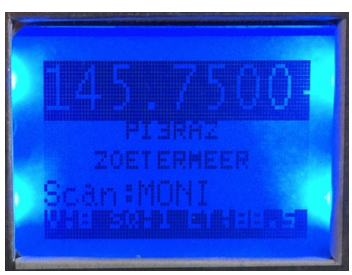
Een andere bekende taal is Arduino, hierbij denk je niet meteen aan een programmeertaal, maar dat is het wel. Arduino staat voor de combinatie van de hardware, de bekende Arduino boards, en de Arduino ontwikkelomgeving (IDE = Integrated Development Environment). Om het laagdrempelig te houden schrijf je geen programma's maar sketches, maar het is echt

programmeren.
Nadat je een programma hebt geschreven en gecompileerd tot een HEX file dient deze in de processor gebrand te worden. Dit gebeurt niet met een soldeerbout maar met een programmer. Dit is bijvoorbeeld een USBTinyISP, voor een paar dollar te koop op EBay.

Met een stukje Windows (of MAC hi) software (bijvoorbeeld AVRDude) kun je de HEX file in de processor branden.
Met de programmer kun je 'In Circuit Programmeren'. Dit wil zeggen dat de processor op de print kan blijven terwijl hij geprogrammeerd wordt. De programmer wordt dus aangesloten op de header CON3 op de print en op de USB poort van de PC.
Ook bij dit programmeerproces onderscheidt Arduino zich van een reguliere processor. Een Arduino processor is standaard uitgerust met een 'bootloader'. Dit is een stukje programma in de processor waarvan de taak is andere programma's te ontvangen via de USB poort en uit te voeren. Dit maakt het mogelijk om een Arduino rechtstreeks te programmeren via de USB en via diezelfde USB te communiceren met de processor zonder dat er een programmer en extra programma's noodzakelijk zijn. Ook dit verlaagt de drempel voor het gebruik van de Arduino.

Afijn, terug naar ons project. Op de website van Jurij staat het programma[6] dat hij heeft geschreven voor het project. Dit programma is geschreven in de taal Bascom. Dit lijkt op Basic maar is bedoeld voor het programmeren van Atmega processoren. Omdat ik geen Bascom compiler had en uiteraard snel wilde testen of een en ander werkte, heb ik aan Jurij de HEX file gevraagd en deze in de processor gebrand. Alles werkte, dus dat was mooi, maar ik was niet enthousiast over de software van Jurij. Niet dat er iets aan mankeert, maar het was gebaseerd op de Sloveense wijze van omgaan met de 2 meter band en de lokale repeaters.
Dus maar eens gaan kijken wat ik met de software kon. Ik vond een demoversie van Bascom[7] en die was in no time geïnstalleerd. Maar het programma van Jurij bleek te groot voor de demoversie van Bascom want deze is beperkt tot 4KB programmagrootte.
Een full versie van Bascom kost €90,-. Een hoop geld voor een projectje. Ik ben, gewapend met het voorbeeld van Jurij, maar gewoon opnieuw begonnen, maar al na een paar regels code was het programma weer te groot voor de demo versie van Bascom. Dit bleek aan de libraries te liggen. Oh ja, dat moet ik even uitleggen: een library is een stuk programma wat gebruikt kan worden door andere programma's. In dit geval betrof het de library die het display aanstuurt. Je wilt deze code niet steeds opnieuw schrijven, dus stop je het programma in een library en maakt het zo herbruikbaar. In de meeste gevallen zijn dit soort libraries al geschreven door echte programmeurs en beschikbaar gemaakt voor het publiek. De Bascom library voor het display en de library voor de RS232 communicatie met de DRA samen zijn samen al bijna 4KB groot, hetgeen het onmogelijk maakt een bruikbaar programma voor onze print te schrijven in de demo versie van Bascom.
Door Jurij werd ik gewezen op een project van Marko Sajovic, ook een Sloveen die voor de print van Jurij een oplossing had gebouwd met de Arduino ontwikkelomgeving.
Dit programma kreeg ik van Marko, maar toen ik het opende bleek er een hoop Sloveens in te staan; al het commentaar (hulpinformatie voor de programmeur) was in het Sloveens, maar erger was dat alle variabelen en constanten (als je gaat programmeren kom je er vanzelf achter wat dat zijn) ook in het Sloveens waren. Dat lijkt geen ramp en doet niets af aan het programma, maar het maakt het wel totaal onleesbaar voor

een niet-Sloveen, ik dus. Als je in de weer gaat met een programma van een ander, voel je je meestal al als een kat in een vreemd pakhuis. Gecombineerd met de taalbarriere was de lol er gauw af. Marko vond het echter wel leuk dat ik het plan had opgevat om met zijn project verder te gaan en verklaarde zich bereid om een en ander in het engels te vertalen. Na een paar dagen kreeg ik inderdaad van hem een redelijk leesbare versie. Deze kon ik na wat gesteggel compileren en in de processor branden en zowaar het werkte. Ik was er om verschillende redenen echter niet kapot van. Volgens mij door verkeerd gebruik van interrupts of door interrupt conflicten met de display library werkte het niet echt lekker soepel en daarbij kwam dat ik de indruk had dat het programma veel te ingewikkeld en daardoor slecht leesbaar was opgezet. Het is programmeurs eigen, goede software wordt alleen door jezelf gebouwd hi!



Goede raad was duur, het Arduino project opnieuw bouwen of de beurs trekken en Bascom kopen? Ik heb het laatste besloten om de volgende redenen; zoals al eerder geschreven, Arduino is laagdrempelig, de programmeertaal is eenvoudig, maar dat heeft als nadeel dat de compiler een hoop voor je gaat regelen en dus mogelijkheden afschermt. Arduino is leuk maar alleen voor overzichtelijke projectjes of als je het wilt mengen met bijvoorbeeld C. Deze beperking staat volgens mij ook aan de basis van de problemen die ik met het project van Marko had. Ik heb dus Bascom gekocht, dat had ik 15 jaar geleden ook al eens gedaan voor een toenmalig project, maar alle sporen daarvan waren uitgewist. Vervolgens heb ik een nieuw programma geschreven om de DRA aan te sturen en de nodige informatie op het display te zetten. Het programma van Jurij was daarbij een prima leidraad met bruikbare voorbeelden.

Na een avondje programmeren kon ik met een rotary encoder de DRA op elke frequentie zetten, de repeatershift en CTCSS codering regelen, omschakelen tussen RX en TX en werden de RX en TX ledjes aangestuurd zoals hoort.

Onderhand is het programma voorzien van de volgende functies:

- Frequentie instelbaar van 134 tot 174 MHz, omdat de DRA818V ook op 70cm kan luisteren is de frequentie ook instelbaar van 430-440 MHz. Maar stel je daar niet te veel van voor, zeker niet met het low-pass filter zoals op de print is bedoeld. Ik heb het niet opgesplitst in 2 banden, VHF en UHF, maar heb het programma zo gebouwd dat als je bij 174MHz bent aangekomen, de transceiver naar 430MHz springt. Aangekomen bij 440MHz springt die terug naar 134MHz. Bij het naar beneden draaien van de frequentie uiteraard andersom.
- 3 scanmodes: 1. Scannen tot er een draaggolf wordt gedetecteerd. 2. Scannen tot er een draaggolf wordt gedetecteerd en als de draaggolf verdwijnt gaat het scannen na 5 seconden verder en 3. Een monitor functie waarbij elke 5 seconden een frequentie wordt gecontroleerd op een draaggolf.
- Frequentie hopping, dit stuk is nog in ontwikkeling maar het zendgedeelte werkt. Het idee is als volgt: na het loslaten van de PTT stuurt de zender een DTMF signaal met daarin de nieuwe frequentie waarop wordt geluisterd. De bedoeling is dat het tegenstation mee schakelt met deze frequentiewijziging en vervolgens gaat uitzenden op de nieuwe frequentie. Als het tegenstation de uitzending beëindigt, zendt ook hij een DTMF signaal uit en bepaalt daarmee op zijn beurt de nieuwe frequentie. Als frequentie hopping aan staat wordt er een frequentie tussen de 144.5 en 145 MHz geselecteerd. Frequentie hopping begint altijd op 145.525 MHz: deze frequentie wordt automatisch geselecteerd als je frequentie hopping aan zet en de huidige frequentie buiten het stuk tussen 144.5 en 145MHz is.
- De stapgrootte kan worden ingesteld op 12.5, 25, 100 en 1000 kHz.
- Squelch instelbaar in 8 stappen.
- Het volume kan worden geregeld met de potmeter, maar de DRA heeft daarvoor ook een functie die vanuit het programma kan worden bediend. Ik ben echter niet gecharmeerd van

deze functie omdat het dynamisch bereik maar minimaal is.
- Het niveau van het microfoon signaal kan ook met een programmafunctie door de DRA worden ingesteld, maar hiervoor geldt hetzelfde bezwaar dat het dynamisch bereik slechts minimaal is.
- De TX en RX CTCSS codes zijn onafhankelijk van elkaar instelbaar.
- De minimale en maximale scanfrequentie zijn instelbaar (bandscannen), zo ook de monitorfrequentie voor het monitorscannen.
- De TX en RX frequentie, de minimale en maximale scanfrequentie, de TX en RX CTCSS frequenties, het squelch, volume en modulatieniveau, de standaard stapgrootte en de scanmode kunnen worden opgeslagen in de EEProm, zodat de transceiver altijd met de gewenste instellingen opstart. De standaard (fabrieks) instellingen kunnen ook worden terug gezet in de EEProm.
- In het programma heb ik een aantal standaard repeaters inclusief bijbehorende CTCSS codes geprogrammeerd. Zodra er een frequentie tussen 145.575 en 145.7875 wordt gekozen, worden de bijbehorende repeatergegevens getoond en wordt de CTCSS goed gezet. Vervolgens kunnen handmatig de CTCSS frequenties worden aangepast, maar bij het wijzigen van de frequentie worden weer de CTCSS frequenties van de geselecteerde repeater ingesteld.

Onderhand is het een leuke transceiver geworden die bij mij dagelijks aan staat om op de repeater mee te luisteren. Standaard start de set op 145.750 met monitor frequentie 145.575, zodat ik ook Utrecht in de gaten kan houden. Achterop is de aansluiting voor het programmeren van de processor naar buiten uitgevoerd, zodat ik kan door ontwikkelen aan het programma. De actuele versie van het programma is beschikbaar voor download[8]. Maar zoals beschreven: dit is een Bascom file en alleen te compileren met de volledige Bascom compiler. Voor wie wel zelf een programmer heeft en verder niet wil knutselen aan de software, is de HEX file beschikbaar[9]. Deze kan in een Atmega328 worden gebrand. Voor wie ook dit een brug te ver is, regel een Atmega328 en kom eens langs op de club, dan brand ik hem ter plaatse. Laat wel even weten van tevoren, zodat ik spullen kan meenemen.

Verdere plannen

Natuurlijk heb ik nog plannen, want er valt nog van alles te beleven met dit setje.
- In de huidige versie van het programma heb ik 19 repeaters geprogrammeerd maar er zijn er veel meer. Het plan is om alle repeaters in het programma te zetten en het zodanig aan te passen dat op alle repeaters afgestemd kan worden. Als er op een repeaterfrequentie meer repeaters beschikbaar zijn, wordt dan naar de volgende repeater in plaats van de volgende frequentie gestapt. Er ontstaat dan zoiets als 145.7375, 145.750 Zoetermeer, 145.750 Nijmegen, 145.750 Groningen, 145.7625 etc.
- Het grote font dat wordt gebruikt om de frequentie te tonen wil ik iets smaller maken, zodat ik achter de frequentie netjes de shift kan tonen. Nu is er net een half min teken zichtbaar omdat ik te weinig pixels beschikbaar heb.
- De 818V kan ook op 70cm luisteren, maar het low-pass filter op de print is hiervoor niet geschikt. Ik zoek nog een mooie oplossing om te kunnen schakelen tussen twee filters.
- Jurij is bezig met een print met twee modules, een 818V en een 818U. Hiermee kan een dualbander worden gebouwd. Die wil ik ook wel bouwen en daar de software geschikt voor maken.
- Frequentie hopping moet nog worden afgebouwd, DTMF decoderen is niet zo complex, maar de beschikbare IC’s hiervoor (bijvoorbeeld een MT8870) geven de tooncodes met 4 bits code en een senselijn. Dit betekent dat er op de processor 5 poorten beschikbaar moeten worden gemaakt, waarbij de senselijn aan een interrupt poort moet worden gehangen. Er moet worden uitgezocht of dat dit met het huidig hardware ontwerp mogelijk is. Anders zouden de 4 bits van de 8870 nog kunnen worden ‘geserialiseerd’, zodat er maar 2 in plaats van 5 pinnen nodig zijn, maar dan wordt het allemaal

zo uitgebreid.
- Ik speel ook nog met de gedachte om het programma in de Arduino IDE te schrijven. Dit heeft ook wel wat voeten in de aarde omdat een standaard Arduino UNO board gebruik maakt van een Atmega328P met een extern Xtal en de print van Jurij is bedoeld voor een Atmega328 (zonder P) zonder extern Xtal. De Arduino IDE snapt de layout van Jurij dan ook niet zomaar. Dit is in de Arduino IDE wel aan te passen met wat hacken, maar dan is meteen het voordeel van de laagdrempeligheid van Arduino verdwenen.
Kortom hou de website en volgende Razzies in de gaten voor de updates van dit project.

Handleiding

De bediening van de transceiver is heel eenvoudig en beperkt zich tot het gebruik van de rotary encoder. Na het tonen van het opstartscherm, wordt het volgende scherm getoond:

Boven de frequentie (reverse), daaronder de naam van de repeater, als het een repeaterfrequentie betreft. Daaronder de plaats van de repeater en daaronder de geselecteerde scanmode. Helemaal onderaan in reverse V:8 SQ:1 CT:88.5 Dit wil zeggen het volume staat op niveau 8, de squech staat op 1 en de CTCSS TX frequentie is 88.5Hz.
Het is nu mogelijk de frequentie aan te passen door te draaien aan de rotary encoder. Door op de rotary encoder te drukken stap je door het menu. Het geselecteerde menu-item wordt getoond op de een-na-onderste regel van het scherm. Als het menu actief is dan wordt de frequentie niet meer reverse getoond maar wordt de menuregel reverse getoond. Door kort op de PTT te drukken ga je terug naar de frequentiemode en worden de aanpassingen in het menu geëffectueerd.

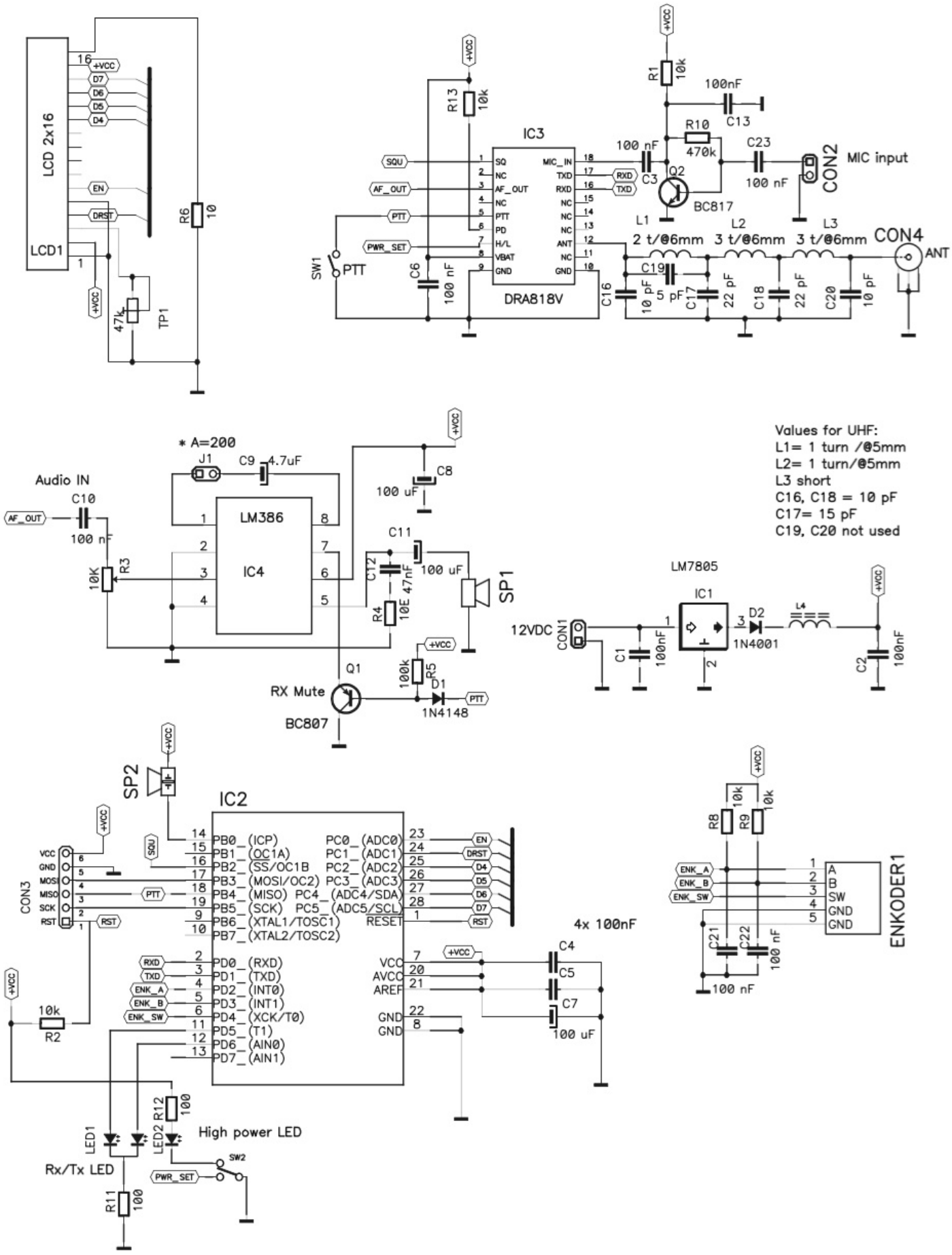
Menu 1. Dit is het scanmenu. Door te draaien aan de rotary encoder kan gekozen worden tussen ‘OFF’ = niet scannen, ’ON’ = scannen tot er een draaggolf wordt waargenomen, ’AUTO’ = scannen tot er een draaggolf wordt waargenomen en als de draaggolf verdwijnt dan wordt het scannen na 5 seconden hervat en ‘MONI’ = hierbij wordt elke 5 seconden de ingestelde monitorfrequentie gecontroleerd. Als er een draaggolf op de monitorfrequentie staat dan blijft de transceiver hierop staan zolang de draaggolf er is. Door de PTT van de microfoon kort in te drukken wordt er terug gesprongen naar frequentiemode en start het scannen. Het scannen wordt beëindigd door de PTT in te drukken.

Menu 2. Dit is de frequentiehopping mode. Als de huidige frequentie buiten de band 144.5 – 145.0MHz valt, dan wordt automatisch de frequentie 145.525 geselecteerd. Na het loslaten van de PTT wordt er een DTMF code uitgezonden en sprint de transceiver naar een andere willekeurige frequentie tussen de 144.5 en 145.0

Schema

Op de volgende bladzijde vind je het schema van de complete transceiver. Hierop zijn de bovenbeschreven wijzigingen nog niet aangegeven (microfoon circuit, dioden vóór de spanningsregelaar.)

[1] http://bit.ly/1LA1x7a
[2] http://bit.ly/1Kq2ksE
[3] http://bit.ly/1fKBtdL
[4] http://bit.ly/1MZYAQj
[5] http://bit.ly/1MZYGr6
[6] http://bit.ly/1hJw9JH
[7] http://bit.ly/1KSbudc
[8] http://pi4raz.nl/download/RDK_DRA_RADIO.zip
[9] http://pi4raz.nl/download/RDK_DRA818_HEX.zip


LCD 2x16
LCD1
+VCC
D7
D6
D5
D4
EN
DRST
R6
10
47k
TP1
IC3
DRA818V
SQU
AF_OUT
PTT
PWR_SET
SW1
PTT
C6
100 nF
R13
10k
R1
10k
100nF
C13
R10
470k
C23
100 nF
CON2
MIC input
100 nF
C3
Q2
BC817
L1
L2
L3
2 t/@6mm
3 t/@6mm
3 t/@6mm
CON4
ANT
C19
10 pF
5 pF
C17
22 pF
C18
22 pF
C20
10 pF
C16
* A=200
J1
C9
4.7uF
Audio IN
C10
100 nF
AF_OUT
10K
R3
LM386
IC4
C8
100 uF
C11
C12
47nF
100 uF
R4
10E
SP1
Q1
RX Mute
BC807
100k
R5
D1
1N4148
PTT
Values for UHF:
L1= 1 turn /@5mm
L2= 1 turn/@5mm
L3 short
C16, C18 = 10 pF
C17= 15 pF
C19, C20 not used
LM7805
IC1
12VDC
CON1
C1
100nF
1N4001
D2
L4
C2
100nF
SP2
IC2
CON3
VCC
GND
MOSI
MISO
SCK
RST
PB0_(ICP)
PB1_(OC1A)
PB2_(SS/OC1B
PB3_(MOSI/OC2)
PB4_(MISO)
PB5_(SCK)
PB6_(XTAL1/TOSC1)
PB7_(XTAL2/TOSC2)
PC0_(ADC0)
PC1_(ADC1)
PC2_(ADC2)
PC3_(ADC3)
PC4_(ADC4/SDA)
PC5_(ADC5/SCL)
RESET
EN
DRST
D4
D5
D6
D7
PD0_(RXD)
PD1_(TXD)
PD2_(INT0)
PD3_(INT1)
PD4_(XCK/T0)
PD5_(T1)
PD6_(AIN0)
PD7_(AIN1)
VCC
AVCC
AREF
GND
GND
4x 100nF
C4
C5
C7
100 uF
10k
R2
R12
100
LED1
LED2
High power LED
Rx/Tx LED
SW2
PWR_SET
R11
100
R8
10k
R9
10k
ENK_A
ENK_B
ENK_SW
A
B
SW
GND
GND
ENKODER1
C21
C22
100 nF
100 nF