Arduino en GPS-data

Hans Wagemans ON4CDU
on4cdu@uba.be

Met een Arduino is het bekijken van GPS-data geen probleem. Met wat extra hardware kunnen we het GPS-signaal ook gebruiken om er een nauwkeurige frequentiestandaard mee te maken.

Inleiding

Arduino en GPS-dataontvangst bieden veel uitdagingen om daar eens mee te experimenteren. Dit artikel beschrijft hoe je een 'GPS Display Unit' kunt maken en GPS voor frequentiestabilisatie kunt gebruiken.
Maar is daar dan geen appje voor? Jazeker, voor de GPS-data bestaat dat. Op je smartphone kun je een app installeren om de belangrijkste data zichtbaar te maken, met de locatie-indicatie desgewenst ook in de Maidenhead-notatie. Maar, met een variant op het oude gezegde 'het geluid uit een zelfgebouwde ontvanger klinkt beter': de gegevens op je zelfgebouwde GPS-display zijn duidelijker…
Het GPS (Global Positioning System) heeft de unieke eigenschap dat je het zowel voor plaatsbepaling als voor tijd(= frequentie)-referentie kunt gebruiken. Een aantal mogelijkheden voor het praktisch gebruik door radioamateurs komt in dit artikel aan bod.
Om de experimenten met de Arduino te kunnen uitvoeren wordt een kleine voorkennis verondersteld, namelijk dat de voorbeelden uit de Arduino IDE geen probleem opleveren, dat men een nieuwe bibliotheek kan laden, en dat men de seriële monitor begrijpt. Wie die basiskennis nog niet heeft, kan op internet hierover veel stuff vinden; bijvoorbeeld het Arduino Cookbook, of blader eens door de workshoppapieren op mijn internetsite: http://on4cdu.net/arduino-workshop/ .

GPS-data

GPS-ontvangers leveren data in NMEA-formaat. Het protocol bestaat uit losse berichten waarin gegevens gegroepeerd zijn. De berichten bestaan uit leesbare ASCII-karakters plus de 'carriage return' en 'line feed'-karakters. Het NMEA-protocol is op het internet goed beschreven. Bekijken we GPS NMEA-data op een display, dan zien we iets als in afbeelding 1.

De berichten beginnen met een '$' en een identificatiewoord. Daarna volgt de data, gescheiden door komma's, en aan het einde

```
$GPGSV,4,3,13,26,07,278,,29,66,203,44,31,43,303,42,33,29,205,34*70
$GPGSV,4,4,13,39,29,154,47*49
$GPGLL,5046.93922,N,00431.96128,E,115450.00,A,D*6E
$GPRMC,115451.00,A,5046.93926,N,00431.96130,E,0.020,,060715,,,D*7C
$GPVIG,,T,,M,0.020,N,0.036,K,D*21
$GPGGA,115451.00,5046.93926,N,00431.96130,E,2,07,1.16,107.5,M,46.2,M,,0000*53
$GPGSA,A,3,02,29,12,25,31,24,06,,,,,1.82,1.16,1.39*0B
$GPGSV,4,1,13,02,34,066,51,06,11,031,44,10,00,050,,12,39,087,50*7C
$GPGSV,4,2,13,14,24,232,,21,00,179,,24,07,148,45,25,79,061,45*73
$GPGSV,4,3,13,26,07,278,,29,66,203,44,31,43,303,42,33,29,205,34*70
$GPGSV,4,4,13,39,29,154,47*49
$GPGLL,5046.93926,N,00431.96130,E,115451.00,A,D*62
$GPRMC,115452.00,A,5046.93931,N,00431.96130,E,0.010,,060715,,,D*7A
$GPVIG,,T,,M,0.010,N,0.019,K,D*2F
$GPGGA,115452.00,5046.93931,N,00431.96130,E,2,07,1.16,107.3,M,46.2,M,,0000*50
$GPGSA,A,3,02,29,12,25,31,24,06,,,,,1.82,1.16,1.39*0B
```

Afb. 1 GPS-data

van het bericht een checksum. Als voorbeeld het bericht met het identificatiewoord GPRMC (GPS Recommended Minimum data):

$GPRMC,115452.00,A,5046.93931,N,00431.96130,E,0.10,,060715,,,D*7A

De data tussen de komma's heeft de volgende betekenis:

Bericht	Voorbeeld	Beschrijving/notatie
Berichtnaam	$GPRMC	RMC bericht
Tijd in UTC	115452.00	hhmmss.ss
Geldigheid van het bericht	A	A (Active = OK) of V (Void = NOK)
Breedtegraad	5046.93931	ddmm.mmmmm
Halfrond	N	noord (N) of zuid (S) van de evenaar
Lengtegraad	00431.96130	dddmm,mmmm
Lengte oost/west	E	oost (E) of west (W) van de nulmeridiaan
Snelheid t.o.v. grond	0.10	in knopen (1 knoop = 1,852 km/h)
Ware koers	-	graden t.o.v. ware noorden
Datum	060715	ddmmyy
Magnetische variatie	-	in graden; niet altijd aanwezig
Richting magnetische variatie	-	oost (E) of west (W); niet altijd aanwezig
Checksum	7A	

De voor ons meest interessante data zijn opgenomen in de GPRMC en GPGGA-strings. Datum, tijd, positie en hoogte zijn voor de hand liggende gegevens. Daarnaast willen we graag iets weten over de betrouwbaarheid van deze informatie. Hiervoor zijn de geldigheid van het bericht (A of V), data uit de GPGGA-string (FIX = 0: onvoldoende informatie voor betrouwbare berekening, FIX >1: berekening betrouwbaar) en de HDOP (Horizontal Dilution Of Precision) kandidaten. Als extra is ook het aantal satellieten waarvan de gegevens verwerkt worden in de ontvanger een goede bron. Let ook eens op de notatie van de positiegegevens in graden en decimale minuten. Het blijkt ook dat het aantal cijfers achter het decimaalteken niet bij alle GPS-ontvangers hetzelfde is.

Arduino GPS-display

Interessant en leerzaam is het om de GPS-ontvanger aan een Arduino Uno of Nano aan te sluiten en de datastrings op een computerscherm te bekijken.
Er bestaan veel GPS-ontvangers die geschikt zijn om op een Arduino aan te sluiten. Erg populair en goedkoop op internet zijn printjes met de aanduiding GY-NEO6MV2.

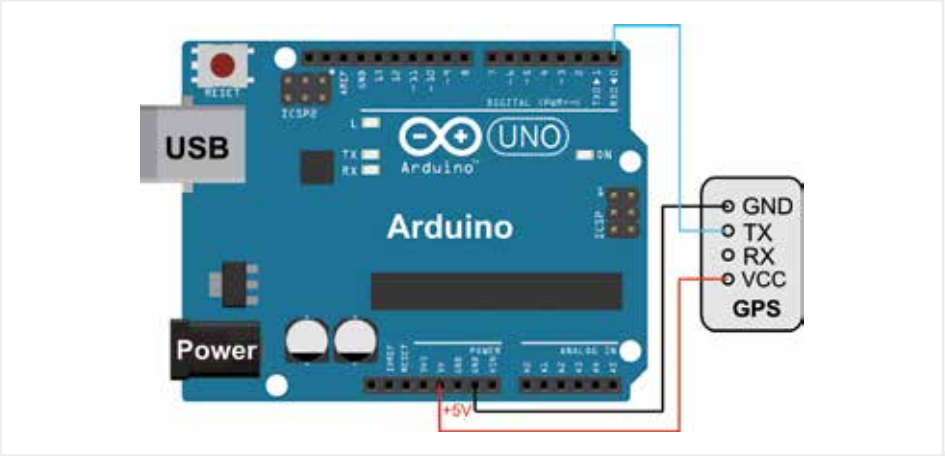
Een GY-NEO6MV2 module met patchantenne

Rockwell Jupiter GPS-ontvanger

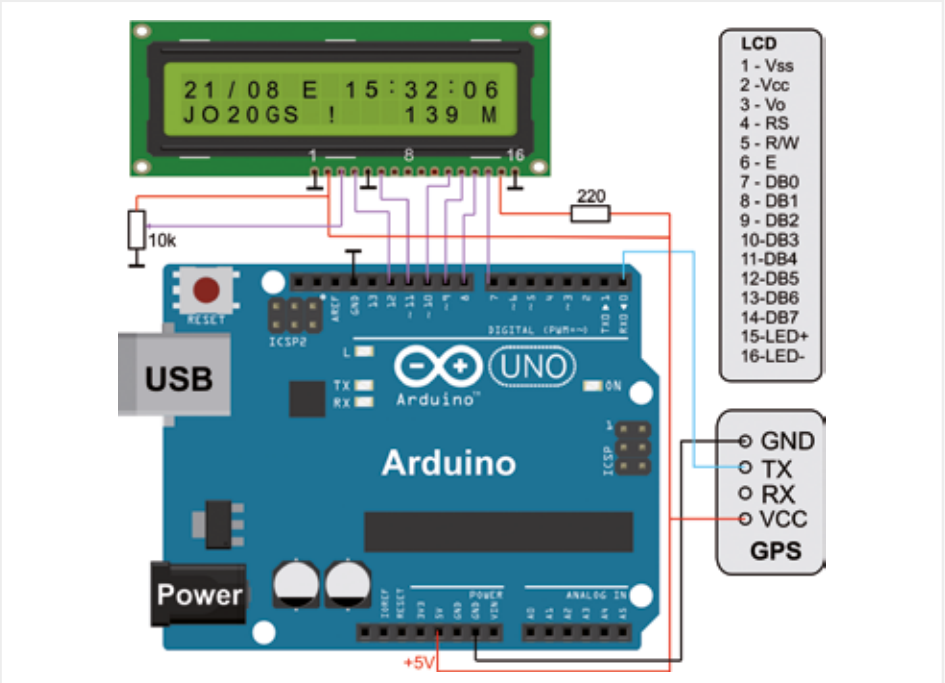
Deze printjes, met een u-blox IC erop, worden met een actieve patchantenne geleverd. De NMEA-berichten zijn beschikbaar op de TX-uitgang van het printje, en de standaardsnelheid daarvan is 9600 bits/sec.

De 'oude' Rockwell Jupiter GPS-ontvanger is ook te gebruiken. Deze ontvangers zijn/waren zeer gewild bij radioamateurs, omdat ze verkrijgbaar en goedkoop waren, en ook nog een 10kHz-uitgangssignaal afgeven dat voor synchronisatiedoeleinden gebruikt kan worden. De aansluitingen van de Rockwell Jupiter staan goed beschreven op http://www.gpskit.nl/ . De standaard bitsnelheid van de NMEA-uitgang is hier 4800 bits/sec.

Start de Arduino IDE op de pc en laad hierin het programma 'GPS-string naar serial monitor' (http://www.veron.nl ; Downloads; Electron bijlagen). Stel de datasnelheid voor de GPS-ontvanger in (Jupiter 4800 en u-blox 9600 bits/sec) en laad het programma in de Arduino. Sluit nu de NMEA-uitgang van de ontvanger aan op de ingang van de UART van de Arduino (pin RX bij Uno; RX0 bij Nano). Met de seriële monitor zal nu de GPS-data op het scherm verschijnen. Denk eraan de seriële monitor op de juiste datasnelheid in te stellen, en wanneer een programma in de Arduino geladen moet worden, de ontvanger los te koppelen van pin RX van de Arduino; deze input wordt ook gebruikt om nieuwe software te laden!
Als alles goed gegaan is zijn de GPS-strings zichtbaar op het scherm. Het is interessant de data te bestuderen en te interpreteren. Trek ook de GPS-antenne eens los en kijk wat er dan gebeurt.
Moderne ontvangers starten veel sneller op en reageren ook sneller op veranderende omstandigheden dan de oude Jupiters. Ze zijn daarom ook geschikt voor 'flight control' toepassingen. Verder kan bij oudere Jupiter ontvangers een onjuiste datum op het scherm verschijnen. Daarover later meer.
We weten nu dat de GPS-ontvanger NMEA-data aan de Arduino levert, en de volgende stap kan worden gezet.
De Roverbox, ontworpen door Christophe ON4IY, is een GPS-informatie display unit. We hebben hem binnen ons microgolfroverstation jarenlang gebruikt. De originele Roverbox staat beschreven op http://www.qslnet.de/member/on4iy/roverbox.html .
Van dat ontwerp heb ik nu een Arduinoversie gemaakt. De nieuwe Roverbox bestaat uit



Afb. 2 Arduino met GPS-ontvanger



Afb. 3 Schema van de experimentele GPS Display Unit

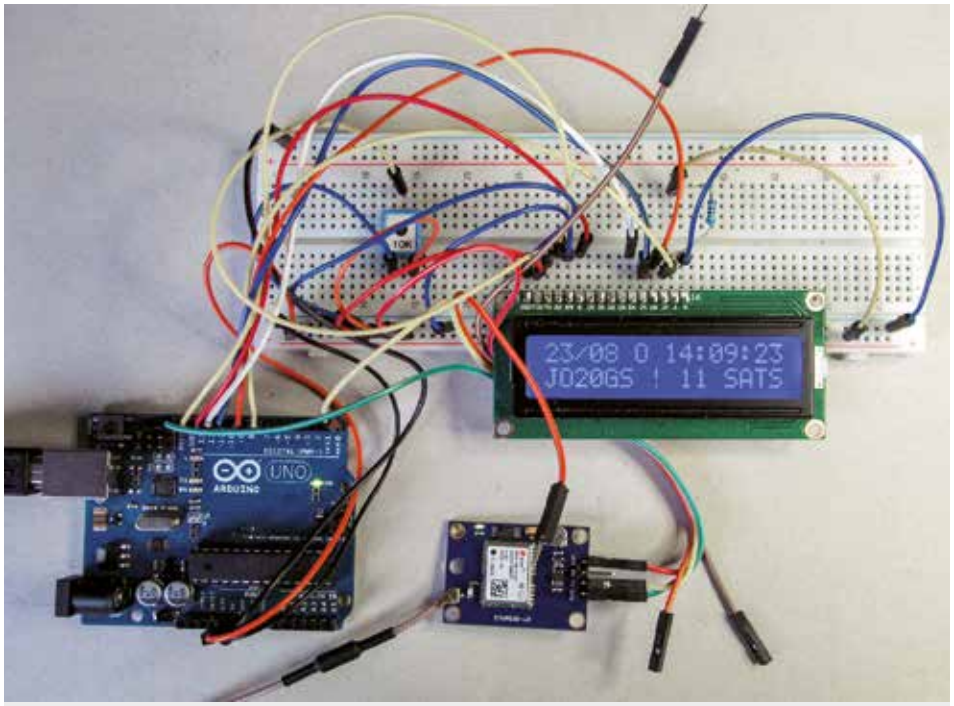
een Arduino, een 16x2 lcd en een GPS-ontvanger.

Voor de experimentele opstelling is geen externe voeding nodig; de usb-aansluiting is de voedingsbron. Er mag natuurlijk wel een externe voeding gebruikt worden. De Uno is zo beveiligd dat er geen voedingconflict optreedt als er zowel een usb- als een externe voeding is aangesloten.

De software genaamd 'GPS display' wordt in de IDE geladen, de interfacesnelheid wordt eventueel aangepast en de software wordt naar de Arduino overgebracht. Nu (en niet eerder) wordt de GPS-ontvanger aangesloten. Als het goed is zal nu data op het lcd te zien zijn. Op de eerste regel de datum, 'O' of 'E' (odd of even), en de tijd. Regel twee begint met de QTH-locator, dan een '?' als de data niet betrouwbaar is of een '!' als de

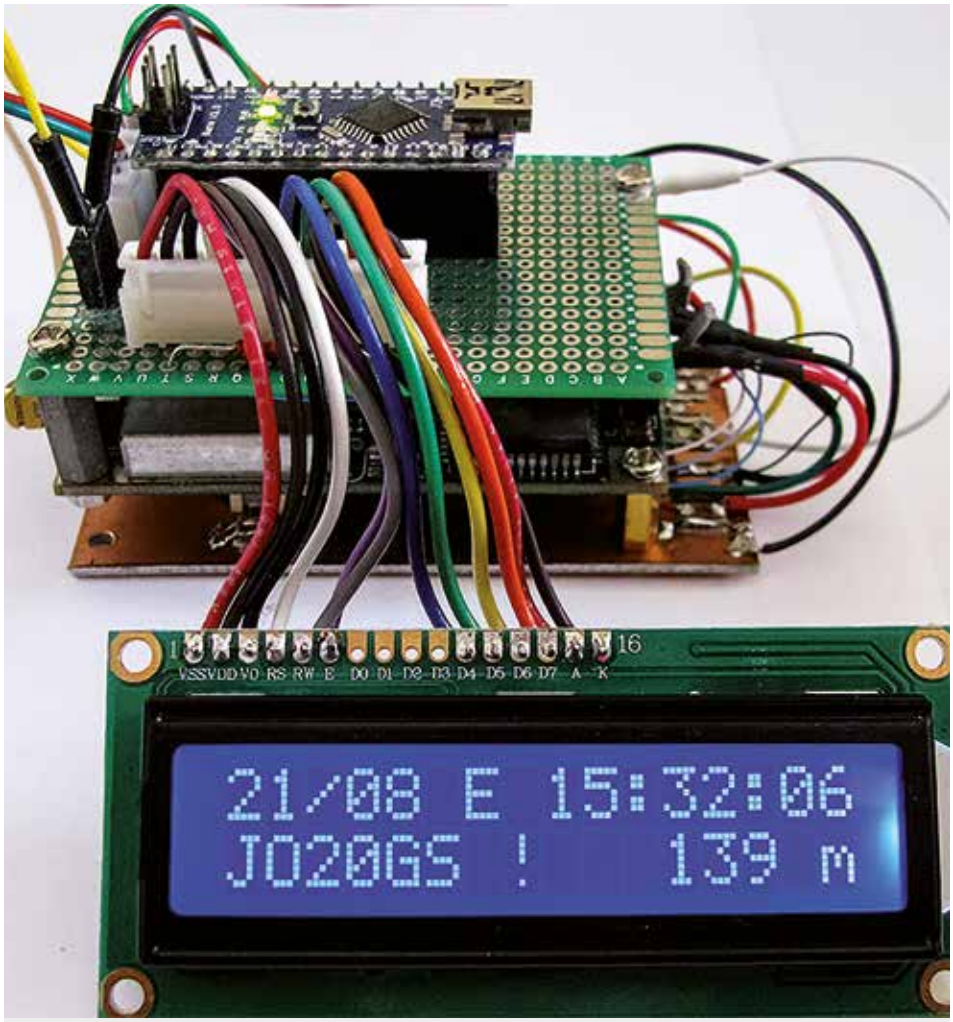
data betrouwbaar is, en dan afwisselend de HDOP, de hoogte ASL en het aantal gebruikte satellieten.
De odd/even-indicatie werd soms gebruikt in moeilijke microgolf-QSO's, waarin afgesproken werd wie tijdens de even of oneven minuten zendt of ontvangt. Voor 'normale' microgolf-vliegtuigscatterverbindingen is dit mechanisme echter te traag en dus niet bruikbaar.
De waarde van de HDOP is een maat voor de kwaliteit van de positie-informatie.

Bij toepassing van een Jupiter zal de tijd iets achterlopen. Dit is een gevolg van de procestijd in de ontvanger. Een correctie (in de software de variabele 'timeoffset') van 2 seconden is nodig. Verder kan het bij een Jupiter gebeuren dat er een onjuiste datum wordt weergegeven. Dit is een soort 'millenniumbug': de teller voor de datum heeft bij een aantal types een beperkte grootte, en is daardoor na 1024 weken (ruim 19 jaar) weer terug bij af. Om toch de juiste datum weer te geven moet eenmaal de juiste datum aan de ontvanger worden aangeboden, en deze informatie moet dan met een back-upbatterij (3 V aan pin 3 van de twintigpolige connector) worden vastgehouden. De back-upbatterij, bijvoorbeeld een CR2032 lithiumknoopcel, is bovendien nuttig omdat de ontvanger nu ook sneller start.

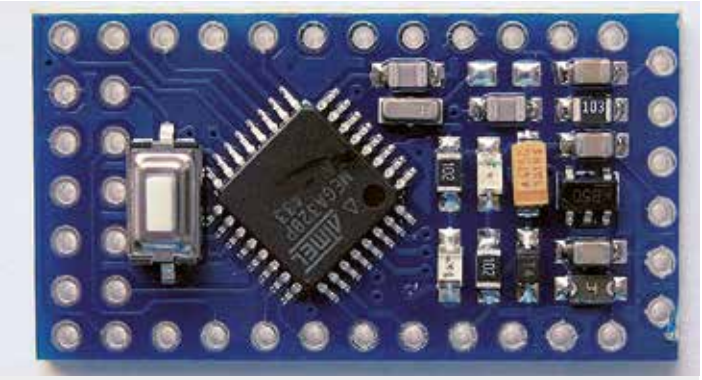


Experimentele opstelling van de GPS Display Unit met de NEO GPS-module

De informatie die aan de ontvanger moet worden toegevoerd staat ook in de software. In de set-up van het programma staat een regel met Serial.println. Maak deze regel actief en pas bij voorkeur ook de datum in deze regel aan. Laad het programma in de Uno en sluit dan op punt 0 (RX Arduino) en 1 (TX Arduino) van de Uno respectievelijk de NMEA-uitgang (pin 11 van de twintigpolige connector) en ingang (pin 12) van de Jupiter aan. Druk op de resetknop van de Uno om het programma opnieuw te starten, en de datum moet nu gecorrigeerd zijn. Dit hoeft maar een keer te gebeuren als er een back-upbatterij op de Jupiter is aangesloten. Maak daarna de regel weer onzichtbaar voor het programma of verbreek de verbinding met pin 12 van de Jupiter.
Denk erom de ontvanger telkens los te koppelen bij het laden van het programma via de usb-aansluiting, want deze aansluiting maakt ook gebruik van de punten 0 en 1 van de Uno.
Wie wat meer ervaring heeft kan ook het programma 'datum correctie Jupiter' gebruiken. Veel informatie over het configureren van de Jupiter ontvanger is te vinden op http://www.jrmiller.demon.co.uk/projects/ministd/jupcom.zip .
Enkele woorden over de software. Het maken van een programma om wat GPS-data op een lcd te schrijven is relatief eenvoudig. De moeilijkheden kwamen toen de antenne afgekoppeld werd en er onzin op het schermpje verscheen. Logisch, want ik had een vast formaat voor de datavelden van de strings aangenomen. Verder bleek ook dat de formaten van de berichten van een NEO en een Jupiter verschillen. In de software is hiermee nu rekening gehouden door telkens netjes de informatie tussen de scheidingskomma's te lezen, en niet van een vast aantal karakters van de data uit te gaan.
In de display unit worden alleen de GPRMC en GPGGA-strings gebruikt. Iedereen is natuurlijk vrij om ook andere datastrings uit te lezen.
De software is opgesplitst in subroutines. De subroutine getField distilleert een be-



Een sandwich bestaande uit een printje met de Arduino Nano, een Jupiter GPS-ontvanger en onderop een printplaatje met een back-upbatterij en een 5V-stabilisator voor de Jupiter. Op de voorgrond een 16x2 lcd.



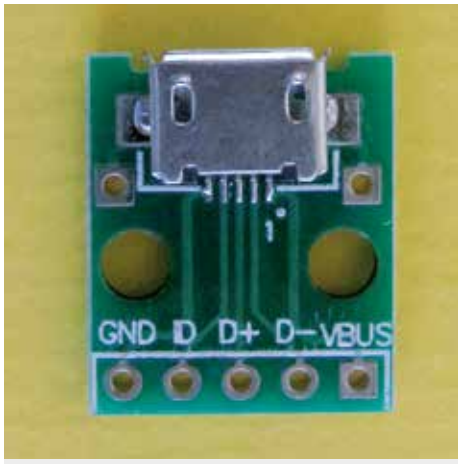
Een kloonversie van Arduino Mini Pro, met aan de linkerkant extra aansluitingen voor de programmer

Rechts het 'verkeerd om' male SMA-contact; links een gemodificeerde SMA-connector die nu daarop past

paald veld uit een string, de subroutine locator berekent uit de positie de QTH-locator, en de belangrijke subroutine displayGPS bepaalt wat er op het lcd te zien is.
De berekening van de locator wijst zichzelf. Tijdens mijn vakantie is de routine op een aantal plaatsen in Europa getest en hij gaf de juiste QTH-locator weer. Een test aan de andere kant van de nulmeridiaan of op het zuidelijk halfrond is niet gebeurd.

Praktische zaken

Als de experimenten met een Arduino geslaagd zijn komt het moment voor een definitief ontwerp en het klaarmaken van een kastje om alles in te stoppen. De Arduino Uno is heel geschikt voor experimenten, maar nogal groot voor een definitief apparaat. Een eenvoudige oplossing is een Arduino Nano te nemen; deze is compatibel en op dezelfde manier te programmeren als de Uno. Wie het nog kleiner wil maken kan voor de Mini (Pro) kiezen, een soort Nano zonder usb-aansluiting.
De Mini is klein (een Atmel processor op een printje met wat componenten er omheen) en goedkoop, maar bezit dus geen usb-aansluiting om het programma te laden. Het programmeren doe je via een externe usb-naar-serieelomzetter, of door een programmer aan te sluiten. Beide oplossingen zijn uitgebreid op het internet beschreven.
De Mini bestaat in twee uitvoeringen: de 5V 16MHz-versie en de 3,3V 8MHz-versie. Verder blijkt dat er kloonversies bestaan die een iets andere print hebben dan de originele. Deze kloonversie gebruik ik graag, omdat daar eenvoudig een programmer op aangesloten kan worden.
Doordat de Arduino Mini ook een 'mini' spanningsstabilisator heeft, kunnen de ontvanger en de verlichting van het lcd niet uit de processorprint gevoed worden. Een externe 5V-stabilisator of het gebruik van een 5V-voeding (bijvoorbeeld een usb-lader voor een mobieltje) zijn dan goede oplossingen.
De ontvangerprintjes GY-NEO6MV2 worden geleverd met een actieve patchantenne. Het kabeltje van de antenne naar de ontvanger is erg kort, omdat de antenne in mobiel-toepassingen vaak direct bij de ontvanger wordt gemonteerd. Om een langere kabel aan de antenne te maken kun je natuurlijk het bestaande plugje eraf knippen, maar mijn voorkeur is om de originele plugjes te laten zitten en aanpassingstukjes te maken. De toegepaste connectoren zijn van het type U.FL en erg klein. Op eBay zijn kabeltjes van U.FL naar SMA en losse U.FL connectors te koop. De U.FL kabeltjes naar SMA komen uit wifi-routers, waarin de conventie voor de SMA-aansluiting precies andersom is dan wij gewend zijn: male en female zijn omgewisseld. Dit is bewust gedaan om te beletten dat grote antennes eenvoudig op deze routers aangesloten kunnen worden. Ik heb deze kabeltjes gekocht en een verloop gemaakt om weer op de gebruikelijke SMA-conventie uit te komen. Het maken van zo'n verloop is relatief simpel doordat de male pen en female bus even groot zijn en dus te verwisselen.
De andere zijde van de U.FL connector heb ik op een klein printje gemonteerd en hierop wordt de patchantenne aangesloten.
De GPS-antenne is niet waterdicht, en als die buiten wordt gemonteerd is bescherming in een plastic doosje aan te raden.
Als voeding voor m'n Arduino display unit is een usb-lader voor de mobiele telefoon gebruikt. Een micro-usb-connector op een printje gemonteerd om de lader op aan te sluiten kost 'niets' op internet, en maakt de montage wel heel eenvoudig.



Micro usb-chassisdeel op een printje voor aansluiting op een usb-lader

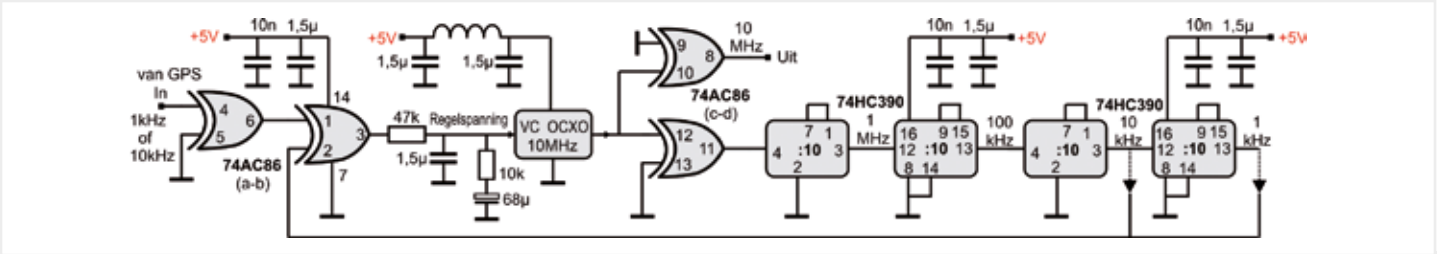
De usb 5V-voeding wordt gebruikt om de Arduino Mini, de verlichting van het lcd en de ontvanger te voeden. De lader kan zowel op de 5 V als op de RAW-aansluiting van de Arduino worden aangesloten. Zeker niet de lcd-verlichting en de GPS-ontvanger op de 5 V van de Arduino Mini aansluiten als de RAW-pin als voedingspunt wordt gebruikt; dat overleeft de spanningsstabilisator op het printje niet!

De definitieve GPS Display Unit

Het kastje van de display unit is van ABS en heeft afmetingen van 140 x 66 x 28. Het is van de firma Hammond en gekocht bij Conrad met bestelnummer 522343–89. In het kastje zit een Arduino Mini (5V 16MHz-versie), een GY-NEO6MV2 ontvanger, en een 16x2 lcd, dat met dunne strookjes blik is vastgemaakt zodat het een beetje te positioneren is in het uitgezaagde gat van het kastje.
Het frontje is gemaakt van fotopapier.
De lay-out voor het frontje is gemaakt met FrontDesigner van Abacom. Dit programma print op het fotopapier precies de maten die in het ontwerp zijn aangeduid.

Frequentiereferentie

Al heel lang wordt er door radioamateurs gebruik gemaakt van GPS om een frequentiebron, meestal 10 MHz, stabiel te maken. De stabiliteit is afkomstig van de klok van de GPS-ontvanger, die gesynchroniseerd wordt met de uiterst stabiele klokken in het GPS. Hierdoor **kan** (helaas niet altijd) de klok in de ontvanger zeer stabiel zijn. Ontwerpen om een externe 10MHz-bron aan de GPS-ontvangerklok te koppelen zijn talrijk. De twee bekendste zijn het ontwerp van VE2ZAZ, dat gebruik maakt van de secondepuls, en het ontwerp van G3RUH (en anderen) dat het 10kHz-signaal van een Jupiter GPS-ontvanger gebruikt.
Het VE2ZAZ-systeem (http://ve2zaz.net/GPS_Std/GPS_Std.htm) heb ik als hoofdreferentiebron in gebruik en synchroniseert



Afb. 4 Schema van de stabilisator

een Isotemp 10MHz-OCXO134-10. Het is geen PLL (Phase Locked Loop), maar een FLL (Frequency Locked Loop). De resultaten daarvan zijn zo goed dat op de 10GHz-amateurband frequentieafwijkingen van enkele Hz gemeten kunnen worden, maar soms is het dan niet duidelijk of dat door mijn bron, de achterzetontvanger of het beluisterde baken komt.
Het maken van zo'n referentie is niet echt moeilijk, maar om goede resultaten te bereiken moet wel veel aandacht aan details worden geschonken, en de bron moet een aantal uren aanstaan om 'op frequentie' te komen.
Als we een bron willen hebben die sneller op frequentie zit, bijvoorbeeld als we die in het veld willen gebruiken, dan kan de 10kHz-uitgang van de Jupiter uitkomst bieden. Het principe is eenvoudig: een fasevergrendeling van de referentiebron met het 10kHz-signaal uit de ontvanger. Pak een stabiele 10MHz-bron, deel het uitgangssignaal naar 10 kHz en vergelijk dat in een fasedetector met het signaal uit de Jupiter. Met een lusfilter sluiten we de lus.
Maar hoe zit dat met de GPS-ontvanger met u-blox ICs? Op het printje GY-NEO6MV2 zit een NEO-6M. In de standaarduitvoering geeft dit IC NMEA-signalen af met een snelheid van 9600 bits/sec, en ook een 1Hz-puls (knipperend ledje op de print). Hiermee zou dus een 10MHz-bron volgens het principe van VE2ZAZ kunnen worden gesynchroniseerd. De chip kan volgens de documentatie echter ook anders worden geconfigureerd; bijvoorbeeld door in plaats van een puls van 1 Hz, een 1kHz-signaal af te geven. Een hogere frequentie is helaas niet mogelijk.
Het configureren gaat met het programma 'u-center'. Dit programma staat op de site van de firma u-blox. Als interface tussen computer en de NEO-6M kan dezelfde usb-naar-serieelomzetter worden gebruikt als voor het programmeren van de Arduino Mini! Als u problemen hebt de juiste driver voor de usb-naar-serieelinterface te vinden, kijk dan eens op de site van Silicon Labs.
Het aanpassen van de tijdpuls van 1 seconde naar een frequentie van 1 kHz is eenvoudig te vinden in het configuratiescherm: kies daar TP. Stel de tijdpuls in op 1000 µs met een duty cycle van 50 procent. Wel nog het commando geven dat de nieuwe instelling bewaard moet worden, anders is de instelling weer verdwenen als de spanning van het IC wordt gehaald. Het 1kHz-signaal is beschikbaar op het ledje op de print; met draadjes heb ik dat aangesloten en het ledje heb ik laten zitten.
Het programma u-center heeft overigens heel veel mogelijkheden om allerlei GPS-informatie op het scherm zichtbaar te maken.

Op internet heeft G3RUH een site gewijd aan GPS-locking met de Jupiter ontvanger http://www.jrmiller.demon.co.uk/projects/ministd/frqstd0.htm. Ook PE9GHZ heeft het een en ander op zijn site beschreven. Het door mij toegepaste schema van fasevergrendeling is een kleine variant op wat op het internet te vinden is. Het gebruik van een exclusive-or als fasedetector maakt de schakeling weinig kritisch, en we hoeven ons geen zorgen te maken over de fase van de ingangssignalen. Een nadeel is wel dat er geen aansluiting is voor een locklampje; je ziet dus niet direct of de schakeling gelocked is. Een meter om de regelspanning te controleren biedt uitkomst. Het duurt zo'n tien minuten voordat de bron op frequentie staat; dat is afhankelijk van het opwarmen van de OCXO en van het toegepaste lusfilter.

Bij gebruik van een 10MHz-OCXO en de Jupiter wordt de vergelijkingsfrequentie 10 kHz. Bij de NEO-6M wordt dat dus 1 kHz. De gebruikte OCXO is een 5V-type, maar ook types met andere spanningen kunnen worden gebruikt. Een opamp om het regelspanningbereik aan de OCXO aan te passen kan dan nodig zijn.
Zelf heb ik het geheel op een gaatjesprint gezet, maar wie op internet zoekt zal vinden dat G3RUH printjes aanbiedt, en Eddy PA9GHZ heeft op zijn site een printlay-out gepubliceerd. Een kleine modificatie kan wel nodig zijn als 1 kHz als referentiesignaal wordt gebruikt. Het door mij gebruikte schema wijkt overigens iets af van de printversies, maar dat is niet essentieel.
Alles in een tochtvrij doosje monteren komt de stabiliteit ten goede. De kwaliteit van de OCXO bepaalt in grote mate de stabiliteit die bereikt kan worden. Optimaliseren van het lusfilter is zinvol om een goed compromis te vinden tussen regelsnelheid en stabiliteit.
Met deze opstelling wordt met de door mij toegepaste OCXO en de NEO-6M een kortetermijnstabiliteit van ongeveer 5×10^{-10} bereikt. Dat betekent in de praktijk een frequentie-instabiliteit van enkele Hz op de 10GHz-band. JT en PI4-demodulatie gaat daarmee probleemloos! Op de spectrum analyzer zijn op onderlinge afstanden van 1 MHz wat spurious signalen te zien. Als we de veelvouden van 10 MHz niet meetellen, zijn alle stoorsignalen minstens 60 dB onderdrukt. Met betere ontkoppelingen van de voedingssporen bleek dat nog verbeterd te kunnen worden. Waarschijnlijk zal een goed ontworpen printje deze spurious nog verder reduceren.

Nawoord

Dit artikel is een verslagje van mijn experimenten. Er zijn veel variaties, aanpassingen en verbeteringen mogelijk.

De frequentiereferentie opgebouwd op gaatjesprint

Bij het nakijken van de specificaties van de NEO-6M viel mijn oog op andere IC's die de firma u-blox kan leveren: GPS-ontvanger-IC's met daarin een TCXO en een tijdpuls-uitgang die geconfigureerd kan worden voor 10 MHz. De prijs viel wat tegen, en om er een te bestellen voor experimenten vond ik een brug te ver.
Ik heb wat op eBay gezocht, en jawel: er wordt voor een duidelijk lager bedrag een 'flight controller' voor Arduino verkocht met daarop een NEO-M8N. Dit IC bevat een TCXO en kan geconfigureerd worden met een 10MHz-tijdpulsuitgang. Een snel experiment met dit IC leerde dat de 10MHz-tijdpuls wel erg veel jitter bevat. Dit komt doordat deze frequentie met behulp van een fractional N-deler wordt afgeleid van de interne klok van het IC. Opschonen met behulp van een PLL is dus nodig, en het verschil met een PLL die via 1, 10 of 100 kHz een bron vergrendelt is dan niet erg groot. Wel kun je bedenken dat als je 100 kHz als vergelijkingsfrequentie kiest, in het ontwerp twee tiendelers vervallen, en dat dus een 74HC390 minder op het printje gesoldeerd hoeft te worden.
Om voor het predicaat 'ontwerp van de simpelste aan GPS-vergrendelde 10MHz-referentie' in aanmerking te komen zou je kunnen proberen de 10MHz-tijdpuls van de NEO-M8N en de 10MHz-OCXO zonder delers in een fasedetector te vergelijken, en dan met een stevig lusfilter de jitter te onderdrukken. Een mooie uitdaging voor u om dat te ontwerpen en te testen? Het resultaat lezen we graag te zijner tijd.

Module met NEO-M8N

Verder ondersteunt de NEO-M8N de GNSS(Global Navigation Satellite System)-gedachte, waarin GLONASS satellieten ook worden meegenomen in de berekeningen. GLONASS is de Russische tegenhanger van het Amerikaanse GPS. Als je de NMEA-data van dit IC bekijkt, zie je data die begint met GP, GL en GN. GP-data is van GPS, en GL-data is van GLONASS. De GN(GNSS)-data wordt gemaakt uit een combinatie van GPS en GLONASS data. De GN-data is daardoor nauwkeuriger en sneller beschikbaar dan de data van enkel GPS of GLONASS.
De Arduinosoftware is heel eenvoudig geschikt te maken voor deze ontvanger. De GPRMC en GPGGA-strings die door GPS gebruikt worden heten nu GNRMC en GNGGA. Deze aanpassing in het programma 'GPS-display' aanbrengen is alles wat nodig is. De resultaten op het display verschijnen sneller, de hoogte is nauwkeuriger en het aantal satellieten dat in de berekeningen gebruikt wordt is nu bijna altijd twaalf. Enkele metingen aan de HDOP bevestigen ook dat de plaatsbepaling nauwkeuriger is.
Enkele suggesties over de Arduinosoftware. Deze is door een amateur zonder programmeerachtergrond geschreven en kan zeker verbeterd worden. Het gebruik van de TinyGPS-bibliotheek is mogelijk zo'n verbetering, en zeker een vereenvoudiging. Deze bibliotheek is nu niet gebruikt omdat ik zelf wilde leren hoe je GPS-strings kunt verwerken, en de software is nu ook voor beginners beter toegankelijk.
Ook kan overwogen worden in plaats van de hardware-UART van de Arduino de SoftwareSerial-bibliotheek te gebruiken. Dat maakt het experimenteren eenvoudiger: de ontvanger hoeft niet telkens losgekoppeld te worden bij het laden van nieuwe software.
De HDOP-weergave op het GPS-display is weinig genuanceerd; dat kwam doordat de Jupiter ontvanger niet veel interessants daarover laat zien. Bij de modules met het NEO IC is dat beter, en nog interessanter is het als deze NEO ook GLONASS in de berekeningen meeneemt. Een HDOP-weergave met cijfers achter de komma is dan een logische verbetering.
Voor experimenten met verbindingen boven de 24 GHz, waarbij een heel nauwkeurige plaatsindicatie noodzakelijk is, kan een QTH-locator met acht karakters overwogen worden. Verder zou bijvoorbeeld ook de zons- of maanpositie zichtbaar gemaakt kunnen worden op het lcd. Een eerste poging daartoe lukte hier niet, maar dat zal zeker een vervolg krijgen.
Hoewel ik geen fouten in de dataontvangst heb kunnen ontdekken, is het wel zo netjes de nu nog niet gebruikte regelchecksums van de NMEA-data te gaan benutten.
Op degenen die nog niet erg veel met de Arduino gedaan hebben zal het vreemd overkomen dat er programmastatements gebruikt worden die niet in de Arduinoreferentie zijn opgenomen. Afhankelijk van de gebruikte processor kent de Arduino nog wat extra instructies. Wie daar meer van wil weten kijkt maar eens op de AVR Libc homepage http://www.nongnu.org/avr-libc/ .
Verder is het mij niet gelukt automatisch de bitsnelheid van de aangekoppelde GPS te bepalen. Wie heeft daar een oplossing voor? De GPS Display Unit bevat een Arduino Mini 5V 16MHz-versie. Een 3,3V 8MHz-versie werkt waarschijnlijk ook op die plaats, maar dat is niet uitgeprobeerd.
De gepubliceerde schakeling van de 10MHz-frequentiereferentie maakt gebruik van asynchrone logica, en is opgebouwd met componenten die al een aantal decennia hier in het bakje lagen. Dat is eigenlijk niet meer van deze tijd; geklokte systemen hebben nu de voorkeur, en zijn vaak ook noodzakelijk bij het toepassen van moderne componenten. De referentie werkt overigens prima en is geschikt voor veel amateurexperimenten. De schakeling levert wel een nogal grillig uitgangssignaal. Sommige PLL's wensen een net, sinusvormig referentiesignaal toegevoerd te krijgen. Dat kan bereikt worden door flink te filteren met 10BASE-T-filters uit oude Ethernet-netwerkkasten, bijvoorbeeld de FB2022.

FB2022 10BASE-T-filter in een netwerkkast

Een net uitgangssignaal kan ook worden bereikt door een lineaire bufferversterker direct achter de OCXO te plaatsen en dus van een digitale buffer af te zien. De afkorting OCXO is trouwens niet helemaal juist. VCOCXO, Voltage Controlled Oven Controlled Xtal Oscillator, of VCTCXO, Voltage Controlled Temperature Compensated Xtal Oscillator, zou beter zijn, maar die lange afkortingen kom je bijna niet tegen in artikelen.
Verder kan een soort lock indicator met een led of een meter worden bedacht door met opamps een schakeling te maken die aangeeft of de regelspanning niet te veel afwijkt van de spanning die nodig is om de OCXO op precies 10 MHz te houden.
Experimenten met een MC4044 of CD4046 als fasedetector met een lock indicator zijn mogelijk de moeite waard. In de gepubliceerde schakeling is een 74AC86 toegepast (AC staat voor Advanced high-speed CMOS); een 74HCT86 doet het op die plaats ook goed.
Wie een 10MHz-referentiesignaal wenst met nog hogere stabiliteit moet uit een ander vaatje tappen, en beginnen met het op de kop tikken van een heel goede 10MHz-bron. Het gebruik van PLL-technieken om op de microgolven voor smalbandcommunicatie een stabiele en nauwkeurige frequentie op te wekken is 'common practice'. Bijna altijd wordt een stabiele 10MHz-bron als uitgangspunt gebruikt. Bij een 10GHz-transverter voor smalbandtechnieken bijvoorbeeld wordt een kristaloscillator van 106,500 MHz in fase vergrendeld aan een 10MHz-bron. De frequentie van de kristaloscillator wordt voor het verkrijgen van een middenfrequentie van 144 MHz met 96 vermenigvuldigd. Dat betekent dat een verloop van 25 Hz van de kristaloscillator resulteert in een afwijking van 2,4 kHz op 10 GHz. Het vergrendelen van de kristaloscillator aan een zeer stabiele bron is dus zinvol.
De firma u-blox liet me weten dat zij niet kunnen garanderen dat je modules met NEO IC's die fabrikanten als hobbyproducten verkopen ook kunt reconfigureren, omdat de producent dat mogelijk geblokkeerd heeft. Bij de producten die ik op eBay gekocht heb, en die zeker in die categorie thuishoren, heb ik dat probleem nog niet ervaren.
Het is de moeite waard de site van u-blox te bezoeken en de grote hoeveelheid informatie over GPS, GNSS et cetera te bekijken.
Op mijn pagina http://on4cdu.net/arduino-en-gps-data/ zijn de Arduino-programma's en ook wat extra foto's en info te vinden.