

# De DDS VFO software

---

EEN BESCHRIJVING VAN DE SOFTWARE NODIG OM EEN DDS VFO TE  
BESTUREN VANUIT EEN ARDUINO, MET ALS OPTIE SPRAAK

# Waarom een DDS VFO

Een stabiele, bruikbare VFO met goede frequentie uitlezing is bijna altijd het zwakke punt bij een zelfbouw project.

Dat zorgde voor mijn eerste ontwerp; in eerste instantie bij Surplus zenders.

Later ook toegepast in het 'Superwiel' en andere projecten.

# Wat is eigenlijk een DDS

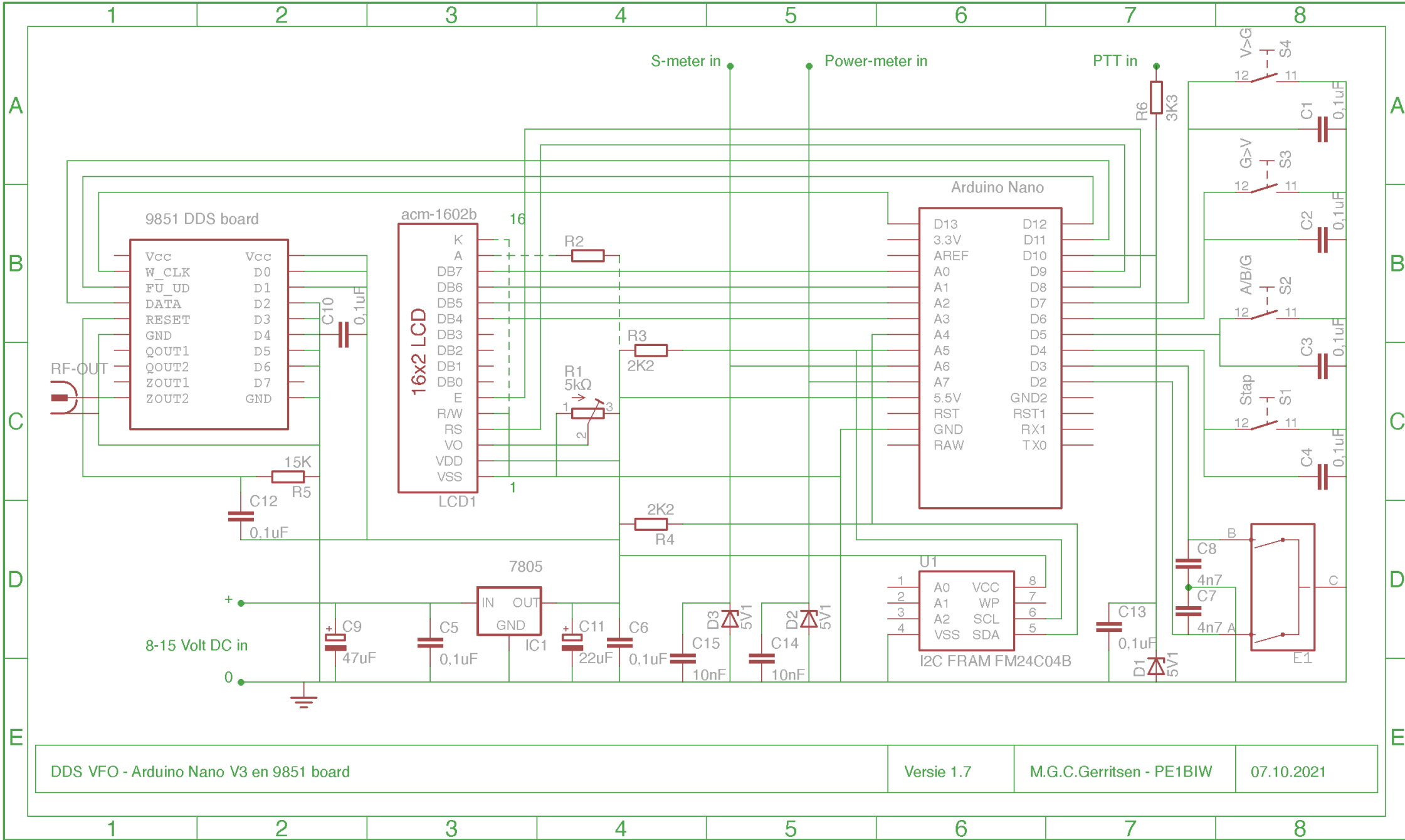
- Frequentieopwekking door **D**irecte **D**igitale **S**ynthese
- De vorm van het signaal opgeslagen als waarden (sinus)
- Een (eventueel extern vergrendelde) referentie oscillator
- Een digitaal analoog omzetter
- Een laag doorlaat filter

# De voorwaarden

- ‘Kristal stabiel’
- Instelbaar over ten minste het HF gebied
- Frequentie stappen instelbaar (10Hz, 100Hz, 1kHz, etc.)
- Geheugen voor laatste toestand na uitschakelen
- Geheugen opslag voor vaak gebruikte frequenties
- Eventueel een MF offset
- ‘Schoon’ uitgangssignaal

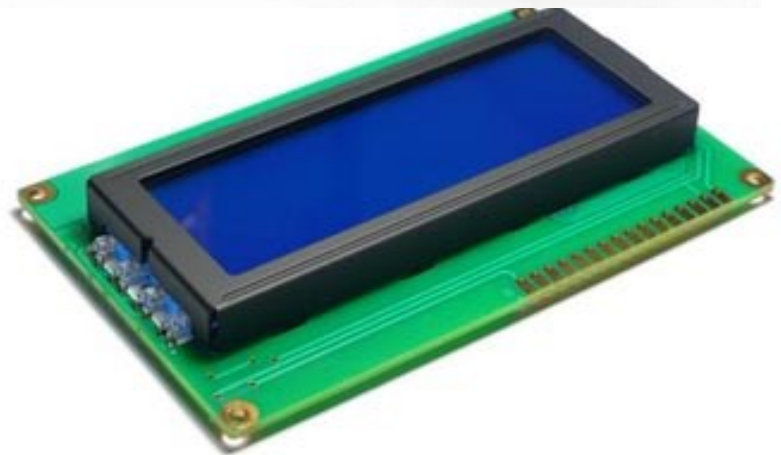
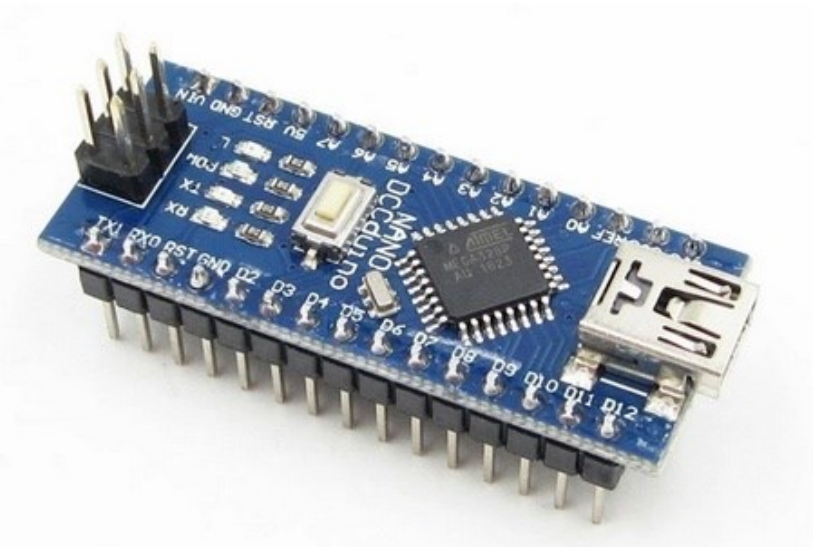
# De onderdelen van het VFO

- een microcontroller (Arduino) om de DDS te besturen
- de DDS chip met referentie oscillator
- draai- en drukknoppen om het geheel te bedienen
- een display om de instellingen weer te geven
- (spraakuitgave)



DDS VFO - Arduino Nano V3 en 9851 board Versie 1.7 M.G.C.Gerritsen - PE1BIW 07.10.2021

Zo zien de onderdelen er (bijvoorbeeld) uit



# Wat zit er (niet) in zo'n Arduino

- een processor
- werk geheugen
- programma opslag geheugen
- EEPROM
- digitale- en analoge in- en uitgangen (pins)
- een USB aansluiting
- I<sup>2</sup>C en SPI bus
- geen geheugen dat vaak gelezen en geschreven kan worden; daarom dus een externe FRAM chip die  $10^{14}$  keer gelezen en geschreven kan worden



# Hoe te programmeren

## Dit is geen cursus Arduino programmeren

- het programma wordt via de USB aansluiting naar de Arduino gestuurd (serieel)
- het programma (sketch) wordt geschreven in een C dialect
- om het programma te schrijven wordt een zgn. IDE gebruikt, een speciale tekstverwerker voor programmeren
- in de IDE wordt het programma gecompileerd en vervolgens (over USB) naar de Arduino gestuurd

# De Arduino IDE

```

drake_tr7_ser_to_speech_118 | Arduino 1.8.19
drake_tr7_ser_to_speech_118
77 int oldStore = -1;
78 byte i2c_stat;
79 struct store
80 {
81     long a_freq;
82     long b_freq;
83     long c_step;
84     int c_mem;
85     int c_mode;
86 }fram_store;
87
88 bool freq_changed = false;
89 long loop_count = 0;
90
91
92 // voorbereiden FRAM
93 extEEPROM myFRAM(kbits_4, 1, 256, 0x50);
94
95 // voorbereiden LCD
96 //           RS EN D4 D5 D6 D7
97 LiquidCrystal lcd(9, 8, A0, A1, A2, A3);
98
99 void setup()
100 {
101     Serial.begin(19200);
102     // encoder en drukknop pennen als ingang instellen
103     pinMode(encoderPin1, INPUT_PULLUP);
104     pinMode(encoderPin2, INPUT_PULLUP);
105     pinMode(buttonStep, INPUT_PULLUP);
106     pinMode(buttonMode, INPUT_PULLUP);
107     pinMode(buttonMemToVfo, INPUT_PULLUP);
108     pinMode(buttonG, INPUT_PULLUP);
109
110     //
111     //
112     //
113     //
114     //
115     //
116     //
117     //
118     //
119     //
120     //
121     //
122     //
123     //
124     //
125     //
126     //
127     //
128     //
129     //
130     //
131     //
132     //
133     //
134     //
135     //
136     //
137     //
138     //
139     //
140     //
141     //
142     //
143     //
144     //
145     //
146     //
147     //
148     //
149     //
150     //
151     //
152     //
153     //
154     //
155     //
156     //
157     //
158     //
159     //
160     //
161     //
162     //
163     //
164     //
165     //
166     //
167     //
168     //
169     //
170     //
171     //
172     //
173     //
174     //
175     //
176     //
177     //
178     //
179     //
180     //
181     //
182     //
183     //
184     //
185     //
186     //
187     //
188     //
189     //
190     //
191     //
192     //
193     //
194     //
195     //
196     //
197     //
198     //
199     //
200     //
201     //
202     //
203     //
204     //
205     //
206     //
207     //
208     //
209     //
210     //
211     //
212     //
213     //
214     //
215     //
216     //
217     //
218     //
219     //
220     //
221     //
222     //
223     //
224     //
225     //
226     //
227     //
228     //
229     //
230     //
231     //
232     //
233     //
234     //
235     //
236     //
237     //
238     //
239     //
240     //
241     //
242     //
243     //
244     //
245     //
246     //
247     //
248     //
249     //
250     //
251     //
252     //
253     //
254     //
255     //
256     //
257     //
258     //
259     //
260     //
261     //
262     //
263     //
264     //
265     //
266     //
267     //
268     //
269     //
270     //
271     //
272     //
273     //
274     //
275     //
276     //
277     //
278     //
279     //
280     //
281     //
282     //
283     //
284     //
285     //
286     //
287     //
288     //
289     //
290     //
291     //
292     //
293     //
294     //
295     //
296     //
297     //
298     //
299     //
300     //
301     //
302     //
303     //
304     //
305     //
306     //
307     //
308     //
309     //
310     //
311     //
312     //
313     //
314     //
315     //
316     //
317     //
318     //
319     //
320     //
321     //
322     //
323     //
324     //
325     //
326     //
327     //
328     //
329     //
330     //
331     //
332     //
333     //
334     //
335     //
336     //
337     //
338     //
339     //
340     //
341     //
342     //
343     //
344     //
345     //
346     //
347     //
348     //
349     //
350     //
351     //
352     //
353     //
354     //
355     //
356     //
357     //
358     //
359     //
360     //
361     //
362     //
363     //
364     //
365     //
366     //
367     //
368     //
369     //
370     //
371     //
372     //
373     //
374     //
375     //
376     //
377     //
378     //
379     //
380     //
381     //
382     //
383     //
384     //
385     //
386     //
387     //
388     //
389     //
390     //
391     //
392     //
393     //
394     //
395     //
396     //
397     //
398     //
399     //
400     //
401     //
402     //
403     //
404     //
405     //
406     //
407     //
408     //
409     //
410     //
411     //
412     //
413     //
414     //
415     //
416     //
417     //
418     //
419     //
420     //
421     //
422     //
423     //
424     //
425     //
426     //
427     //
428     //
429     //
430     //
431     //
432     //
433     //
434     //
435     //
436     //
437     //
438     //
439     //
440     //
441     //
442     //
443     //
444     //
445     //
446     //
447     //
448     //
449     //
450     //
451     //
452     //
453     //
454     //
455     //
456     //
457     //
458     //
459     //
460     //
461     //
462     //
463     //
464     //
465     //
466     //
467     //
468     //
469     //
470     //
471     //
472     //
473     //
474     //
475     //
476     //
477     //
478     //
479     //
480     //
481     //
482     //
483     //
484     //
485     //
486     //
487     //
488     //
489     //
490     //
491     //
492     //
493     //
494     //
495     //
496     //
497     //
498     //
499     //
500     //
501     //
502     //
503     //
504     //
505     //
506     //
507     //
508     //
509     //
510     //
511     //
512     //
513     //
514     //
515     //
516     //
517     //
518     //
519     //
520     //
521     //
522     //
523     //
524     //
525     //
526     //
527     //
528     //
529     //
530     //
531     //
532     //
533     //
534     //
535     //
536     //
537     //
538     //
539     //
540     //
541     //
542     //
543     //
544     //
545     //
546     //
547     //
548     //
549     //
550     //
551     //
552     //
553     //
554     //
555     //
556     //
557     //
558     //
559     //
560     //
561     //
562     //
563     //
564     //
565     //
566     //
567     //
568     //
569     //
570     //
571     //
572     //
573     //
574     //
575     //
576     //
577     //
578     //
579     //
580     //
581     //
582     //
583     //
584     //
585     //
586     //
587     //
588     //
589     //
590     //
591     //
592     //
593     //
594     //
595     //
596     //
597     //
598     //
599     //
600     //
601     //
602     //
603     //
604     //
605     //
606     //
607     //
608     //
609     //
610     //
611     //
612     //
613     //
614     //
615     //
616     //
617     //
618     //
619     //
620     //
621     //
622     //
623     //
624     //
625     //
626     //
627     //
628     //
629     //
630     //
631     //
632     //
633     //
634     //
635     //
636     //
637     //
638     //
639     //
640     //
641     //
642     //
643     //
644     //
645     //
646     //
647     //
648     //
649     //
650     //
651     //
652     //
653     //
654     //
655     //
656     //
657     //
658     //
659     //
660     //
661     //
662     //
663     //
664     //
665     //
666     //
667     //
668     //
669     //
670     //
671     //
672     //
673     //
674     //
675     //
676     //
677     //
678     //
679     //
680     //
681     //
682     //
683     //
684     //
685     //
686     //
687     //
688     //
689     //
690     //
691     //
692     //
693     //
694     //
695     //
696     //
697     //
698     //
699     //
700     //
701     //
702     //
703     //
704     //
705     //
706     //
707     //
708     //
709     //
710     //
711     //
712     //
713     //
714     //
715     //
716     //
717     //
718     //
719     //
720     //
721     //
722     //
723     //
724     //
725     //
726     //
727     //
728     //
729     //
730     //
731     //
732     //
733     //
734     //
735     //
736     //
737     //
738     //
739     //
740     //
741     //
742     //
743     //
744     //
745     //
746     //
747     //
748     //
749     //
750     //
751     //
752     //
753     //
754     //
755     //
756     //
757     //
758     //
759     //
760     //
761     //
762     //
763     //
764     //
765     //
766     //
767     //
768     //
769     //
770     //
771     //
772     //
773     //
774     //
775     //
776     //
777     //
778     //
779     //
780     //
781     //
782     //
783     //
784     //
785     //
786     //
787     //
788     //
789     //
790     //
791     //
792     //
793     //
794     //
795     //
796     //
797     //
798     //
799     //
800     //
801     //
802     //
803     //
804     //
805     //
806     //
807     //
808     //
809     //
810     //
811     //
812     //
813     //
814     //
815     //
816     //
817     //
818     //
819     //
820     //
821     //
822     //
823     //
824     //
825     //
826     //
827     //
828     //
829     //
830     //
831     //
832     //
833     //
834     //
835     //
836     //
837     //
838     //
839     //
840     //
841     //
842     //
843     //
844     //
845     //
846     //
847     //
848     //
849     //
850     //
851     //
852     //
853     //
854     //
855     //
856     //
857     //
858     //
859     //
860     //
861     //
862     //
863     //
864     //
865     //
866     //
867     //
868     //
869     //
870     //
871     //
872     //
873     //
874     //
875     //
876     //
877     //
878     //
879     //
880     //
881     //
882     //
883     //
884     //
885     //
886     //
887     //
888     //
889     //
890     //
891     //
892     //
893     //
894     //
895     //
896     //
897     //
898     //
899     //
900     //
901     //
902     //
903     //
904     //
905     //
906     //
907     //
908     //
909     //
910     //
911     //
912     //
913     //
914     //
915     //
916     //
917     //
918     //
919     //
920     //
921     //
922     //
923     //
924     //
925     //
926     //
927     //
928     //
929     //
930     //
931     //
932     //
933     //
934     //
935     //
936     //
937     //
938     //
939     //
940     //
941     //
942     //
943     //
944     //
945     //
946     //
947     //
948     //
949     //
950     //
951     //
952     //
953     //
954     //
955     //
956     //
957     //
958     //
959     //
960     //
961     //
962     //
963     //
964     //
965     //
966     //
967     //
968     //
969     //
970     //
971     //
972     //
973     //
974     //
975     //
976     //
977     //
978     //
979     //
980     //
981     //
982     //
983     //
984     //
985     //
986     //
987     //
988     //
989     //
990     //
991     //
992     //
993     //
994     //
995     //
996     //
997     //
998     //
999     //
1000    //

```

# Wat zijn de onderdelen van de code

- commentaar (versie informatie)
- de gebruikte bibliotheken (`#include` libraries)
- definitie van constantes (`#define`)
- definitie van globale variabelen (`bool`, `int`, `long`, `struct`)
- de eenmalige instellingen: `setup()`
- de programma lus: `loop()`
- functies: `functienaam()`
- interrupt functie(s)

# Versie informatie

```
1 // Een 9850 DDS VFO
2 // v1.09 04.08.2021 M.G.C. Gerritsen / H. van Rees - PE1BIW / PA0VRE
3 // v1.10 28.11.2022 MG toevoeging pin 10 en code om frequentie serieel te versturen
4 // v1.11 29.11.2022 MG aanpassing string voor frequentie over serieel, nu 'Fxxxxxxx\r\n'
5 // V1.12 02.12.2022 MG toevoeging uitvoer VFO, geheugen en stapgrootte
6 // V1.13 05.12.2022 MG gescheiden vfo+frequentie en step
7 // V1.14 08.12.2022 MG ptt stuurt achter elkaar vfo frequentie en stap, stap stuurt stap
8 // V1.15 10.12.2022 MG extra functies voor geheugen opslag, etc.
9 // V1.16 11.12.2022 MG aanpassing voor debugging
10 // V1.18 17.01.2023 MG toevoeging eenmalige uitgave spraak frequentie na verandering frequentie
11 // V1.18 18.01.2023 MG toevoeging spraak commando voor 'geen stap in geheugen mode'
12
```

# Bibliotheken (#include)

```
23  
24 #include <LiquidCrystal.h>  
25 #include <EEPROM.h>  
26 #include <Wire.h>  
27 #include <extEEPROM.h>  
28
```

# Constance waarden (#define)

```
35
36 #define MIN_FREQ    5050.e03    // lage grens 5.05MHz
37 #define MAX_FREQ    5550.e03    // hoge grens 5.55MHz
38 #define MF_OFFSET   -5050.e03    // middenfrequent offset voor display, bijv. -5.05.e06
39
40 #define OFFSET 0                // pas dit aan voor oscillator afwijking
41
42 #define VFO_A 0
43 #define VFO_B 1
44 #define MEM    2
45 #define STORE  3
46 #define STORED -1
47 #define CANCEL -2
```

# Constance waarden (#define)

```
50  
51 // pen namen  
52 #define encoderPin1 2  
53 #define encoderPin2 3  
54 #define buttonStep 4  
55 #define buttonMode 5  
56 #define buttonMemToVfo 6  
57 #define buttonStore 7  
58  
59 #define buttonSay 10  
60  
61 #define ddsPinData 11  
62 #define ddsPinFq_Ud 12  
63 #define ddsPinW_Clk 13  
64
```

# Globale variabelen

```
70 long currentStep = 1000;
71 int lastEncoded = 0;
72 int currentMode = VFO_A;
73 int storeMode;
74 int currentMem = 0;
75 int oldMem = -1;
76 int currentStore = 0;
77 int oldStore = -1;
78 byte i2c_stat;
79 struct store
80 {
81     long a_freq;
82     long b_freq;
83     long c_step;
84     int c_mem;
85     int c_mode;
86 }fram_store;
87
```



# setup()

```
99 void setup()
100 {
101   Serial.begin(19200);
102   // encoder en drukknop pennen als ingang instellen
103   pinMode(encoderPin1, INPUT_PULLUP);
104   pinMode(encoderPin2, INPUT_PULLUP);
105   pinMode(buttonStep, INPUT_PULLUP);
106   pinMode(buttonMode, INPUT_PULLUP);
107   pinMode(buttonMemToVfo, INPUT_PULLUP);
108   pinMode(buttonStore, INPUT_PULLUP);
109   pinMode(buttonSay, INPUT_PULLUP);
110
111   // interrupts voor de encoder
112   attachInterrupt(digitalPinToInterrupt(2), updateEncoder, CHANGE);
113   attachInterrupt(digitalPinToInterrupt(3), updateEncoder, CHANGE);
114
115   // pennen naar de DDS print
116   pinMode(ddsPinW_Clk, OUTPUT);
117   pinMode(ddsPinFq_Ud, OUTPUT);
118   pinMode(ddsPinData, OUTPUT);
119 }
```

# loop()

Dit is de hoofdprogramma lus; de Arduino blijft deze eeuwig uitvoeren.

Wat zit er in die loop()?

- controleer of de frequentie of geheugenpositie veranderd is en onderneem actie
- controleer of een knop gedrukt is en onderneem actie
- als de frequentie veranderd is, spreek die dan na een bepaalde tijd uit

# Loop()

Hier wordt aan de hand van de mode, als de frequentie verandert is, de nieuwe frequentie ingesteld:

```
174 void loop()
175 {
176     switch (currentMode)
177     {
178         // interrupt gestuurde frequentie zonodig aanpassen
179         case VFO_A:
180             if (currentFreqA != oldFreq)
181             {
182                 ddsSetFrequency((double)currentFreqA);
183                 showFreq(VFO_A, 0, currentFreqA);
184                 oldFreq = currentFreqA;
185                 writeFRAM();
186                 freq_changed = true;
187             }
188             break;
189         case VFO_B:
```

# Loop()

Hier wordt, als de frequentie verandert is, de frequentie na verloop van tijd uitgesproken:

```
219 | // eenmalige spraakuitgave na frequentie- of geheugenverandering
220 | if (freq_changed)
221 | {
222 |   if(loop_count++ == 60000)
223 |   {
224 |     freq_changed = false;
225 |     loop_count = 0;
226 |     if (currentMode == MEM)
227 |       sayStore(currentMem);
228 |     sayFreq(currentMode);
229 |   }
230 | }
```

# Loop()

Hier worden de druktoetsen afgevraagd en, als ze ingedrukt zijn, de desbetreffende functie aangeroepen:

```
232 //afvragen van de Stap knop
233 if (digitalRead(buttonStep) == false) updateStepsize();
234
235 // afvragen van de Mode knop
236 if (digitalRead(buttonMode) == false) updateMode();
237
238 // afvragen van de Geheugen->VFO A/Afbreken knop
239 if (digitalRead(buttonMemToVfo) == false)
240 {
241     if (currentMode == STORE) cancelStore();
242     else moveMemToVfo();
243 }
244
```

# Functies

Een voorbeeld van een functie: de Mode knop (VFO A/B /Geheugen) wordt ingedrukt.

```
518 void updateMode()
519 {
520     switch (currentMode)
521     {
522         case VFO_A:
523             currentMode = VFO_B;
524             oldFreq = 0; // forceer update van VFO display
525             break;
526         case VFO_B:
527             currentMode = MEM;
528             oldMem = -1; // forceer update van VFO display
529             break;
530         case MEM:
531             currentMode = VFO_A;
532             oldFreq = 0; // forceer update van VFO display
533             break;
534     }
535     sayType(currentMode, currentMem);
536     delay(500);
537 }
```

# Interrupt functie

Een deel van de interrupt functie aangeroepen door verdraaien van de rotary encoder:

```
436 // --- rotary encoder functie aangeroepen door interrupts ---
437 void updateEncoder()
438 {
439     int MSB = digitalRead(encoderPin1);
440     int LSB = digitalRead(encoderPin2);
441
442     int encoded = (MSB << 1) | LSB;
443     int sum = (lastEncoded << 2) | encoded;
444
445     if (sum == 0b0111)
446     {
447         switch (currentMode)
448         {
449             case VFO_A:
450                 currentFreqA += currentStep;
451                 if (currentFreqA > MAX_FREQ) currentFreqA = MAX_FREQ;
452                 break;
453             case VFO_B:
454                 currentFreqB += currentStep;
455                 if (currentFreqB > MAX_FREQ) currentFreqB = MAX_FREQ;
456                 break;
```

# De DDS programmeer functie

Het instellen van de frequentie van de DDS d.m.v. een functie:

```
608
609 void ddsSetFrequency(double frequency)
610 {
611     // bereken het frequentie 32 bit woord en stuur naar de DDS byte voor byte
612     // en als laatste stuur de controle/fase byte (0x09)
613     unsigned long freq = (unsigned long)frequency * 4294967296.0 / (125.e06 + OFFSET);
614     for (int b = 0; b < 4; b++)
615     {
616         ddsSendByte(freq & 0xff);
617         freq >>= 8;
618     }
619     ddsSendByte(0x00);
620     ddsPulse(ddsPinFq_Ud);
621 }
```



# Optionele spraak uitgave

De serieel-over-USB interface van de Arduino wordt gebruikt om een stroom karakters naar een PC/Mac/Raspberry Pi te sturen.

Die karakterstromen hebben een afgesproken formaat en kunnen dus diverse, herkenbare 'boodschappen' sturen, zoals:

- frequentie
- stap-grootte
- VFO A, B of Geheugen
- etc.

# Optionele spraak uitgave

Aan de PC kant loopt een programma, dat luistert op de serieel-over-USB poort, de inkomende tekst stroom herkent en via een bibliotheek dit uitspreekt als een tekst.

Dit is een zogenaamd Python script.

# Dank

Dank U voor uw aandacht zover!

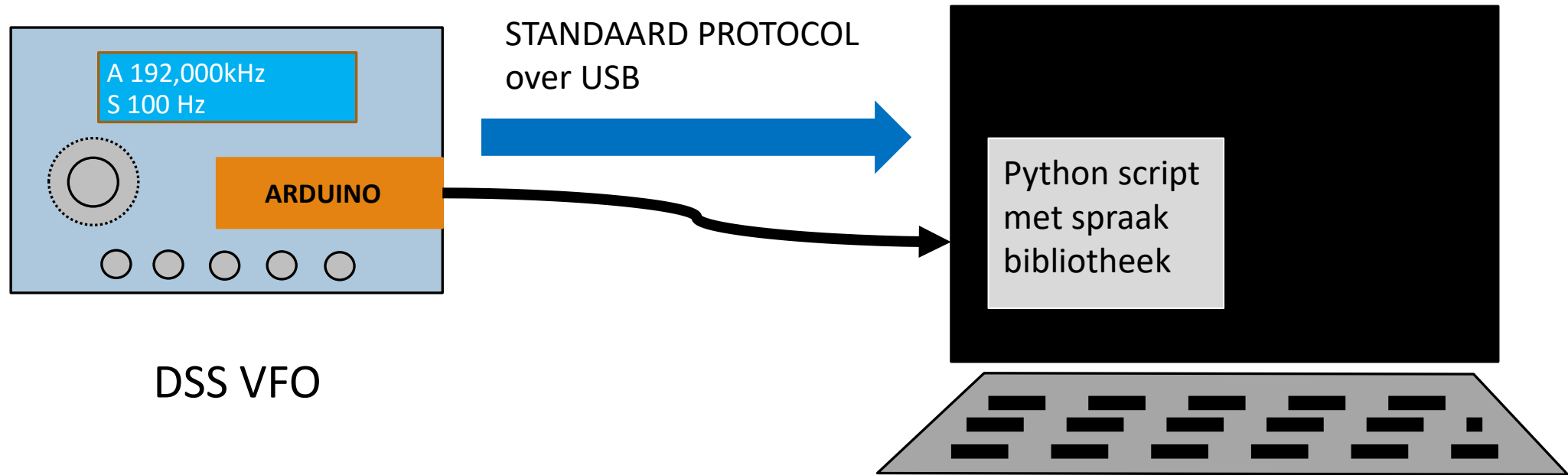
Vragen?

[PE1BIW@ZIGGO.NL](mailto:PE1BIW@ZIGGO.NL)

of

3705kHz AM ;-)

# Het prototype voor de spraaktests



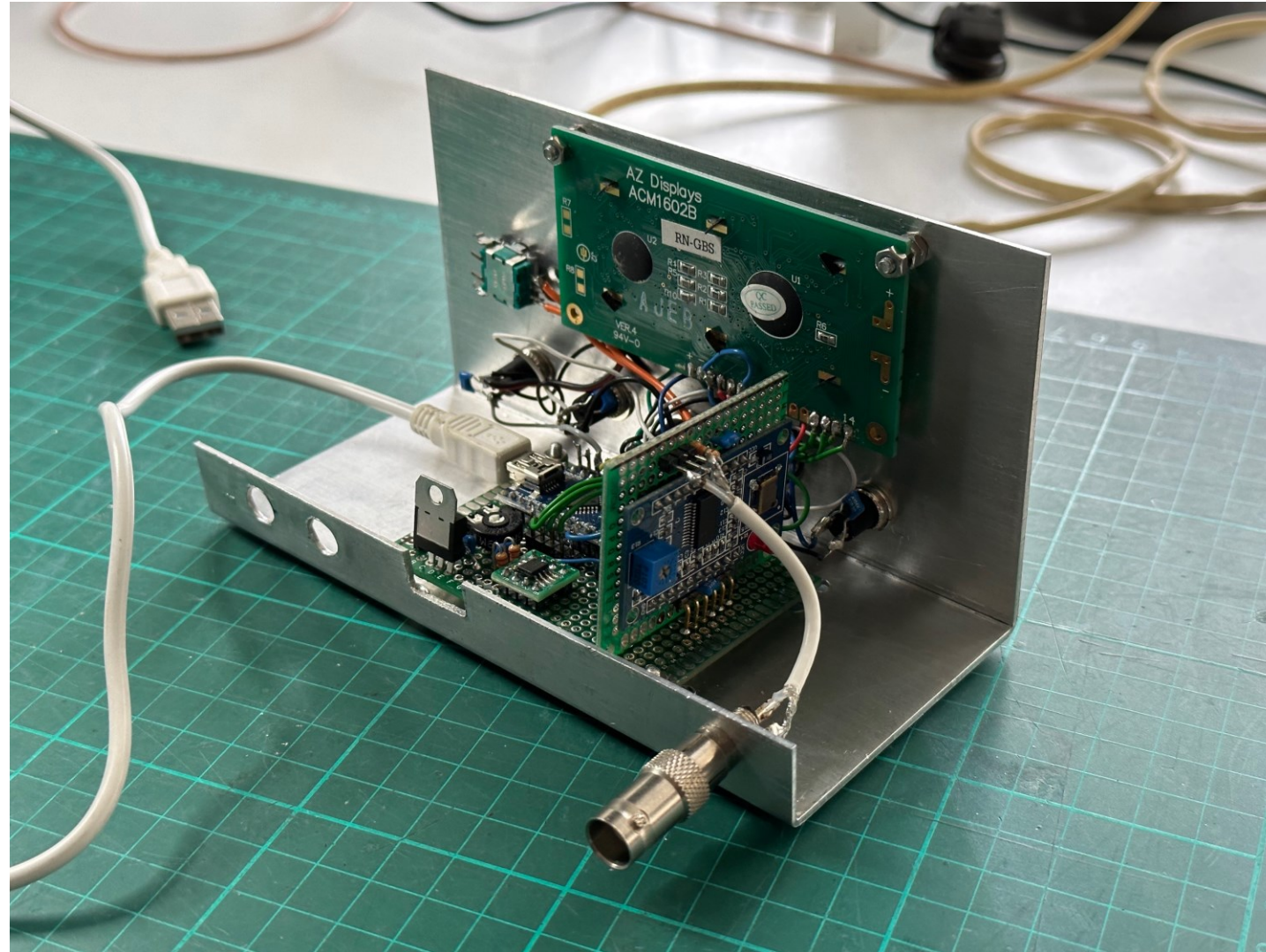
DSS VFO

Apple, Windows, Linux PC of Raspberry PI

# Het prototype voor de spraaktests



# Het prototype voor de spraaktests



# Het prototype voor de spraaktests

