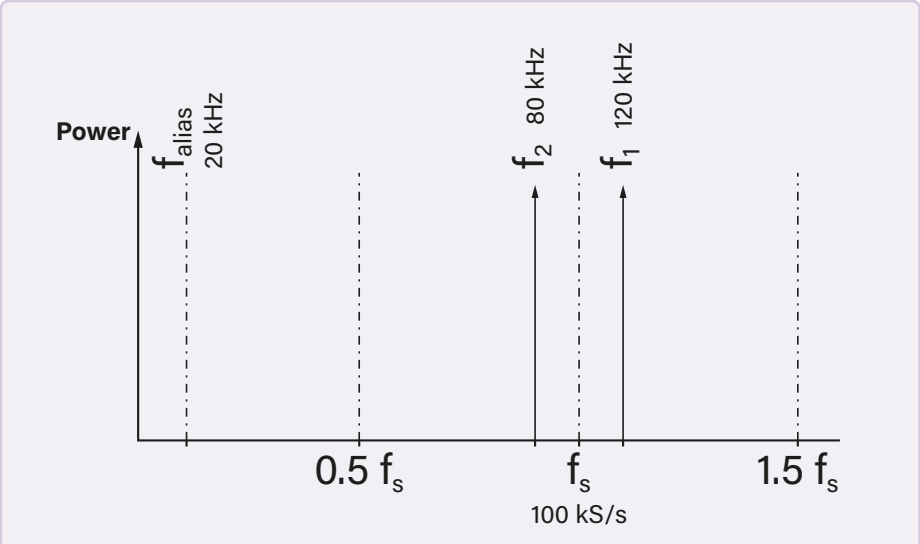
Sub-Nyquist-sampling in de praktijk

hogere frequenties betrouwbaar bemonsteren met subsampling

Sebastian Westerhold (AI5GW) (Duitsland)

Iedereen die te maken heeft met bemonsteringssystemen zal zich snel bewust worden van de beperkingen die worden opgelegd door het Nyquist-Shannon bemonsteringstheorema. Volgens deze regel kunnen signalen met frequenties hoger dan de helft van de bemonsteringsfrequentie niet meer betrouwbaar worden gedetecteerd. Dit is echter maar de halve waarheid. Een goede methode om signalen met een hogere frequentie te bemonsteren staat bekend als subsampling, ook wel sub-Nyquist-sampling genoemd. Dit artikel presenteert de methode op een praktische manier, met behulp van een eenvoudige Arduino Uno.

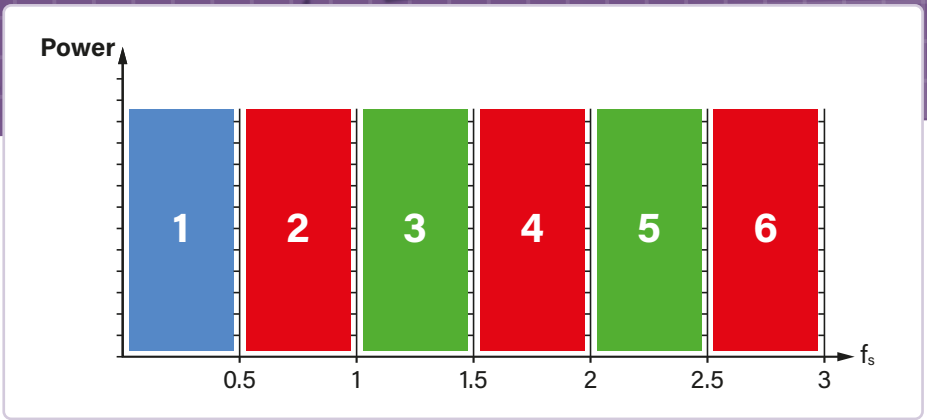


Figuur 1. Bij een bemonsteringsfrequentie van 100 kS/s veroorzaken zowel een signaal van 120 kHz (f_1) als een signaal van 80 kHz (f_2) een aliasingeffect (f_{ALIAS}) van 20 kHz.

Het probleem

Het Nyquist-Shannon bemonsteringstheorema is uiterst belangrijk voor bemonsteringssystemen. De zeer vereenvoudigde versie ervan zegt dat de bemonsteringsfrequentie minstens tweemaal de hoogste frequentie in het te bemonsteren signaal moet bedragen. Zoiets heb je vast wel eens gehoord of gelezen.

Wat gebeurt er als je een zuiver sinusvormig signaal van 120 kHz bemonstert met een bemonsteringsfrequentie van slechts 100 kS/s? Dit resuleert in wat we aliasing bij 20 kHz noemen. De verkregen samples zien er dan uit alsof een sinusvormig signaal met een frequentie van 20 kHz (120 kHz – 100 kS/s) is bemonsterd. En aliasingfouten moeten met alle middelen worden vermeden. Of niet soms? Of aliasingfouten een storende factor zijn of daarentegen als nuttige techniek kunnen worden gebruikt, hangt af van de specifieke toepassing. In het voorbeeld met de bemonsteringsfrequentie van 100 kS/s zou het onmogelijk zijn een 20kHz-signaal te onderscheiden van een 120kHz-signaal; beide signalen zouden eruitzien als een 20kHz-signaal. Overigens zou een signaal van 80 kHz ook een aliasingsignaal van 20 kHz (100 kS/s – 80 kHz) opleveren (**figuur 1**). Deze omstandigheid is echter alleen van belang als een onderscheid tussen de genoemde signaalfrequenties echt nodig is. Als alle daadwerkelijk in het signaal voorkomende frequentiecomponenten binnen een duidelijk herkenbaar bereik vallen, kan zinvol worden bemonsterd terwijl aliasingeffecten opzettelijk worden uitgebuit. De eenduidigheid is gegeven als alle in het te bemonsteren signaal voorkomende frequentiecomponenten binnen dezelfde Nyquist-zone vallen. Het frequentiebereik van 0 Hz (DC) tot de helft van de bemonsteringsfrequentie (f_s) wordt Nyquist-zone 1 genoemd (**figuur 2**). Deze zone is het 'normale' werkgebied voor



Figuur 2. Nyquist-zones 1…6. Zone 1 is het 'normale' bereik, zones 2, 4 en 6 zijn inverterende bereiken, zones 3 en 5 = niet-inverterende zones.

een bemonsteringssysteem. Nyquist-zone 2 strekt zich uit van 0,5 f_s tot f_s. Deze zones gaan dan theoretisch oneindig door met intervallen van 0,5 fs. De Nyquist-zones met even nummers hebben een bijzondere eigenschap: binnen deze zones vindt een inversie van het frequentiespectrum plaats.

Als de bandbreedte van een te detecteren signaal dus beperkt is tot het bereik van een van deze Nyquist-zones, kan subsampling worden gebruikt om signalen te detecteren waarvan de frequentiecomponenten de bemonsteringsfrequentie aanzienlijk overschrijden.

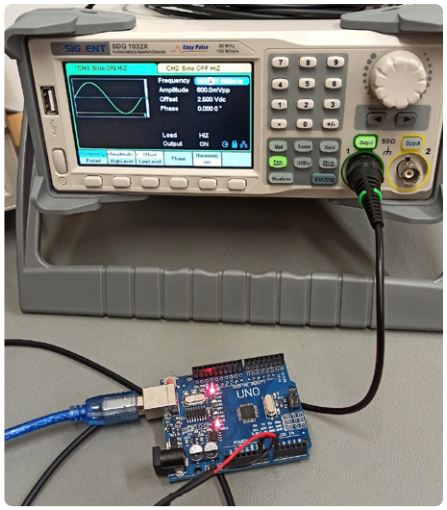
Een praktijkvoorbeeld

Voor een project moet de frequentie van een signaal in een bereik van ongeveer 160 kHz tot 165 kHz betrouwbaar worden gedetecteerd met een Arduino Uno [2] (**figuur 3**). Om technische redenen is de bemonsteringsfrequentie ingesteld op ongeveer 154 kS/s met behulp van de betreffende registers (prescaler 1:8) (strikt genomen is dat 153,846 S/s; voor de duidelijkheid wordt hieronder de afgeronde waarde gebruikt). De signaalfrequentie is dus enkele kHz hoger dan de bemonsteringsfrequentie zelf, maar kleiner dan 1,5 fs (overeenkomend met Nyquist-zone 3). Daardoor ligt de te verwachten aliasing als gevolg van subsampling tussen 6 kHz (160 kHz – f_s) en 11 kHz (165 kHz – f_s).

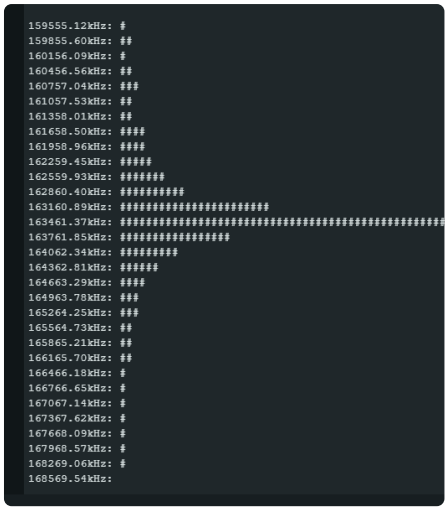
De Arduino-code is zo geschreven dat de A/D-converter van de Arduino een interruptroutine (ISR) gebruikt om het signaal op analoge ingang 0 te bemonsteren totdat een buffer met 512 discrete samples is gevuld. De (aliasing-)signaalfrequentie wordt dan uit deze waarden berekend voordat de volgende 512 samples worden ingelezen en het spel opnieuw begint.

Om de frequentie te bepalen is het Goertzel-algoritme gebruikt, een speciale vorm van de discrete Fouriertransformatie die zuinig met resources omgaat [1]. Het resultaat wordt vervolgens voor demonstratiedoeleinden uitgevoerd via de seriële interface en visueel geformatteerd voor de seriële monitor (**figuur 4**).

Volgens hetzelfde principe zou je bijvoorbeeld een IF van 455 kHz van een radio-ontvanger in Nyquist-zone 6 (3 f_s, overeenkomend met ongeveer 462 kHz) met subsampling kunnen converteren en FSK-signalen (NAVTEXT, RTTY enzovoort) zelfs met de niet extreem krachtige Arduino Uno kunnen demoduleren.



Figuur 3. Testopstelling met Arduino en signaalgenerator.



Figuur 4. De resultaten worden naar de seriële monitor gestuurd.

Opmerkingen

Om subsampling naar wens te laten werken, moet de gebruikte ADC de vereiste analoge bandbreedte hebben. Deze is gewoonlijk veel groter dan de bemonsteringsfrequentie. In de datasheet vind je deze waarde meestal als "Full power bandwidth". Met de ATmega328P lijkt subsampling heel netjes te werken tot in het MHz-bereik, zoals mijn eigen experimenten suggereren. Helaas leverden mijn experimenten met de Raspberry Pi Pico geen veelbelovende resultaten op.

vertaling: Willem den Hollander — 220629-03

Vragen of opmerkingen?

Hebt u vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via sebastian@baltic-lab.com of naar de redactie van Elektor via redactie@elektor.com.

Gerelateerde producten

- **Siglent SDG1032X 2-ch Signal Generator (30 MHz) (SKU 20276)**
  www.elektor.nl/20276
- **OWON XSA810 Spectrum Analyzer (1 GHz) (SKU 19714)**
  www.elektor.nl/19714
- **HackRF One Software Defined Radio (1 MHz to 6 GHz) (SKU 18306)**
  www.elektor.nl/18306
- **MonoDAQ-U-X Multifunctional USB Data Acquisition System (50 kS/s) (SKU 18766)**
  www.elektor.nl/18766

WEBLINKS

[1] Sebastian Westerhold (2022) – Goertzel-bibliotheek voor Arduino: https://github.com/AI5GW/Goertzel
[2] Arduino-sketch bij dit artikel: http://www.elektormagazine.nl/220629-03