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

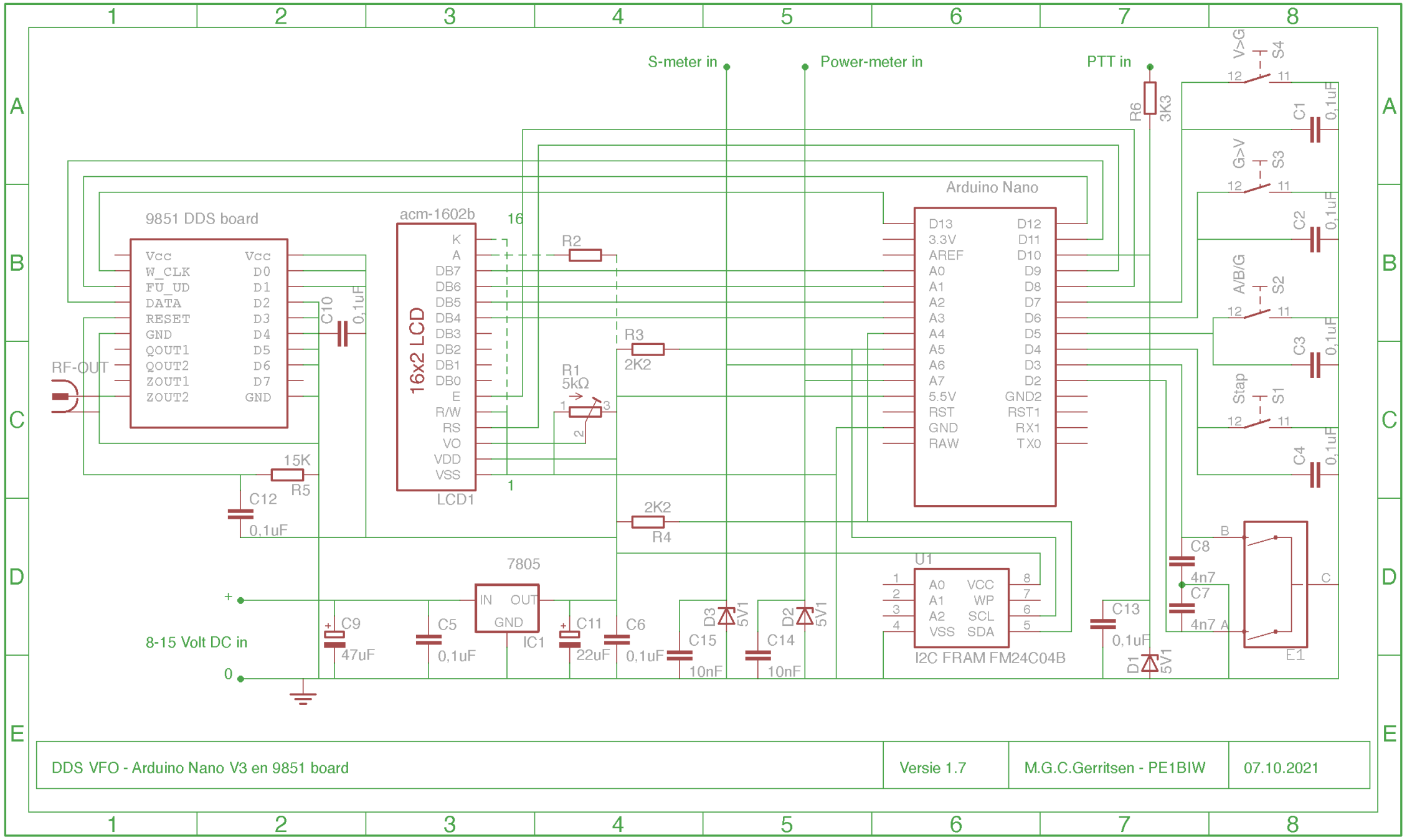
- Frequentieopwekking door Directe Digitale Synthese
- De vorm van het signaal opgeslagen als waarden (sinus)
- Een (eventueel extern vergrendelde) referentie oscillator
- Een digitaal analoog omzetter
- Een laag doorlaat filter

De voorwaarden

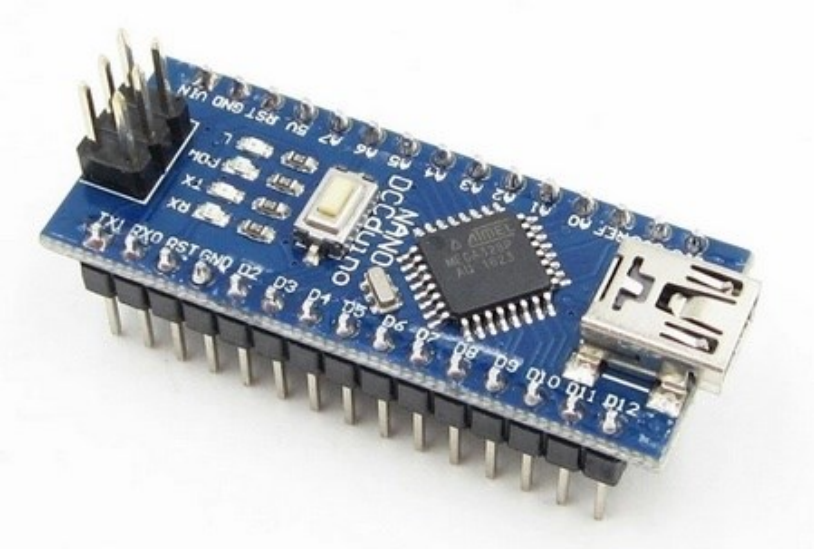
- ‘Kristal stabiel’
- Instelbaar over ten minste het HF gebied
- Frequentie stappen instelbaar (10Hz, 100Hz, 1kHz, etc.)
- Geheugen voor laatste toestand na uitschakelen
- Geheugen opslag voor vaak gebruikte frequenties
- Eventueel een MF offset
- ‘Schoon’ uitgangssignaal

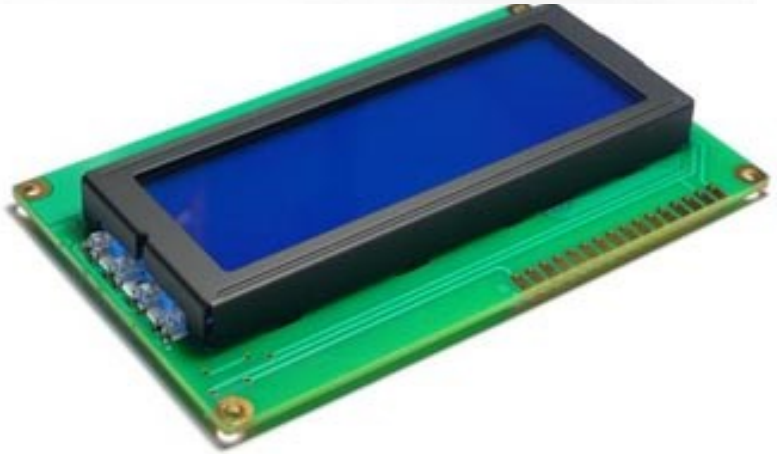
De onderdelen van het VFO

- 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)


S-meter in
Power-meter in
PTT in
9851 DDS board
Vcc
W_CLK
FU_UD
DATA
RESET
GND
QOUT1
QOUT2
ZOUT1
ZOUT2
Vcc
D0
D1
D2
D3
D4
D5
D6
D7
GND
RF-OUT
C10
0,1uF
acm-1602b
16x2 LCD
LCD1
K
A
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
RS
VO
VDD
VSS
16
1
R2
R3
2K2
R1
5kΩ
15K
R5
C12
0,1uF
2K2
R4
Arduino Nano
D13
3.3V
AREF
A0
A1
A2
A3
A4
A5
A6
A7
5.5V
RST
GND
RAW
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
GND2
RST1
RX1
TX0
R6
3K3
V>G
S4
G>V
S3
A/B/G
S2
Stap
S1
C1
C2
C3
C4
0,1uF
7805
IN
OUT
GND
IC1
8-15 Volt DC in
0
C9
47uF
C5
0,1uF
C11
22uF
C6
0,1uF
D3
5V1
D2
5V1
C15
10nF
C14
10nF
U1
A0
A1
A2
VSS
VCC
WP
SCL
SDA
I2C FRAM FM24C04B
C13
0,1uF
D1
5V1
C8
4n7
C7
4n7
B
A
C
E1
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

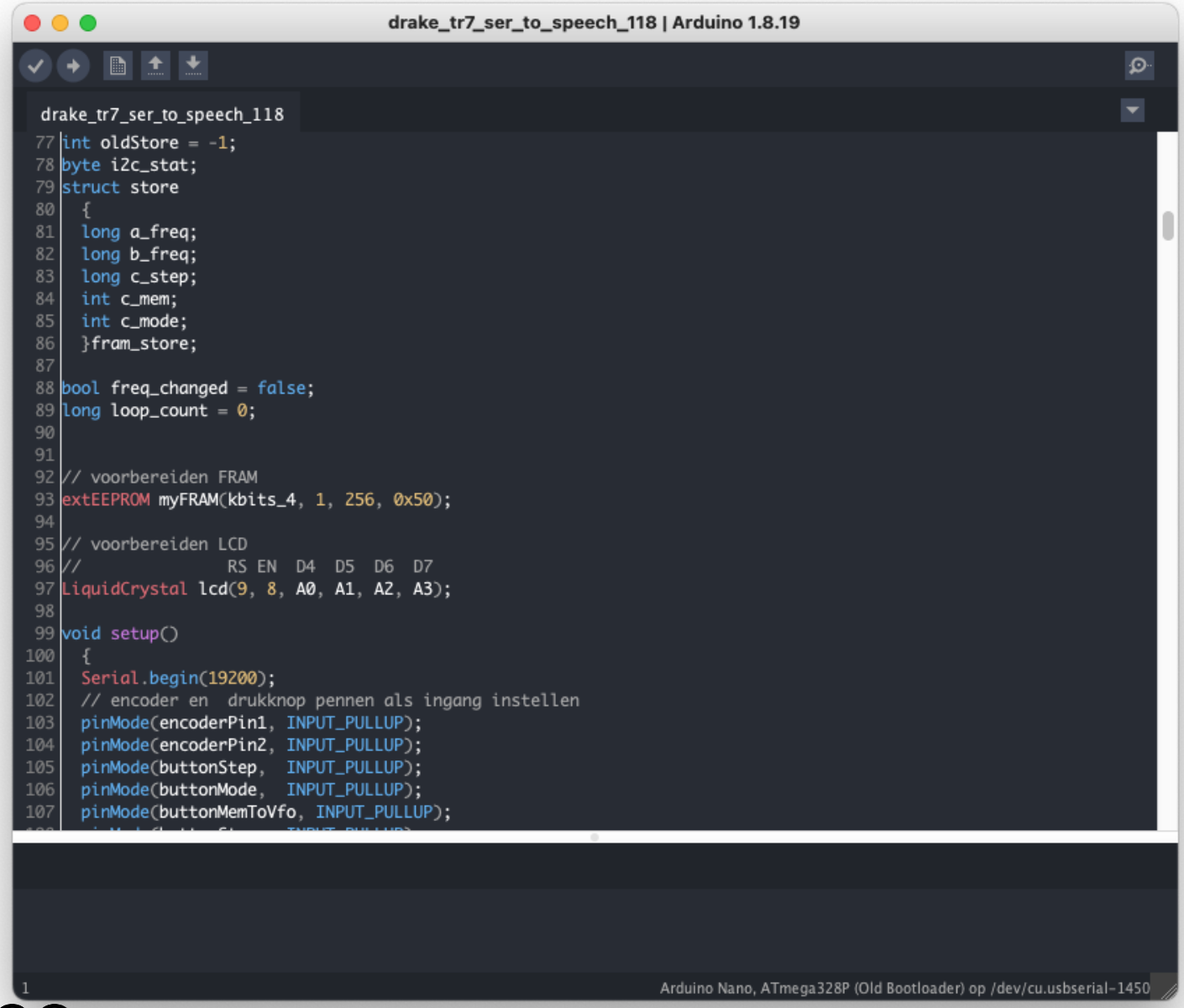
- een processor
- werk geheugen
- programma opslag geheugen
- EEPROM
- digitale- en analoge in- en uitgangen (pins)
- een USB aansluiting
- I^2C en SPI bus
- geen geheugen dat vaak gelezen en geschreven kan worden; daarom dus een externe FRAM chip die 10^{14} keer gelezen en geschreven kan worden

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



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

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

```

Bibliotheken (#include)

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

Constante 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
```

Constante waarden (#define)

```
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);
```

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     // interrrupt 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    }
```

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!

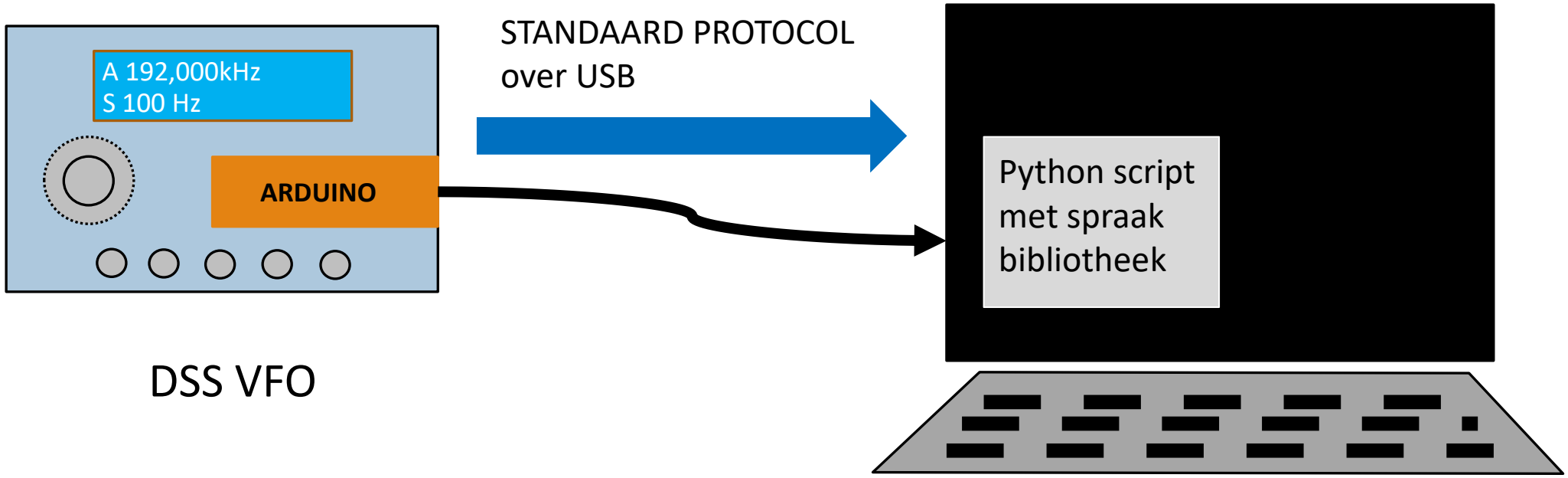
Vragen?

PE1BIW@ZIGGO.NL

of

3705kHz AM ;-)

Het prototype voor de spraaktests

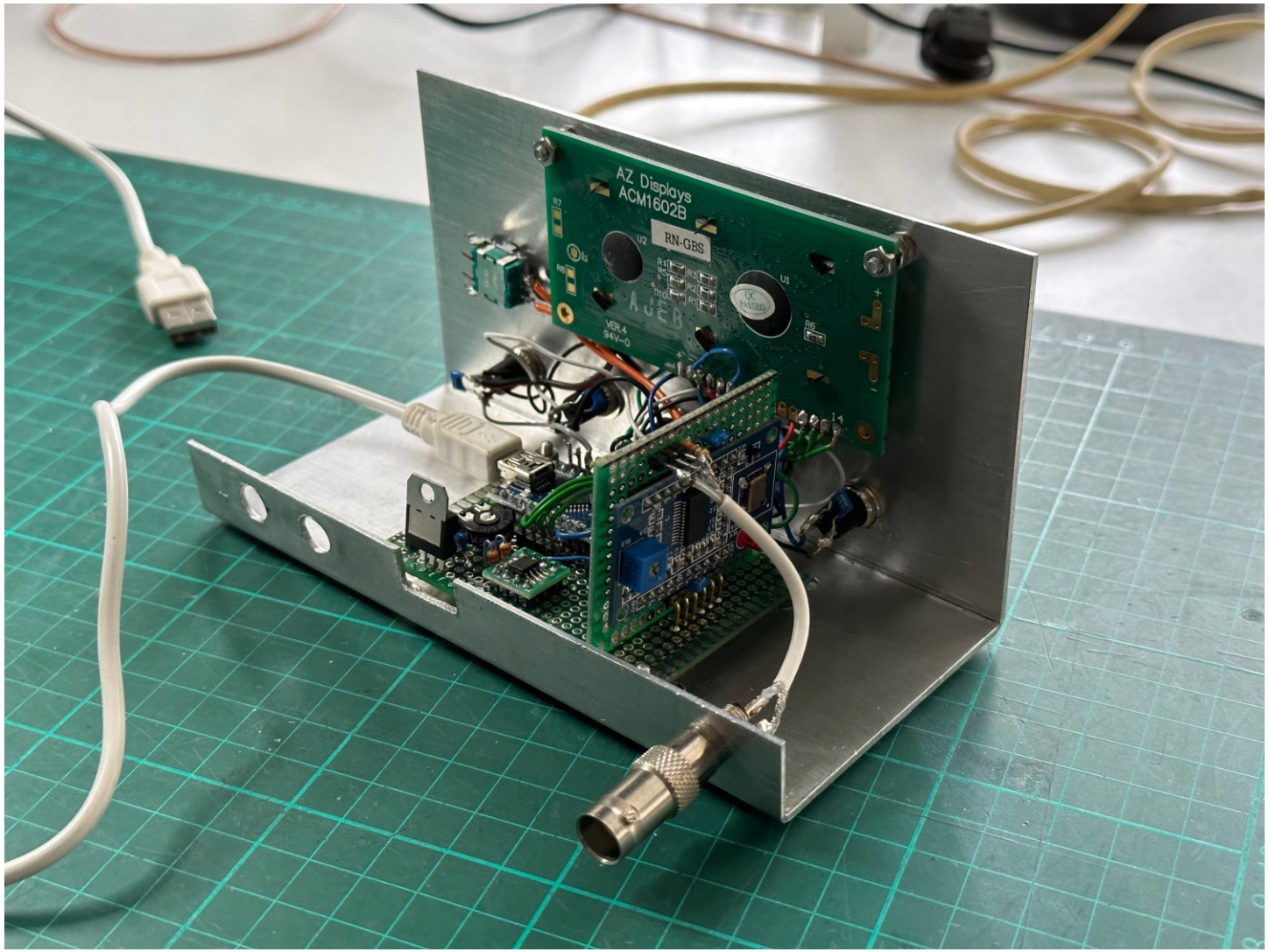


DSS VFO

Apple, Windows, Linux PC of Raspberry PI

Het prototype voor de spraaktests

Het prototype voor de spraaktests



Het prototype voor de spraaktests