GNSS Location hardware benchmarking

Realisatie

Bachelor in de IT-Factory Elektronica-ICT
keuzerichting IoT

Cisse T'Syen

Academiejaar 2021-2022

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

INHOUDSOPGAVE

1 VOORWOORD

Dit document is een schriftelijke weergave van de stage die Cisse T'Syen bij Ordina Belgium, binnen het derde jaar IT-Factory aan Thomas More Hogeschool Geel, heeft uitgevoerd. Deze stage is een vereiste voor de bachelor Elektronica-ICT te behalen. De stage vindt plaats in het tweede semester van het schooljaar 2021-2022 van 28 februari tot 27 mei.

Allereerst wil ik Ordina bedanken om mij de kans te geven, hier in dit leidend consultancy bedrijf, mij te laten werken met de nieuwste technologieën.
Ik bedank ook de leerkrachten van Thomas More Geel en in het speciaal Dirk Mervis, mijn stagebegeleider, om me te begeleiden doorheen mijn stage.
Natuurlijk wil ik ook de collega's van Jworks bedanken voor de hulp tijdens mijn stage. Hier wil ik vooral mijn stagementoren Duncan Casteleyn en Bart Coppens bedanken voor het vlotte verloop van mijn stage en me altijd te helpen waar nodig. Ook wil ik Frederick Bousson en Nils Devos bedanken voor het bedenken en coördineren van de stageopdracht.

2 TERMINOLOGIE

- Stagebegeleider	Dirk Mervis, stagebegeleider en leerkracht aan Thomas More hogeschool Geel.
- Stagementoren	Duncan Casteleyn, Java Developer en Bart Coppens, Software Developer bij Ordina Belgium
- GNSS	Global navigation satellite system
- SBAS	Satellite Based Augmentation System
- RTK	Real-Time Kinematic, nauwkeurige techniek voor plaatsbepaling
- SparkFun	Fabrikant break-out board
- u-blox	Fabrikant GNSS-module
- SparkFun GPS-RTK-NEO-M8P-2 board	De SparkFun GPS-RTK Board is een krachtige break-out board voor de NEO-M8P-2 module van u-blox
- NEO-M8P-2 module	Hoge nauwkeurigheidsmodule voor GNSS en gps-locatie oplossingen inclusief RTK, gemaakt door U-blox.
- Jworks	Afdeling bij Ordina Belgium.
- RTK-bord	Verwijzing naar SparkFun GPS-RTK-NEO-M8P-2 board.
- U-center	Programma van u-blox voor de module te configureren en testen.
- GUI	Graphical User Interface
- CUI	Character-based User Interface
- MQTT	MQTT (MQ Telemetry Transport) is een lichtgewicht open berichtenprotocol. Verder uitgelegd in hoofdstuk MQTT
- Saas	Software as a service (SaaS) is een software distributie model waarbij een Cloud provider toepassingen host en deze via het internet ter beschikking stelt van eindgebruikers.
- NMEA	National Marine Electronics Association NMEA. NMEA bestond al lang voordat GPS werd uitgevonden. Volgens de NMEA-website werd de vereniging in 1957 opgericht door een groep elektronicadealers om een betere communicatie met de fabrikanten tot stand te brengen.

3 INLEIDING

In dit document ga ik het hebben over de realisatie van mijn stage. Het doel van mijn stage is de verschillende Global Navigation Satellite Systems (GNSSs) en de mogelijke correctiemethodes te onderzoeken en te testen aan de hand van het SparkFun GPS-RTK NEO-M8P-2 board. Dit zodat Ordina een methode heeft voor nauwkeurige plaatbepaling, zodat ze deze aan hun klanten kunnen aanbieden.

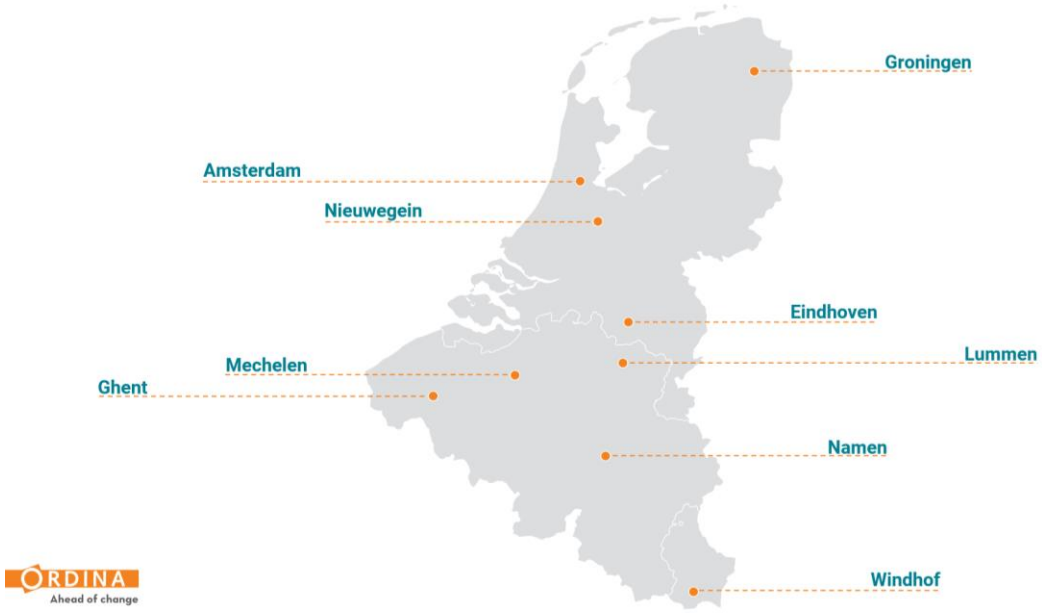
Bij het begin van het onderzoek moest er een plan van aanpak worden opgesteld. Deze is te vinden op mijn persoonlijke portfolio. Hierin staan de grote lijnen die gevolgd moeten worden tijdens het onderzoek. In mijn onderzoek verloopt dit plan van aanpak niet helemaal systematisch, omdat er vaak op een vorige stap moet terug gekeerd worden, om het uiteindelijke resultaat te behalen. Nadat dit plan van aanpak was opgesteld begon de realisatie.

In deze realisatie onderzocht ik eerst de verschillende GNSS-systemen, hierbij kwamen ook de verschillende SBAS-systemen bij kijken. Hierna onderzocht ik de verschillende mogelijkheden om correctiedata, zoals RTK, toe te passen op de SparkFun GPS-RTK NEO-M8P-2 board. Terwijl schoolde ik me bij in de wereld van Cloud, om deze precieze plaatsbepaling data online te krijgen en zo overal bereikbaar te maken.

4 STAGEBEDRIJF ORDINA

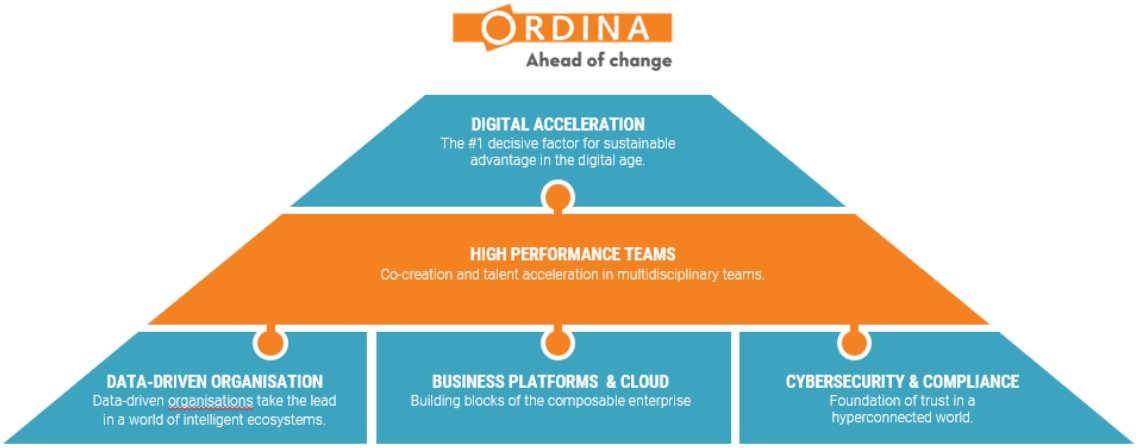
Figuur 1 Logo Ordina

Ordina is opgericht in 1973 en de grootste, lokale aanbieder van IT-diensten in de Benelux, met circa 2.750 medewerkers. Zij richten zich op het geven van een digitale voorsprong aan hun klanten in de volgende sectoren: financiële dienstverlening, overheid, industrie en zorg. Dit doen ze door het ontwerpen, bouwen en beheren van digitale oplossingen. Ordina helpt haar klanten om de uitdagingen en veranderingen in hun business voor te blijven, Ahead of change.



Figuur 2 Vestigingen Ordina

Ze zijn gevestigd op negen plaatsen gespreid over de Benelux. Het hoofdkantoor van België bevindt zich in Mechelen en dat van Nederland in Nieuwegein.



Figuur 3 Ordina business propositie

Ordina heeft vijf business proposities (Figuur 3):

- **High-performance teams** zijn multidisciplinaire coalities van professionals die perfect op elkaar zijn ingespeeld en de relevante technologieën volledig beheersen, van Java tot Microsoft.

- Ordina verhoogt voortdurend uw **veiligheid** en weerbaarheid met een integrale aanpak die rekening houdt met techniek, organisatievraagstukken en persoonlijke aspecten. Daarnaast hanteren zij een agile aanpak, met flexibele teams en gespecialiseerde expertise afgestemd op uw specifieke business professionals.

- Als intelligente**, data gestuurde organisatie** weet u precies wat er in uw omgeving gebeurt. Dankzij de data die u verzamelt, bent u in staat om nieuwe behoeften in de markt te herkennen en uw organisatie te ontwikkelen door deze data sneller en gerichter in te zetten om processen te optimaliseren en voortdurend waarde toe te voegen aan de service die u zowel aan medewerkers als klanten biedt.

- Ordina verbindt **organisaties in de digitale wereld** met klanten of medewerkers via een breed scala aan technische mogelijkheden. Denk aan intuïtieve websites, apps en andere ICT-toepassingen die relevant en waardevol zijn voor de doelgroepen. Dit noemen zij digitale acceleratie.

- Ordina haalt het beste uit uw **businessplatformen**. Ten eerste: ze zorgen ervoor dat ze de continuïteit van uw processen garanderen. Ten tweede: ze optimaliseren de ICT-strategie om maximaal te profiteren van technologische mogelijkheden, kosten te beheersen en platformen up-to-date en rendabel te houden. Ten derde: ze zorgen ervoor dat uw business platformen veelzijdig genoeg zijn om mee te evolueren met uw organisatie en het ecosysteem waarin u opereert.

Figuur 4 Units Ordina

Ten slotte wil ik het nog even hebben over de verschillende units die zich bevinden binnen in Ordina. In bovenstaande afbeelding (Figuur 4) kan u deze allemaal terugvinden. Voornamelijk ga ik het hebben over de unit waar ik stage heb gedaan genaamd JWorks.

JWorks is een Java & Javascript development afdeling die full stack, backend en AWS-Solutions programmeert. Ze gebruiken technische en culturele trends zoals DevOps, immutable infrastructure, extreme programming en microservices om snel te ontwikkelen, met een time-to-market sneller dan ooit. Als officiële bijdragers van Netflix ademen ze open source. Wanneer ze softwareoplossingen ontwikkelen voor hun klanten, gebruiken ze opensource tools zoals Spring Boot, Spinnaker, Kubernetes en Cloud Foundry.

Ze houden ook van experimenteren met Internet of Things. Zo maken ze prototypes met Arduino’s en Raspberry Pi’s en konden ze al voor toonaangevende bedrijven projecten uitwerken, zoals de ontwikkeling van het LoRa-platform bij Proximus.

5 GEBRUIKTE TECHNOLOGIEËN

5.1 Microsoft Azure

Microsoft Azure, gewoonlijk Azure genoemd, is een Cloud computing dienst gecreëerd door Microsoft voor het bouwen, testen, implementeren en beheren toepassingen en diensten via door Microsoft beheerde datacenters.

Figuur 5 Azure logo

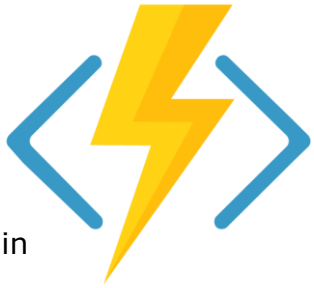
5.1.1 Azure IoT Hub

Azure IoT Hub maakt zeer veilige en betrouwbare communicatie mogelijk, doormiddel van MQTT, tussen Internet of Things (IoT)-toepassingen en de apparaten die deze beheert. Azure IoT Hub biedt een Cloud-hosted oplossing backend om elk apparaat virtueel te verbinden. Deze wordt gebruikt om de GNSS-data naar de Cloud te sturen doormiddel van MQTT (zie hoofdstuk: MQTT).

Figuur 6 Logo Azure IoT Hub

5.1.2 Azure Functions

Azure Functions is een gebeurtenis gestuurd, serverloos rekenplatform dat kan helpen om indelingsproblemen op te lossen. Deze wordt tijdens de stage gebruikt om de data van de IoT Hub te nemen en deze door te sturen naar de database zodat deze data wordt opgeslagen, dit gebeurt in Javascript.



Figuur 7 Logo Azure Functions

5.1.3 Azure Cosmos DB-API for MongoDB

De Azure Cosmos DB API voor MongoDB maakt het gemakkelijk om Cosmos DB te gebruiken alsof het een MongoDB database is. U kunt gebruikmaken van uw ervaring met MongoDB en uw favoriete MongoDB-stuurprogramma's, SDK's en tools blijven gebruiken door uw toepassing te verwijzen naar de API voor MongoDB-accountconnectiestring. Deze wordt tijdens de stage gebruikt om de GNSS-data op te slagen.

Figuur 8 Logo Azure Cosmos DB-API for MongoDB

5.1.4 Azure Key Vault

Azure Key Vault is een Cloud service voor veilige opslag van en toegang tot geheimen. Een geheim is alles waar u de toegang tot wilt controleren, zoals API-sleutels, wachtwoorden, certificaten of crypto grafische sleutels. Deze Key Vault is aangemaakt tijdens de stage, maar niet meer verder op ingegaan omdat de opstelling in een test fase werd gebruikt. Buiten deze test fase moet dit wel werken wegens veiligheidsredenen.

Figuur 9 Logo Azure Key Vault

5.2 Microsoft Azure DevOps

Azure DevOps biedt ontwikkelaarsdiensten om teams in staat te stellen hun werk te plannen, samen te werken aan de ontwikkeling van code en applicaties te bouwen en uit te rollen. Azure DevOps ondersteunt een collaboratieve cultuur en een reeks processen die ontwikkelaars, projectmanagers en bijdragers samenbrengen om software te ontwikkelen. Deze wordt in mijn stage gebruikt voor het opstellen van boards waar work items inkomen die moeten uitgewerkt worden. Ook wordt het Git gedeelte gebruikt om al de code op te slagen.

Figuur 10 Azure DevOps logo

5.3 MQTT

MQTT (MQ Telemetry Transport) is een lichtgewicht open berichtenprotocol dat netwerk cliënten met beperkte middelen een eenvoudige manier biedt om telemetrie-informatie te verspreiden in omgevingen met weinig bandbreedte. Ideaal voor Internet of Things doeleinden. Dit protocol wordt gebruikt tijdens de stage om data van de Raspberry Pi door te sturen naar de Cloud.

Figuur 11 Logo MQTT

5.4 Java Spring Boot

Java Spring Boot ook wel Spring Boot is een open source Java-gebaseerd framework dat gebruikt wordt om een micro Service te creëren. Het is ontwikkeld door Pivotal Team en wordt gebruikt om standalone en productie-klare Spring applicaties te bouwen. Door Spring Boot is het gemakkelijker en sneller om webapplicaties en microservices te ontwikkelen. Dit wordt gebruikt tijdens de stage om de backend te ontwikkelen. In deze backend wordt er data uit de database gehaald in de toekomst kan er met deze data een JSON-API gemaakt worden voor verdere front-end doeleinden.

Figuur 12 Logo Spring Boot

5.5 Terraform

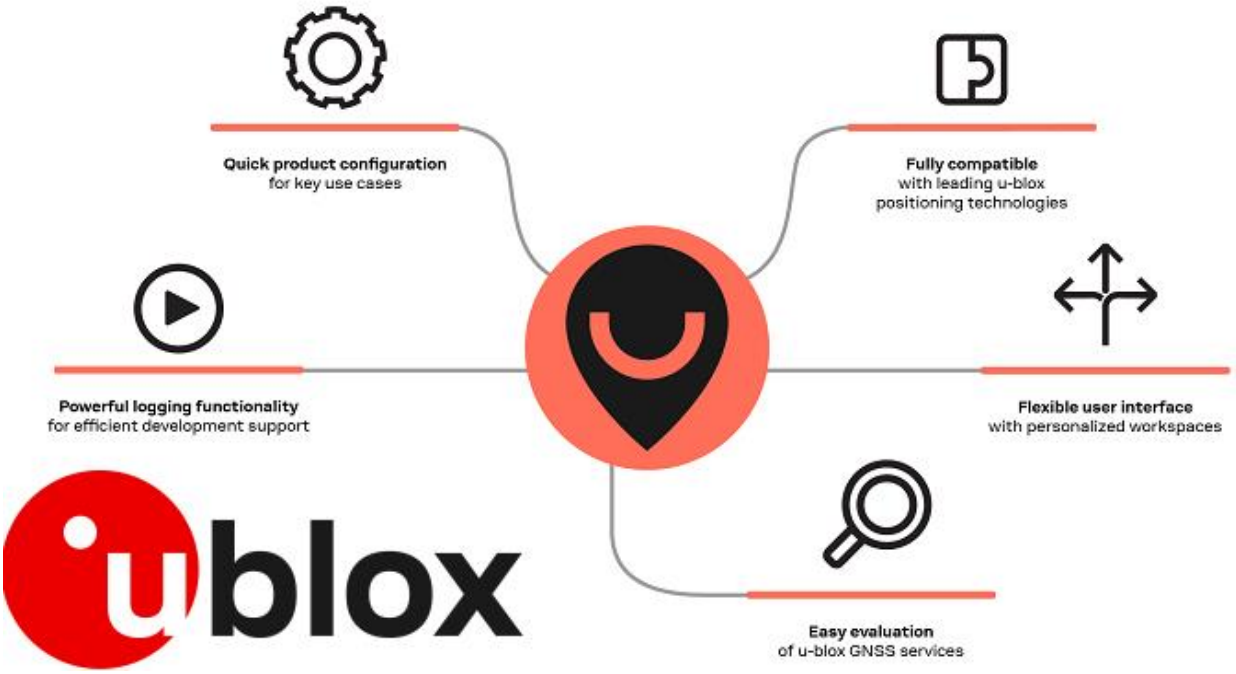
HashiCorp Terraform is een infrastructure as code-tool waarmee u zowel Cloud- als lokaal resources kunt definiëren in leesbare configuratiebestanden die u kunt bewerken, hergebruiken en delen. Vervolgens kunt u een consistente workflow gebruiken om uw gehele infrastructuur gedurende de gehele levenscyclus te provisioneren en beheren. Terraform kan low-level componenten zoals computing, storage en networking resources beheren, alsook high-level componenten zoals DNS entries en SaaS-features. Hier ben ik

niet heel ver op ingegaan tijdens mijn stage ik heb mijn infrastructuur geïmporteerd waardoor ik deze altijd zou kunnen opspinnen als er iets misloopt.

Figuur 13 Logo Terraform

5.6 u-center

u-center is een GNSS-evaluatiesoftware die gemakkelijk te gebruiken en te personaliseren is en compatibel is met toonaangevende u-blox-technologieën. u-center 2, de volgende generatie van de software, is ontworpen voor het volgen van activa en draagbare apparaten. u-center wordt gebruikt tijdens de stage om de module te configureren in de test fase. u-center 2 wordt tijdens de stage vooral gebruikt om de data te visualiseren.



Figuur 14 u-center

5.7 Andere

Volgende technologieën worden in verdere hoofdstukken besproken:

- Real-Time Kinematic (RTK)
- Global navigation satellite system (GNSS)
- Satellite Based Augmentation System (SBAS)
- SparkFun GPS-RTK Board - NEO-M8P-2
- Raspberry Pi
- RTKLIB

6 HARDWARE

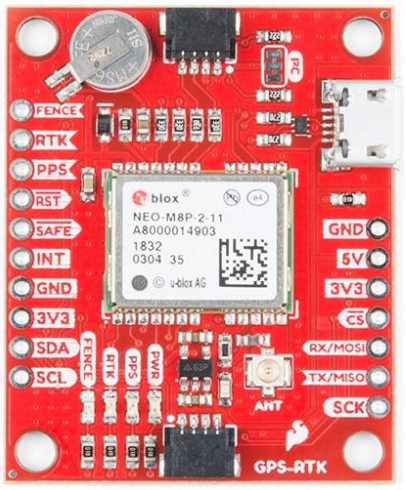
6.1 SparkFun GPS-RTK Board - NEO-M8P-2

Het SparkFun GPS-RTK Board is een krachtig breakout board voor de NEO-M8P-2 module van U-blox. De NEO-M8P-2 is een hoog nauwkeurige module voor het gebruik van GNSS en gps-locatie oplossingen zoals RTK. Deze module is uniek omdat deze ingesteld kan worden als rover maar ook als base station.

Figuur 15 SparkFun GPS-RTK Board

6.1.1 Communicatiepoorten

De NEO-M8P-2 is uniek omdat hij vier communicatiepoorten heeft die alle vier gelijktijdig actief zijn. Je kunt NMEA-data lezen over I²C terwijl je configuratie commando's over de UART stuurt en vice versa. De enige beperking is dat de SPI pinnen zijn toegewezen aan de I²C en UART pinnen, dus het is of SPI of I²C+UART. De USB-poort is te allen tijde beschikbaar.

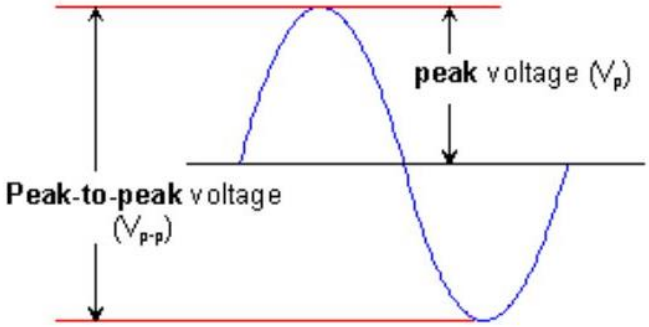


Figuur 16 SparkFun GPS-RTK Board geheel

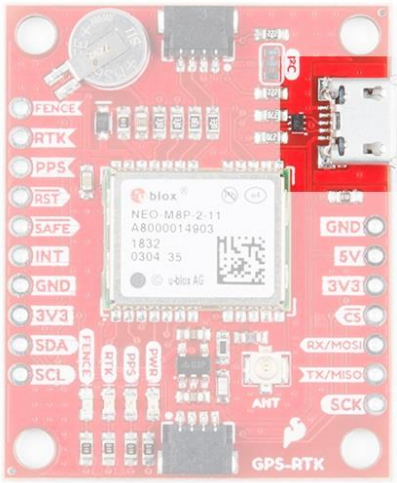
6.1.2 USB

De micro-B connector maakt het eenvoudig om de NEO-M8P aan te sluiten op u-center voor configuratie en snelle weergave van NMEA-zinnen. Het is ook mogelijk om een Raspberry Pi over USB aan te sluiten.

Een 3,3V regelaar is voorzien om de 5V USB te regelen tot 3,3V die de module nodig heeft. Externe 5V kan worden toegepast of een directe voeding van 3,3V kan worden verstrekt. Als u de printplaat rechtstreeks van 3,3 V voorziet, moet dit een schone voeding met minimale ruis zijn (minder dan 50 mV VPP-rimpel is ideaal voor nauwkeurige plaatsbepaling). 50mV VPP-rimpel betekend dat de piek tot piekwaarde van het ruis signaal niet groter mag zijn dan 50mV.



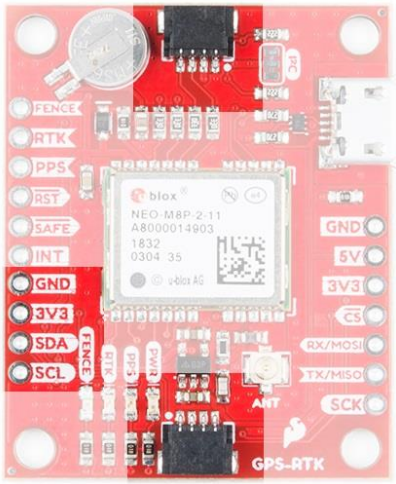
Figuur 17 VPP-ruis signaal



Figuur 18 SparkFun GPS-RTK Board USB

6.1.3 I²C

Alle functies zijn toegankelijk via de I²C poorten inclusief het lezen van NMEA-zinnen, het zenden van UBX-configuratie strings, het pipen van RTCM-data naar de module, etc. Het pipen van RTCM-data doe ik via seriële communicatie omdat dit niet gaat via RTKLIB, de tool dat ik gebruik om correctiedata door te sturen naar de module (zie hoofdstuk: RTK-data connecteren met RTKLIB). SparkFun heeft ook 2 Qwiic connectoren dit systeem zorgt ervoor dat er gemakkelijk gebruik kan gemaakt worden van I²C zonder pinnen aan de printplaat te moeten solderen. Ik maak gebruik van de gewone I²C poorten waar ik de accurate positie data uit aflees met een Raspberry Pi.

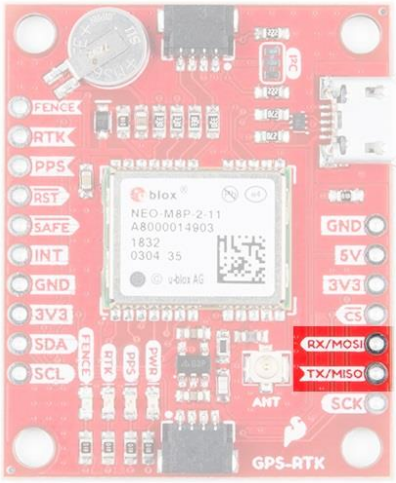


Figuur 19 SparkFun GPS-RTK Board I²C

6.1.4 UART/Serieel

De klassieke seriële pinnen zijn beschikbaar op de NEO-M8P maar worden gedeeld met de SPI pinnen. Zorg hiervoor dat de DSEL-jumper op de achterkant van de printplaat openstaat.

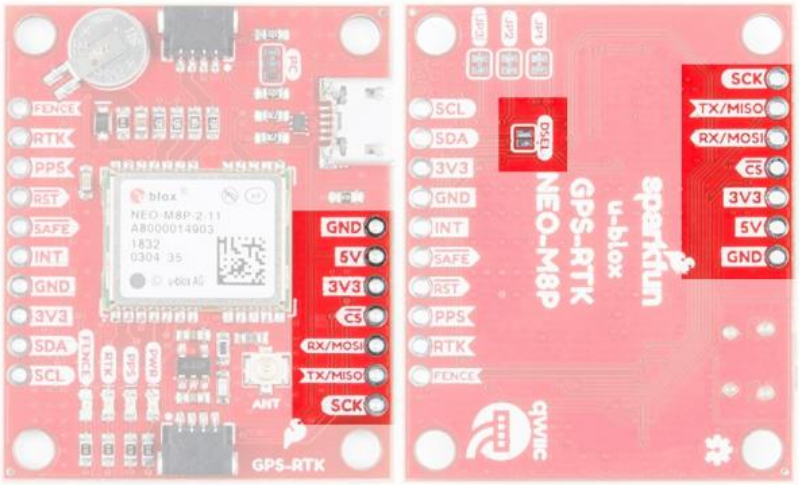
- MISO = TX uit de NEO-M8P
- MOSI = RX naar NEO-M8P



Figuur 20 SparkFun GPS-RTK Board UART

6.1.5 SPI

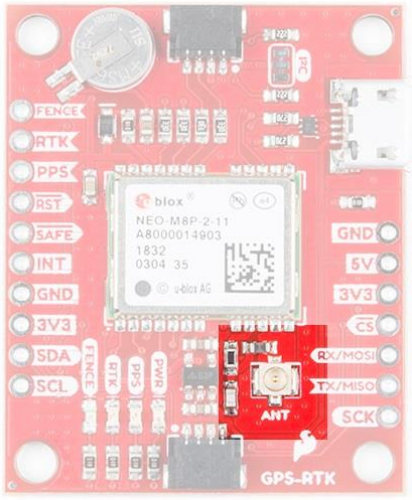
De NEO-M8P kan ook worden geconfigureerd voor SPI-communicatie. Standaard is de SPI-poort uitgeschakeld. Om SPI in te schakelen sluit je de DSEL-jumper op de achterkant van de print. Het sluiten van deze jumper schakelt de UART en I²C interfaces uit.



Figuur 21 SparkFun GPS-RTK Board SPI

6.1.6 Antenne

In onderstaande afbeelding wordt de aansluiting van de antenne weergegeven doormiddel van een U. FL connector.

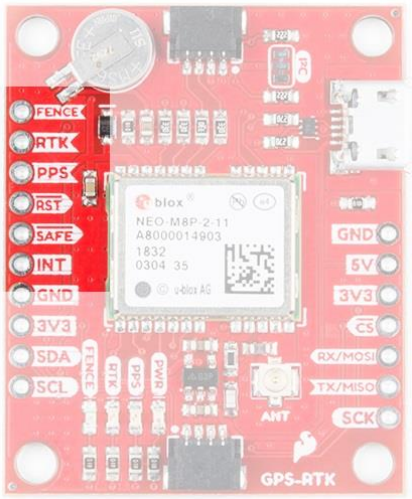


Figuur 22 SparkFun GPS-RTK Board Antenne aansluiting

6.1.7 Controlepinnen

Deze pennen worden gebruikt voor diverse extra aansturingen van de NEO-M8P:

- FENCE: Geofence uitgangspin. Geconfigureerd met U-Center. Zal hoog of laag gaan wanneer een geofence is ingesteld. Nuttig voor het triggeren van alarmen en acties wanneer de module een geprogrammeerde perimeter verlaat.
- RTK: RTK-uitgangspin. Blijft hoog wanneer de module normale gps-wijze is. Begint te knipperen wanneer RTCM-correcties worden ontvangen en de module in RTK float mode gaat. Gaat laag wanneer de module in RTK vaste modus gaat en nauwkeurige locaties op cm-niveau begint uit te zenden.
- PPS: Puls-per-seconde uitgangspin. Begint te knipperen op 1Hz wanneer de module basis GPS/GNSS positievergrendeling krijgt.
- RST: Reset ingangspin. Trek deze lijn laag om de module te resetten.
- SAFE: Safeboot ingangspin. Deze is nodig voor firmware updates van de module en dient in het algemeen niet gebruikt of aangesloten te worden.
- INT: Interrupt input/output pin. Kan worden gevormd gebruikend U-Center om de module uit diepe slaap te brengen of om een onderbreking voor diverse modulestaten uit te voeren.



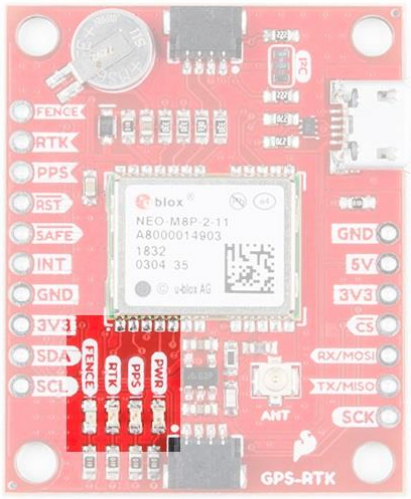
Figuur 23 SparkFun GPS-RTK Board controlepinnen

6.1.8 LEDs

Het bord bevat vier status-LED's, zoals aangegeven in de onderstaande afbeelding.

- De voedings-LED (PWR) gaat branden wanneer 3,3 V wordt geactiveerd via USB, via de Qwiic-bus of de 3v3 pin.
- De puls per seconde (PPS) LED zal oplichten bij elke succesvolle update zodra een positievergrendeling is bereikt.

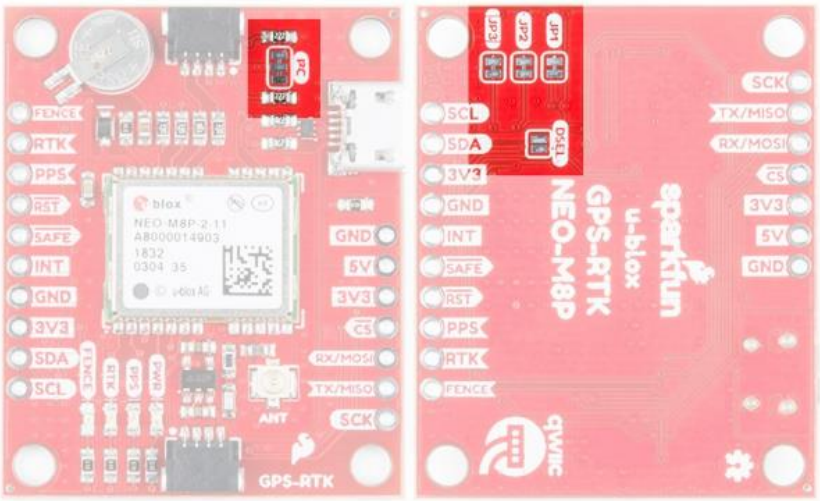
- De RTK LED brandt constant bij het inschakelen. Zodra RTCM-data succesvol is ontvangen zal deze beginnen te knipperen. Dit is een goede manier om te zien of de NEO-M8P RTCM ontvangt van verschillende bronnen.
- De FENCE LED kan worden geconfigureerd om aan/uit te gaan voor geofencing toepassingen.



Figuur 24 SparkFun GPS-RTK Board LEDs

6.1.9 Jumpers

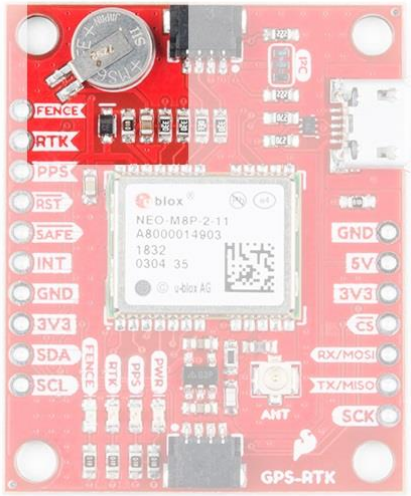
Er zijn in totaal 4 jumpers te vinden aan de achterzijde van de printplaat. Door DSEL te sluiten wordt de SPI-interface ingeschakeld en de UART- en I²C-interfaces uitgeschakeld. USB zal nog steeds functioneren. Het doorknippen van de I²C-jumper verwijdert de 2,2k Ohm weerstanden van de I²C-bus. Als u veel apparaten op uw I²C bus heeft, kunt u deze jumpers beter verwijderen. De jumpers JP1, JP2, JP3 zijn aan de achterzijde van het bord aangebracht om de verschillende status-LED's te kunnen isoleren.



Figuur 25 SparkFun GPS-RTK Board jumpers

6.1.10 Back-up batterij

De MS621FE oplaadbare batterij onderhoudt de RAM (BBR) op de NEO-M8P. Dit maakt een veel snellere positievergrendeling mogelijk. De BBR wordt ook gebruikt voor module configuratie retentie. De batterij wordt automatisch opgeladen wanneer de stroom wordt ingeschakeld en zou de instellingen en GNSS-baangegevens tot twee weken zonder stroom moeten behouden.



Figuur 26 SparkFun GPS-RTK Board back-up batterij

6.2 Antenne

Eerst had ik een goedkope antenne gekregen die ergens op het kantoor lag. Met deze goedkope antenne waren de signalen die ik ontving zwak, onnauwkeurig en duurde lang voordat ik deze ontving. Hierdoor was het testen een hele uitdaging en moeilijk om te doen. Daarom is een goede nauwkeurige antenne, zoals de ANN-MB-00 GNSS multiband antenne, noodzakelijk.

Figuur 27 Goedkope antenne

6.2.1 ANN-MB-00 GNSS multiband antenne

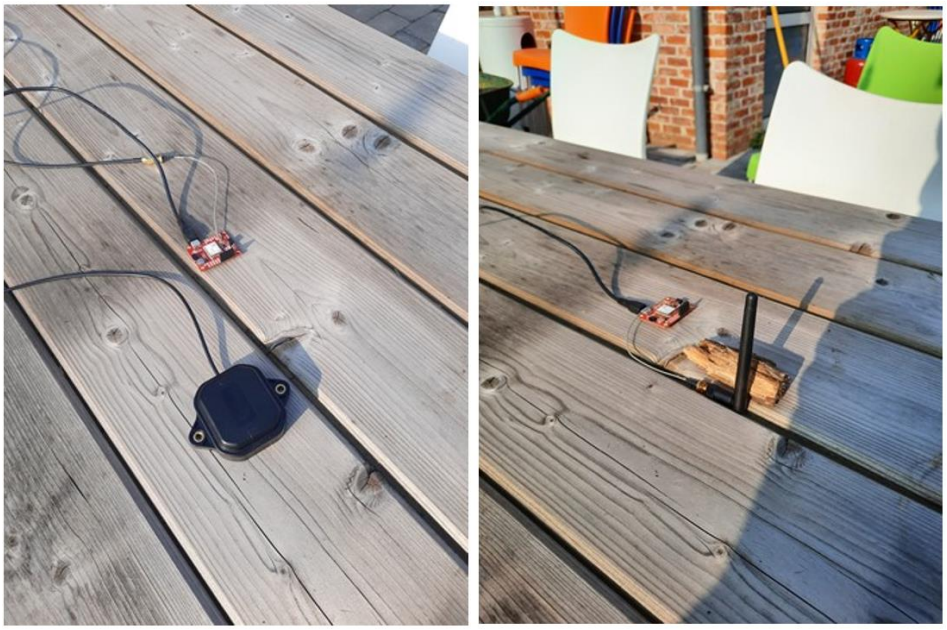
De ANN-MB-00 GNSS multiband antenne is uniek ten opzichte van andere GNSS/GPS antennes omdat hij ontworpen is om zowel de klassieke L1 GPS band als de vrij recent gelanceerde L2 GPS band te ontvangen.

Verder is de u-Blox ANN-MB-00 goed geconstrueerd met een magnetische basis met bevestigingsgaten voor extra verankering voor de ruwste omgevingen. Hij is geschikt voor gebruik met elke GPS/GNSS ontvanger met dubbele L1/L2 ontvangst en ondersteunt GPS, GLONASS, Galileo en BeiDou.

Figuur 28 ANN-MB-00 GNSS multiband antenne

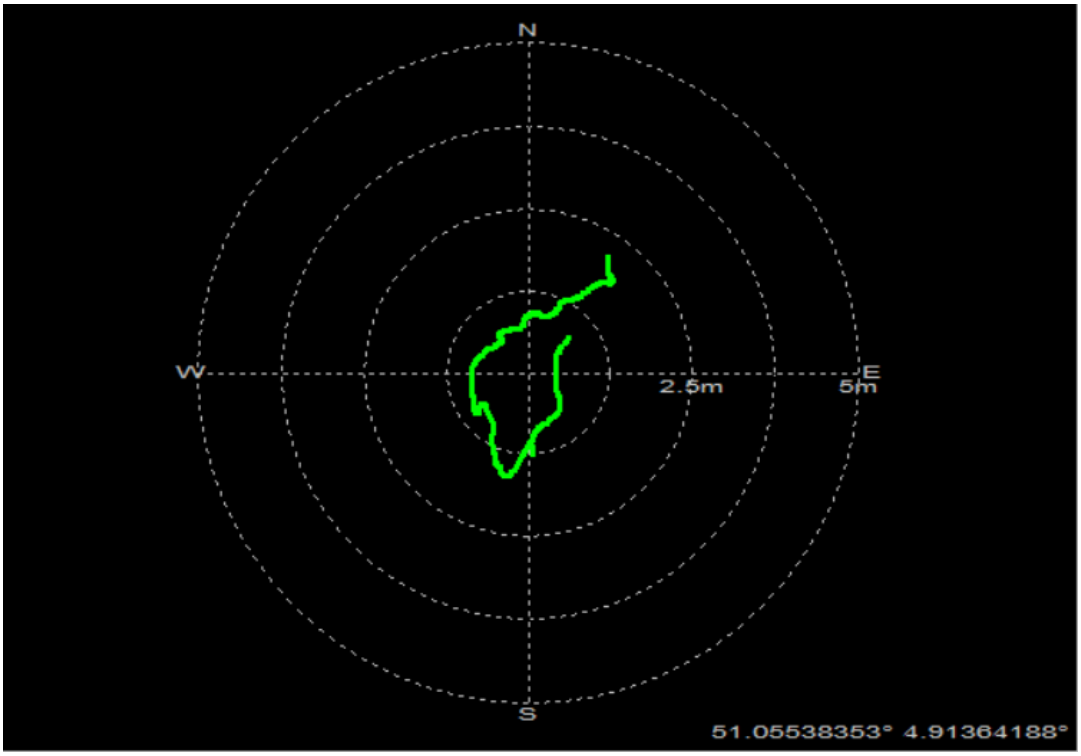
6.2.2 Testen van de antennes

Ik heb beide antennes 5 minuten naast elkaar getest terwijl ze buiten op de tafel lagen, met als correctiemethode SBAS (zie hoofdstuk: SBAS). Dit zijn de resultaten (hoe compacter de datapunten hoe nauwkeuriger):



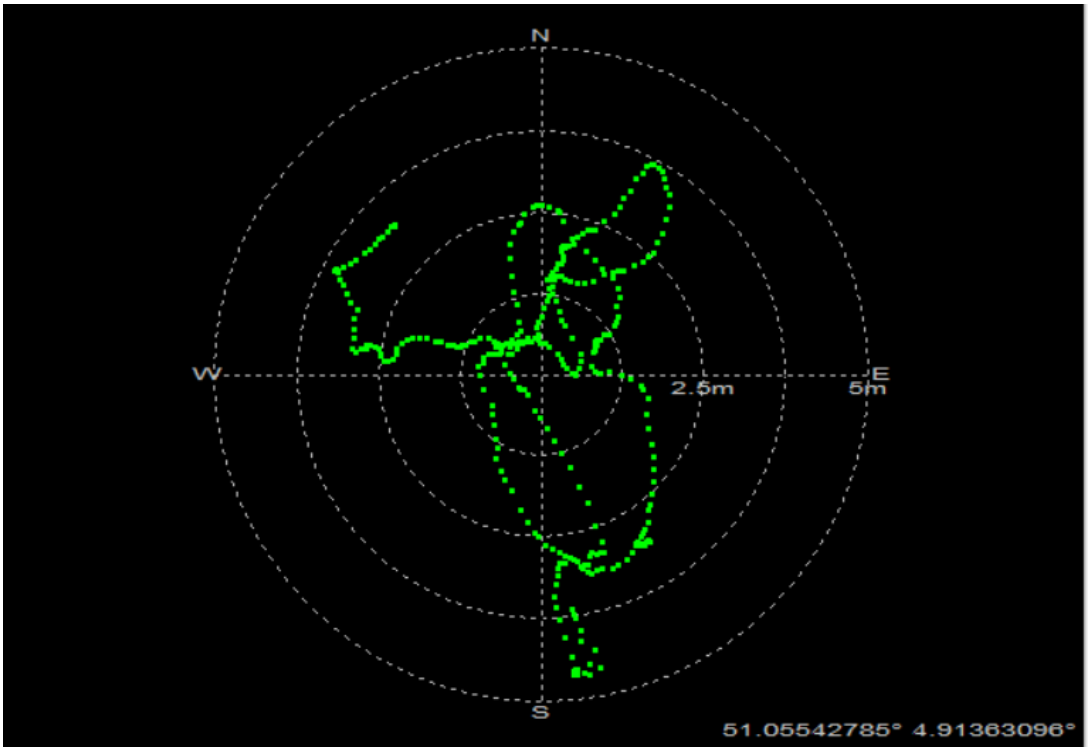
Figuur 29 Antennes buiten testen

- U-blox ANN-MB-00-00 (nieuwe antenne):
    - Je kunt zien dat de antenne binnen een bereik van max 2,5m blijft als hij stilstaat. Hij kreeg meteen verbinding toen ik de module startte.



Figuur 30 Test antenne U-blox ANN-MB-00-00

- Onbekende goedkope antenne:
    - Ten eerste duurde het veel langer om een verbinding te maken, een paar minuten in plaats van enkele seconden. De datapunten liggen verder uit elkaar, dat betekent dat er veel speling zit tussen elk datapunt en dus minder nauwkeurig is. Terwijl je bij de andere antenne kunt zien dat ze veel dichter bij elkaar liggen. Als laatste kun je ook zien dat het bereik van de antenne over de 2,5 meter is gegaan.

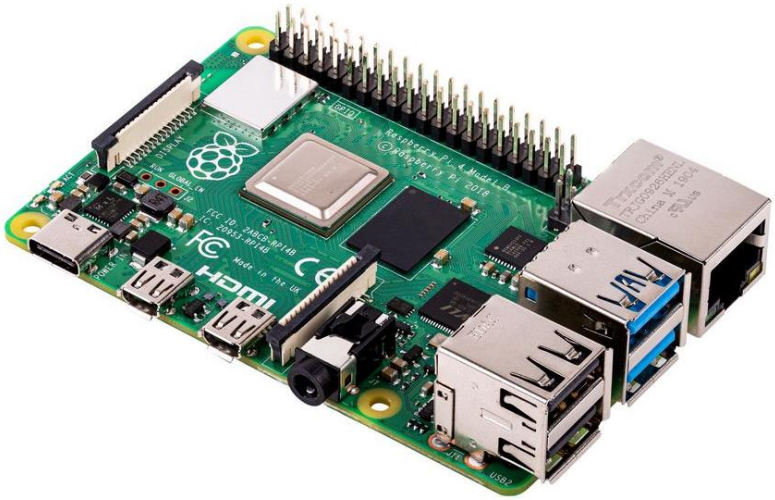


Figuur 31 Test goedkope antenne

Hieruit kunnen we concluderen dat het noodzakelijk is een goede antenne te hebben als we nauwkeurig willen werken.

6.3 Raspberry Pi 4B

De Raspberry Pi is een kleine computer met uitwendige I/O pinnen. Deze kunnen gebruikt worden om te connecteren met verschillende sensoren en andere modules. Het model dat gebruikt wordt in tijdens de stage is een Raspberry Pi 4B met 4G ram. Dit model heeft een ingebouwde WIFI-module, dit is handig om de Raspberry Pi draadloos te verbinden en om draadloos RTK-correctiedata te ontvangen (zie hoofdstuk RTK). Deze Raspberry Pi wordt verbonden met de SparkFun GPS-RTK Board - NEO-M8P-2, de data die wordt uitgelezen wordt doorgestuurd naar de Cloud.



7 ONDERZOEK EN TESTEN

7.1 GNSS

7.1.1 Wat is GNSS

Global Navigation Satellite System (GNSS) verwijst naar een groep van satellieten die vanuit de ruimte signalen uitzenden die plaats- en tijdsbepalingsgegevens doorgeven aan GNSS-ontvangers. De ontvangers gebruiken deze gegevens vervolgens om de locatie te bepalen, GNSS biedt wereldwijde dekking.

De prestaties van GNSS worden beoordeeld aan de hand van vier criteria:

1. Nauwkeurigheid: het verschil tussen de gemeten en de werkelijke positie, snelheid of tijd van een ontvanger.
2. Integriteit: het vermogen van een systeem om een betrouwbaarheidsdrempel te bieden en, in geval van een afwijking in de positiebepalingsgegevens, een alarmsignaal te geven.
3. Continuïteit: het vermogen van een systeem om zonder onderbreking te functioneren.
4. Beschikbaarheid: het percentage van de tijd dat een signaal voldoet aan de bovengenoemde criteria voor nauwkeurigheid, integriteit en continuïteit.

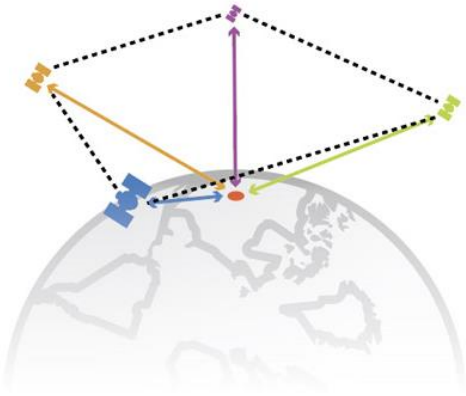
7.1.2 Voorbeelden GNSS

- Galileo van Europa
- NAVSTAR Global Positioning System (GPS) van VS
- Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) van Rusland
- Beidou van China

7.1.3 GPS

Er zijn ongeveer 32 GPS satellieten in de ruimte. Dit is een relatief groot systeem dat werkt door middel van zorgvuldige planning en ijking. De locatie wordt berekend door de ontvanger, dit door middel van de afstand en tijd tussen hem en de satelliet. Deze berekening gaat als volgt, de snelheid van het signaal (lichtsnelheid) wordt vermenigvuldigd met de atoomklok tijd van de satelliet. De locatie kan al bepaald worden aan de hand van drie satellieten, maar dit is niet nauwkeurig. Er zijn in totaal vier satellieten nodig om een nauwkeurige locatiebepaling te krijgen. De eerste drie satellieten worden gebruikt om de locatie te bepalen in drie dimensies x-, y- en z-coördinaten. De

vierde satelliet wordt gebruikt om de tijd te bepalen die het signaal nodig heeft om van de satelliet naar de ontvanger te reizen, zoals in onderstaande figuur.



Figuur 32 Gps-diagram

7.1.4 Galilleo

Deze GNSS wordt verduidelijkt omdat dit het GNSS van Europa is. Eerst was het gepland om dit systeem te gebruiken met het RTK-bord, maar na veel problemen zag ik in één van de datasheets dat Galileo nog niet ondersteund werd. Dit omdat Galileo nog in een ontwikkel fase zat toen het RTK-bord werd geproduceerd.

Galileo is Europa's Global Navigation Satellite System (GNSS), deze levert verbeterde plaats- en tijdsbepalingsinformatie. Dit heeft positieve gevolgen voor veel Europese diensten en gebruikers zoals:

- Dankzij Galileo kunnen gebruikers hun exacte positie kennen met een grotere precisie dan wat andere beschikbare (GNSS) systemen bieden.
- De producten die mensen elke dag gebruiken, van het navigatietoestel in uw auto tot een mobiele telefoon, profiteren van de grotere nauwkeurigheid die Galileo biedt.
- Hulpdiensten maken gebruik van Galileo
- Galileo zou de europese wegen en spoorwegen veiliger en effeciënter maken.

Bovendien biedt Galileo Europa en de Europese burgers onafhankelijkheid en soevereiniteit, een reeks milieuvoordelen en verschillende nieuwe diensten die specifiek zijn voor het Galileo-programma (open dienst, commerciële dienst, opsporing en redding). In tegenstelling tot andere Global Navigation Satellite Systems is Galileo onder burgercontrole.

Het Galileo-systeem zal, zodra het volledig operationeel is, wereldwijd vier hoogwaardige diensten aanbieden:

1. **Open Service (OS)**: Galileo open en gratis dienst die is opgezet voor plaatsbepalings- en tijdsbepalingsdiensten. In de toekomst zal de open

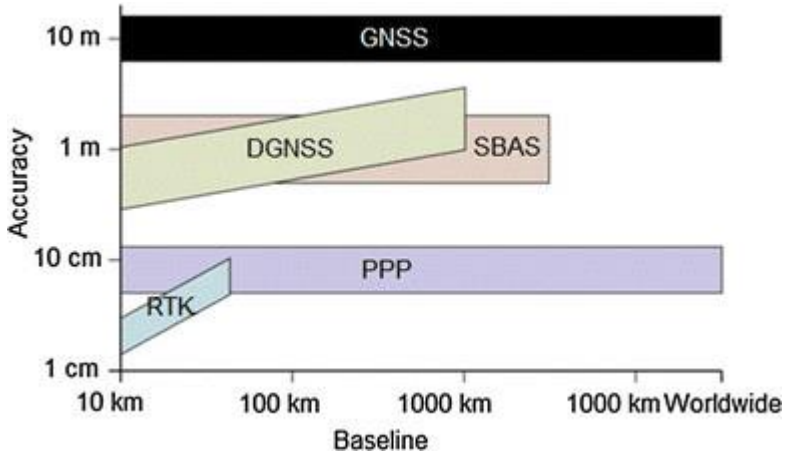
dienst van Galileo ook navigatieboodschapauthenticatie bieden, waarmee de positie van de gebruiker kan worden berekend aan de hand van geauthenticeerde gegevens die uit het navigatiebericht zijn geëxtraheerd.

2. **High Accuracy Service (HAS)**: Een dienst die het OS aanvult door een extra navigatiesignaal en diensten met toegevoegde waarde te leveren in een andere frequentieband. Het HAS-signaal kan worden gecodeerd om de toegang tot de HAS-diensten van Galileo te controleren.
3. **Public Regulated Service (PRS)**: Dienst die beperkt is tot door de overheid gemachtigde gebruikers, voor gevoelige toepassingen die een hoog niveau van dienstcontinuïteit vereisen.
4. **Search and Rescue Service (SAR)**: De Europese bijdrage aan COSPAS-SARSAT, een internationaal satellietgebaseerd opsporings- en reddingsalarmsysteem in noodsituaties.

7.1.5 Verbetering nauwkeurigheid GNSS

De nauwkeurigheid van een GNSS wordt beïnvloed door talrijke factoren, waarvan atmosferische interferentie de belangrijkste is, aangezien signalen door de ruimte en in de atmosfeer van de aarde terechtkomen. Zonder rekening te houden met deze fouten zullen we onnauwkeurigheden zien in de posities die zijn uitgezonden.

In onderstaande grafiek ziet u een aantal verschillende methodes die de prestaties van het GNSS kunnen verbeteren. We gaan vooral ingaan op Real Time Kinematics (RTK) en Satellite Based Augmentation System (SBAS), omdat RTK de nauwkeurigste is en SBAS een van de bekendste.



Figuur 33 Grafiek prestatie verbetering methodes

7.2 SBAS

De prestaties van wereldwijde satellietnavigatiesystemen (GNSS's) kunnen worden verbeterd door regionale Satellite Based Augmentation System (SBAS). SBAS verbetert de nauwkeurigheid en betrouwbaarheid van GNSS-informatie door signaalmeetfouten te corrigeren en door informatie te voorzien over de nauwkeurigheid, integriteit, continuïteit en beschikbaarheid van zijn signalen.

SBAS maakt gebruik van GNSS-metingen die worden verricht door nauwkeurig gelokaliseerde referentiestations die over een heel continent zijn verspreid. Alle gemeten GNSS-fouten worden doorgegeven aan een centraal rekencentrum, waar differentiële correcties en integriteitsberichten worden berekend. Deze berekeningen worden vervolgens uitgezonden over het bestreken gebied met behulp van geostationaire satellieten (satelliet die zich steeds boven dezelfde plaats op aarde bevindt) die dienen als een aanvulling, of overlay, op het oorspronkelijke GNSS-bericht. SBAS kan nauwkeurig zijn tot op ongeveer 2m.

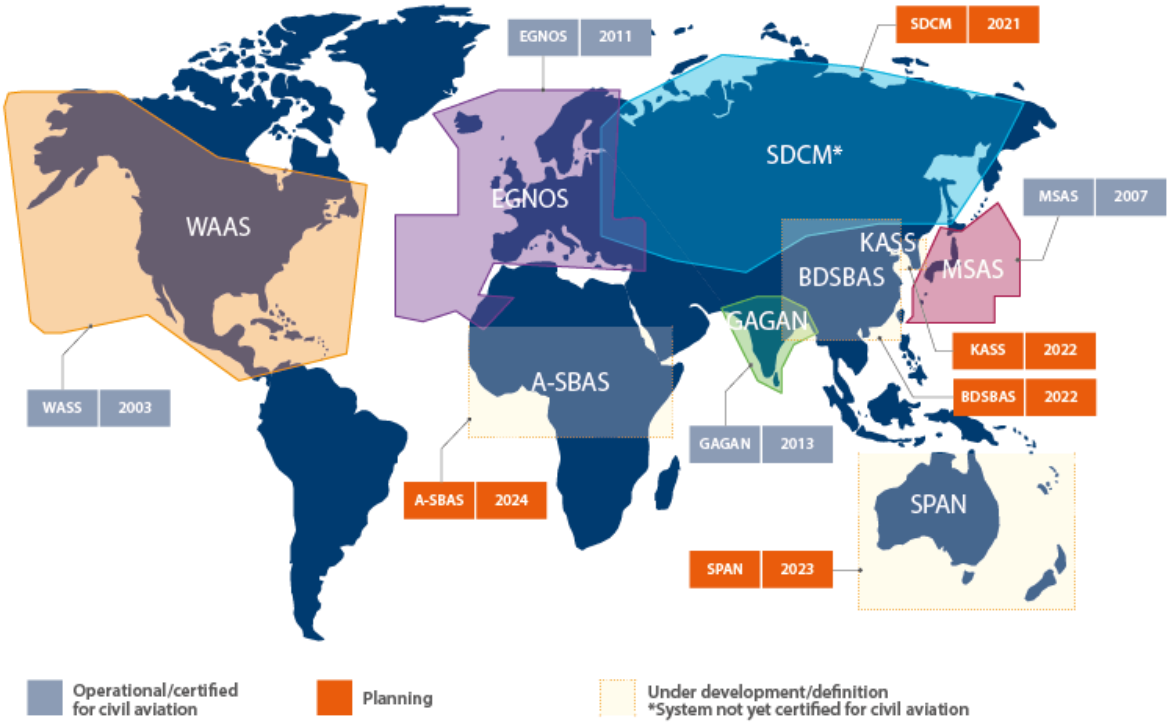
7.2.1 Bestaande SBAS

Verschillende landen hebben hun eigen Satellite Based Augmentation System ingevoerd. In Europa bijvoorbeeld bereikt EGNOS het grootste deel van de Europese Unie (EU), samen met een aantal aangrenzende landen en regio's.

Andere nationale SBAS'en:

- Europa: Europese overlaydienst voor geostationaire navigatie (EGNOS)
- VS: Wide Area Augmentation System (WAAS)
- Japan: Michibiki Satellite Augmentation System (MSAS)
- India: GPS-ondersteunde GEO-navigatie (GAGAN)
- China: BeiDou SBAS (BDSBAS) (in ontwikkeling)
- Zuid-Korea: Korea Augmentation Satellite System (KASS) (in ontwikkeling)
- Rusland: Systeem voor Differentiële Correcties en Monitoring (SDCM) (in ontwikkeling)
- ASECNA: SBAS voor Afrika en de Indische Oceaan (A-SBAS) (in ontwikkeling)
- Australië en Nieuw-Zeeland: Southern Positioning Augmentation Network (SPAN) (in ontwikkeling)

(EUSPA (© EUSPA 2021), 2021)



Figuur 34 Verschillende SBAS

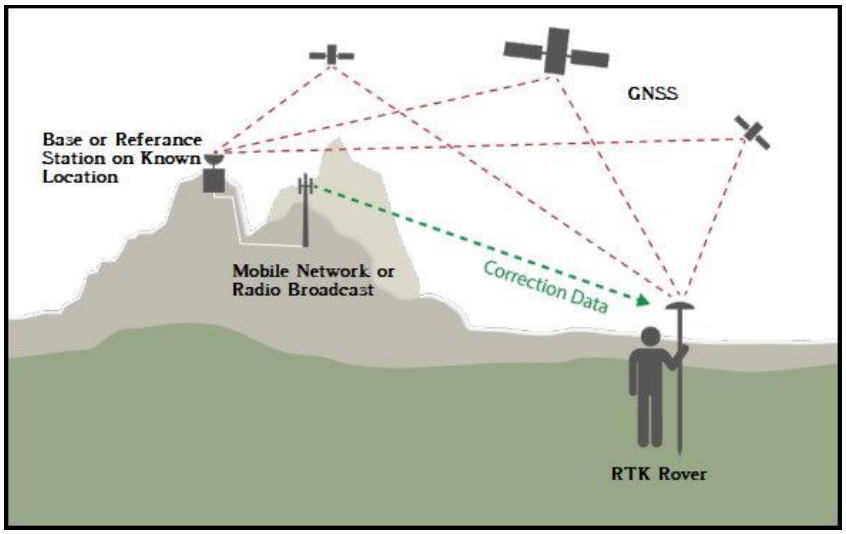
7.2.2 Testen SBAS

Voor bepaalde toepassingen is SBAS al nauwkeurig genoeg. Dit heb ik getest doormiddel van de antenne op mijn auto te bevestigen en een test rit te maken. Dit is te zien in de volgende video: https://youtu.be/kG72kCYS0KY . In deze video is te zien dat de nauwkeurigheid al vrij goed is met SBAS, de wijzer blijft vrij goed op de weg. De video is versneld omdat hij anders te lang was. Onderstaande afbeeldingen geven de opstelling weer.

Figuur 35 Opstelling SBAS test

7.3 RTK

RTK staat voor Real-Time Kinematic en is een techniek dat gebruikt maakt van afstandsbepalingen op basis van draaggolven. Deze neemt de gewone signalen van GNSS tezamen met een correctiedata stroom en berekend zelf de precieze locatie. Hierdoor kunnen we een nauwkeurigheid bereiken tot op 1cm.



Figuur 36 RTK

7.3.1 Berekening van het bereik

Op een basis conceptueel niveau is het bereik berekend door het bepalen van de hoeveelheid draaggolven tussen de satelliet en de rover, deze wordt dan vermenigvuldigt met de lengte van de draaggolf.

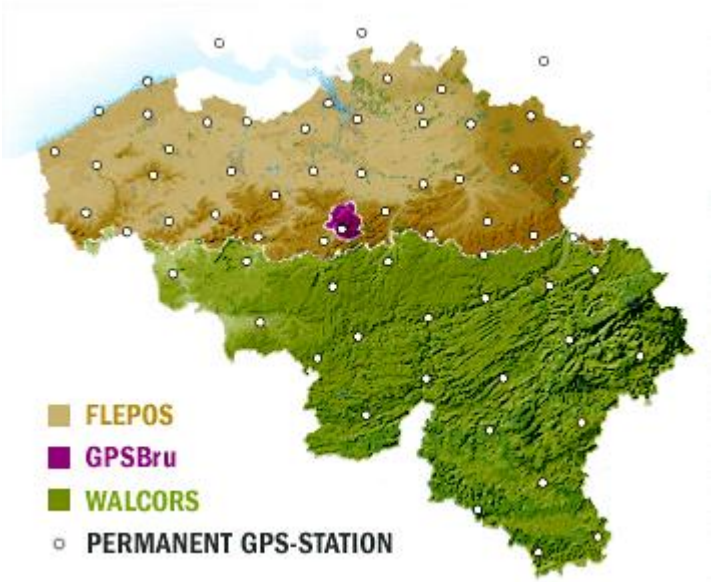
In dit bereik zitten nog steeds fouten, deze fouten worden veroorzaakt door de satelliet klok en efemeriden maar eveneens door ion sferische en troposferische vertragingen. Dit kunnen we oplossen door correctiedata van het basisstation naar de rover te sturen (figuur 7) over mobiel netwerk of een andere soort van draadloze communicatie zoals LoRa.

7.3.2 Netwerk RTK

Netwerk RTK is gebaseerd op het gebruik van verschillende verspreide vaste referentie stations. Afhankelijk van de toepassing stuurt de rover zijn locatie door naar de referentie stations, hierdoor is de rover niet afhankelijk van 1 basisstation maar kan deze verbinden met verschillende referentie stations. Hierdoor is er een groter bereik dan bij het enkele vaste basisstation wat ongeveer 10 km is.

7.3.3 Hoe correctiedata ontvangen?

Er zijn een paar manieren om deze correctiedata te ontvangen. Je kan zelf een basisstation bouwen of gebruik maken van een service. Niet alle services zijn gratis, bijvoorbeeld Skylark, deze service heeft bereik over 3 continenten maar is betalend. Gelukkig heeft de Vlaamse overheid een service opgezet genaamd FLEPOS (zie puntje FLEPOS). In onderstaande afbeelding worden de andere services weergegeven die in België gebruikt kunnen worden.



Figuur 37 Correctie services België

7.4 Verschil RTK en SBAS

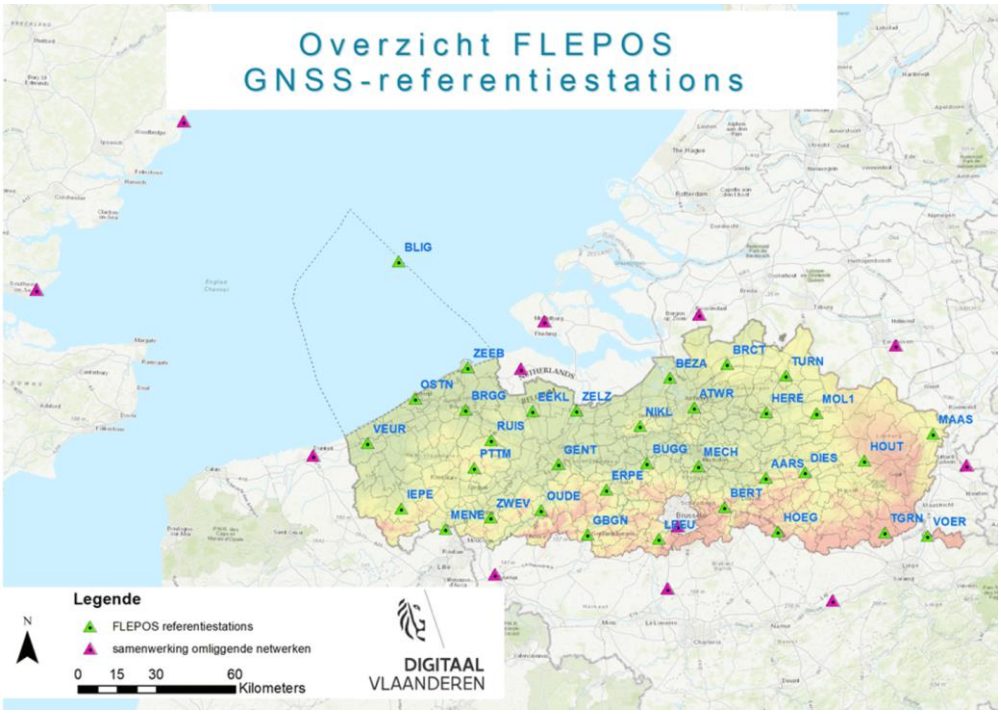
Het grootste verschil is de nauwkeurigheid. SBAS kan goed genoeg zijn voor sommige taken waarbij er niet de hoogste nauwkeurigheid nodig is. Als deze hoogste nauwkeurigheid wel nodig is, kan er best RTK gebruikt worden.

SBAS gebruikt een centraal rekencentrum, die dan doormiddel van geostationaire satellieten de correctiedata stuurt naar de rover. RTK gebruikt geen centraal rekencentrum, deze krijgt de correctiedata gestuurd en berekend deze positie zelf.

Een voordeel van SBAS is dat deze bijna een wereldwijde dekking biedt en gratis is. Dit is bij RTK niet altijd het geval, er kan wereldwijde dekking gegeven worden maar deze is niet gratis. Ook zijn er veel meer referentie stations nodig voor RTK om de nauwkeurigheid te garanderen en is de apparatuur duurder om RTK te kunnen gebruiken.

7.5 FLEPOS

Flemish Positioning Service (FLEPOS) is een dienstverlening van Digitaal Vlaanderen waarbij correctiestromen van navigatiesatellieten via een mobiel netwerk, met het NTRIP-protocol, worden verspreid. Door deze correctiestromen is het mogelijk om overal in Vlaanderen een nauwkeurige positie te bepalen. Gebruik maken van FLEPOS-diensten is volledig gratis, de verbindingskosten over mobiele netwerken, zoals 4G, zijn niet gratis.



Figuur 38 FLEPOS Referentie stations

7.5.1 Gebruik FLEPOS

FLEPOS wordt gebruikt binnen verschillende markten die nood hebben aan een nauwkeurige plaatsbepaling:

- Landmeetkunde, fotogrammetrie
- Precisielandbouw
- GIS-inventarisatie
- Hydrografie
- Werktuigenbesturing
- Baggerwerken
- …

Voor elke van deze markten is er een specifieke RTK-dienstverlening opgezet. Ook biedt FLEPOS-data voor wetenschappelijk onderzoek aan. Voordat FLEPOS gebruikt kan worden moet eerst het registratieformulier ingevuld en goedgekeurd worden op de website van Digitaal Vlaanderen.

7.5.2 Mountpoints

Voor de correctiedata te ontvangen moet er verbonden worden met een mountpoint van FLEPOS. In onderstaande figuur worden alle beschikbare mountpoints weergegeven. De meest gebruikte datastromen zijn FLEPOSVRS31GR en FLEPOSVRS32GREC. Het is de gebruiker vrij één van de andere beschikbare datastromen te nemen voor de GNSS-ontvanger. Datastroom FLEPOSMARIVRS31GR wordt enkel aanbevolen voor meting op de Noordzee omdat deze datastroom gebruik maakt van de 2 overzeese stations ALDB en SHOE en het station BLIG dat zich op de Bligh zandbank situeert.

MOUNTPOINT IN FLEPOS 3.0	GPS	GLONASS	Galileo	Beidou	Type	Oplossing	Formaat	TYPE ABONNEMENT					
								SURVEY	LANDBOUW	MACHINESTURING	MARITIEM	ONDERWIJS	TEST
FLEPOSVRS23G	x	-	-	-	RTK	VRS	RTCM 2.3	x	x	x	x	x	x
FLEPOSNRS23G						NRS	RTCM 2.3	x	x	x	x	x	x
FLEPOSDGNSS23GR	x	x	-	-	DGNSS	VRS	RTCM 2.3	x	x	x	x	x	x
FLEPOSVRS31GR	x	x	-	-	RTK	VRS	RTCM 3.1	x	x	x	x	x	x
FLEPOSVRSCMRGR							CMR	-	x	x	x	x	x
FLEPOSVRSCMRPLUSGR							CMR+	-	x	x	x	x	x
FLEPOSNRS31GR						NRS	RTCM 3.1	x	x	x	x	x	x
FLEPOSNRSCMRGR							CMR	-	x	x	x	x	x
FLEPOSNRSCMRPLUSGR							CMR+	-	x	x	x	x	x
FLEPOSMARIVRS31GR[3]	x	x	-	-	RTK	VRS	RTCM 3.1	-	-	-	x	x	x
FLEPOSVRS32GRE	x	x	x	-	RTK	VRS	RTCM 3.2 MSM	x	x	x	x	x	x
FLEPOSVRSCMRXGRE							CMRx	-	x	x	x	x	x
FLEPOSNRS32GRE						NRS	RTCM 3.2 MSM	x	x	x	x	x	x
FLEPOSNRSCMRXGRE							CMRx	-	x	x	x	x	x
FLEPOSVRS32GREC	x	x	x	x	RTK	VRS	RTCM 3.2 MSM	x	x	x	x	x	x
FLEPOSVRSCMRXGREC							CMRx	-	x	x	x	x	x
FLEPOSNRS32GREC						NRS	RTCM 3.2 MSM	x	x	x	x	x	x
FLEPOSNRSCMRXGREC							CMRx	-	x	x	x	x	x

Figuur 39 Mountpoints FLEPOS

7.6 RTK-data connecteren met RTKLIB

7.6.1 Wat is RTKLIB?

Figuur 40 Logo RTKLIB

RTKLIB is een open source programmapakket voor GNSS-plaatsbepaling. In dit pakket zitten verschillende tools genaamd:

- RTKPLOT, wordt gebruikt om de signalen te plotten.
- RTKCONV, wordt gebruikt om ruwe data om te vormen naar leesbare data.
- STRSVR, wordt gebruikt om data streams te splitten en voor data van in naar output te sturen.
- RTKPOST, wordt gebruikt om berichten te posten, naar output of RTKNAVI.
- NTRIP-browser, dit is een browser voor alle beschikbare mountpoints in te bekijken.
- RTKNAVI, wordt gebruikt om input, output en logstream te configureren
- RTKGET, wordt gebruikt om GNSS-producten en data te downloaden.

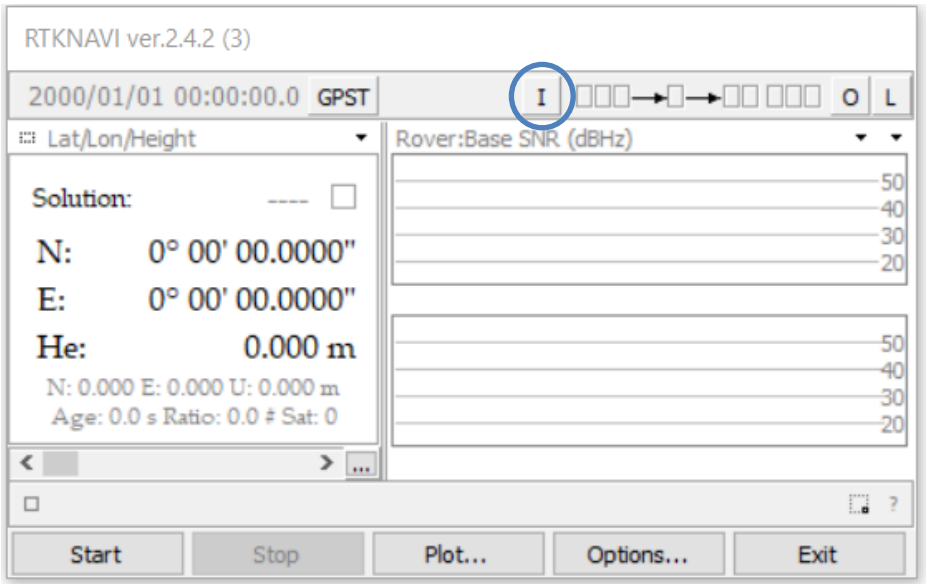
7.6.2 Stappenplan connecteren (Windows)

Je kan RTKLIB op Windows gebruiken om de correctiedata door te sturen naar het RTK-bordje. Deze methode heb ik vooral voor testing gebruikt omdat deze een grafische interface heeft hierdoor kon ik gemakkelijker fouten opsporen.

Stap 1: Registreer voor FLEPOS zoals in het hoofdstuk hierboven. Zodra je geregistreerd bent en je inloggegevens hebt kan je aan de slag.

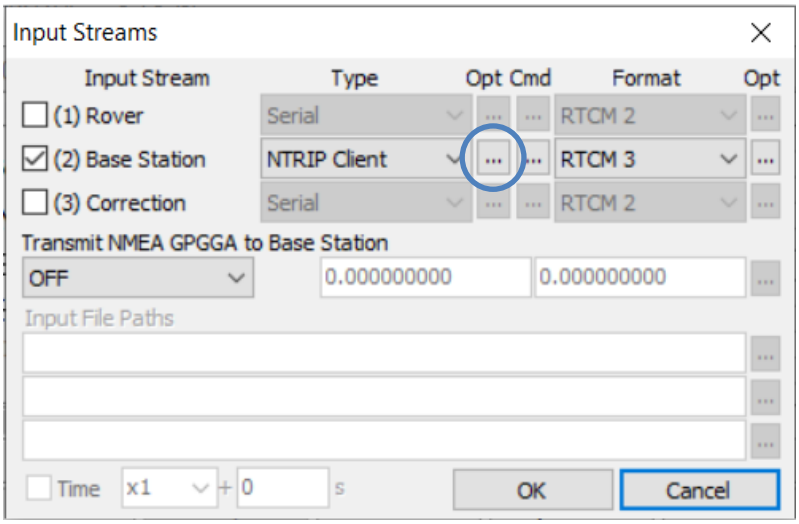
Stap 2: Installeer RTKLIB http://www.rtklib.com/

Stap 3: Wanneer je RTKLIB hebt geïnstalleerd kan je de rtklaunch.exe openen. Deze gaat een menu openen waar je verschillende toepassingen kan uitvoeren. Dan klik je op de RTKNAVI-knop, RTKNAVI zorgt ervoor dat we kunnen verbinden met RTCM-feeds van verschillende providers. Hierin klik je op de “I” knop



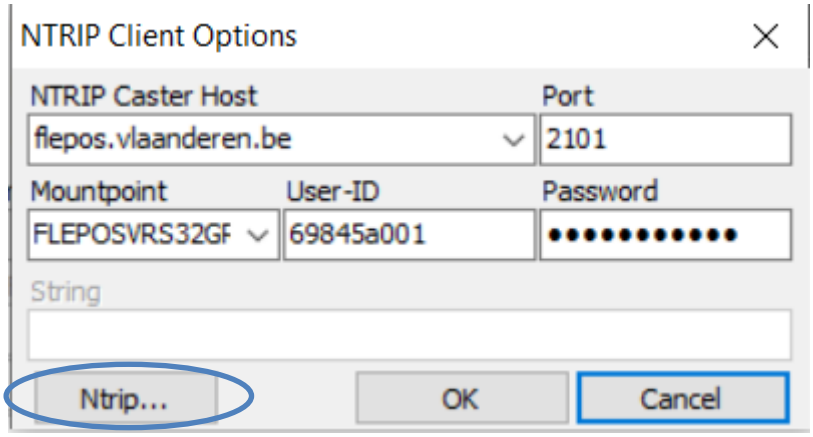
Figuur 41 RTKNAVI

In het Input Streams venster klik je op het selectievakje van de base station, hier duid je NTRIP Client (NTRIP staat voor Networked Transport of RTCM via Internet Protocol) aan van het drop down venster en kies je voor RTCM3 als formaat. Dan druk je op de 3 kleine puntjes onder Opt.



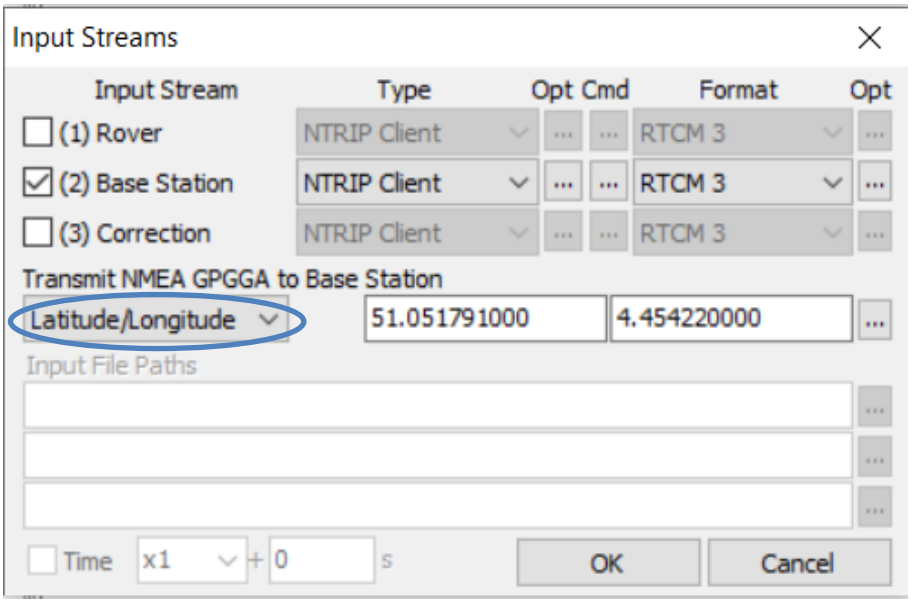
Figuur 42 Input Streams RTKLIB

Dan komt het NTRIP Client Options venster te boven, hier geef je de NTRIP Caster Host in, poort, mountpoint dat nog te bepalen is, User-ID/username, Password. Voor de mountpoint te bepalen druk je op de knop Ntrip…, hier is de lijst te vinden met alle mogelijke Mountpoints. Kies de mounpoint dat het beste bij uw doeleind past. NEO-M8P werkt allen met RTCM 3.x. Ik gebruik FLEPOSVRS32GREC omdat deze alle GNSS-systemen kan gebruiken en een van de twee datastromen is die het meeste wordt gebruikt van FLEPOS. Om deze te gebruiken kopieer je de naam van de Mountpoint die je wil gebruiken in het Mountpoint vakje. Daarna kan je de Ntrip browser sluiten. Wanneer alle inloggegevens zijn ingevuld klik je op OK om de NTRIP Client Options te sluiten.



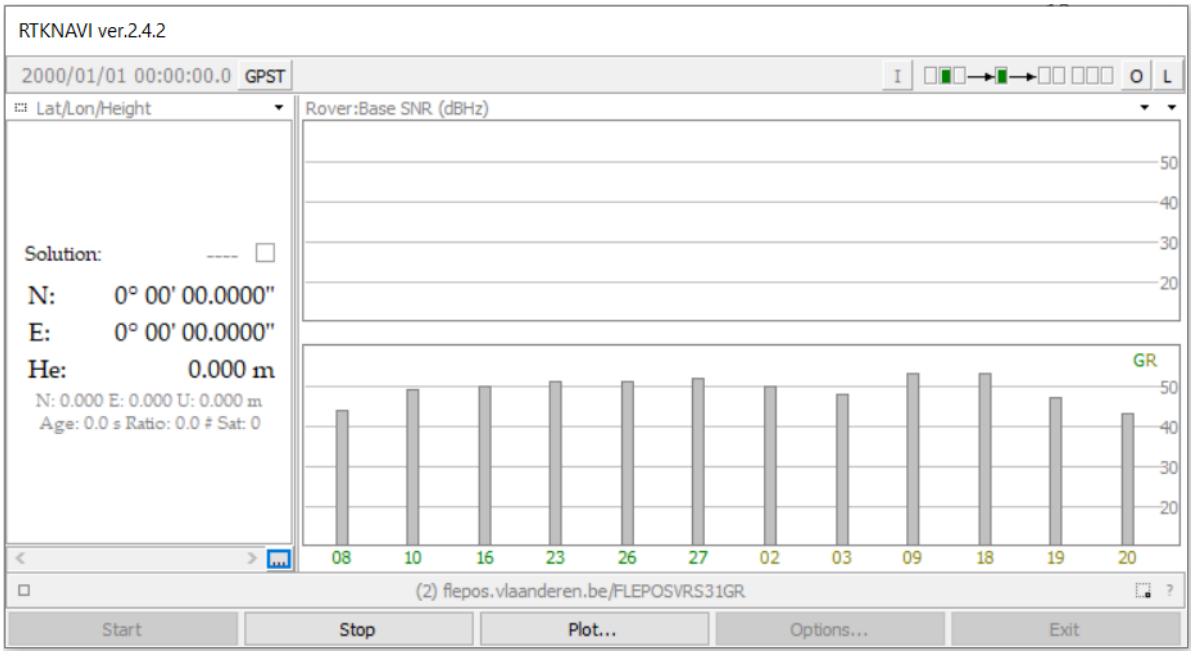
Figuur 43 NTRIP Client Options

Hierna duid je in het drop down menu van "Transmit NMEA GPGGA to base station" Latitude en Longitude aan. Hier geef je de lat/long in een straal van ongeveer 15km op de plek waar de rover zich bevindt. Omdat de software van FLEPOS een NMEA GGA met de initiële positie van de GNSS-ontvanger verwacht om zo een VRS (Virtual Reference Station) of NRS-correctiestroom te kunnen genereren. Andere software dan RTKLIB, stuurt de positie van de rover steeds door naar FLEPOS.

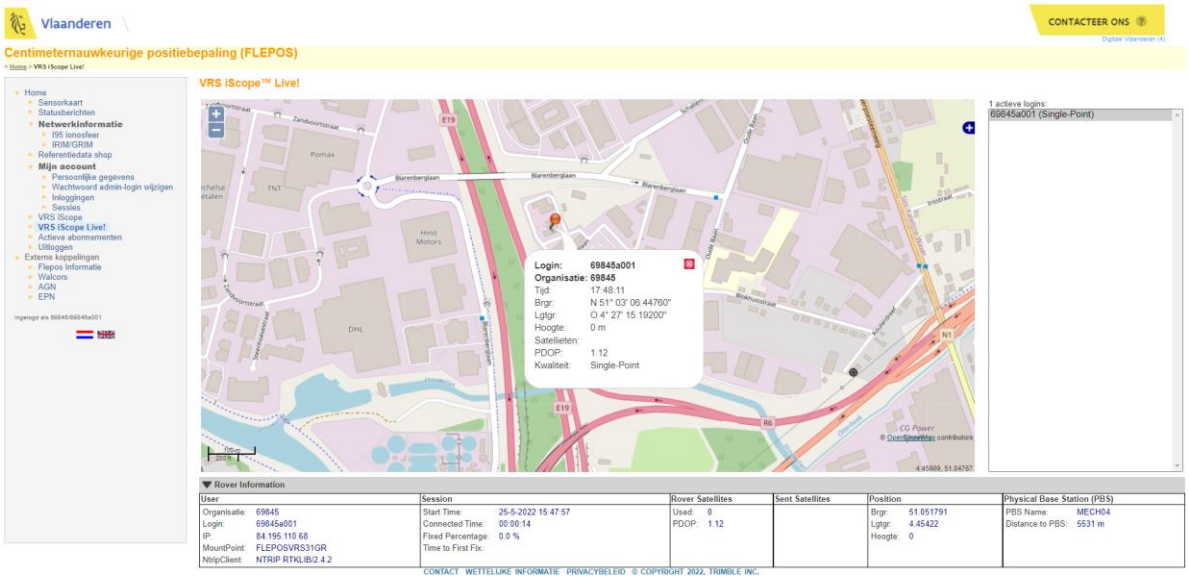


Figuur 44 Input Streams transmit fixed data

Het venster Input Streams zou geconfigureerd moeten zijn, deze kan je ook sluiten door op OK te drukken. Dan klik je op start om de connectie te testen naar de FLEPOS-server.



Figuur 45 RTKNAVI receiving correction data

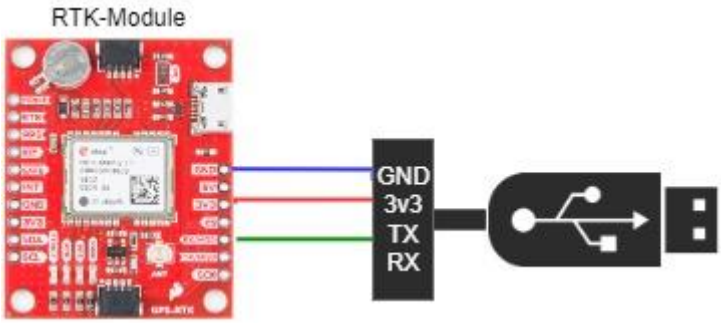


Figuur 46 VRS iScope Live

In bovenstaande afbeelding ziet u in de “VRS IScope Live” van FLEPOS dat er een virtueel referentiestation is aangemaakt.

Als we eenmaal de correctiedata binnenkrijgen zoals hierboven kunnen we deze data loggen naar het RTK-bord met seriële communicatie, doormiddel van een usb naar seriële adapter zoals in onderstaande afbeelding.

Figuur 47 USB naar serieel adapter



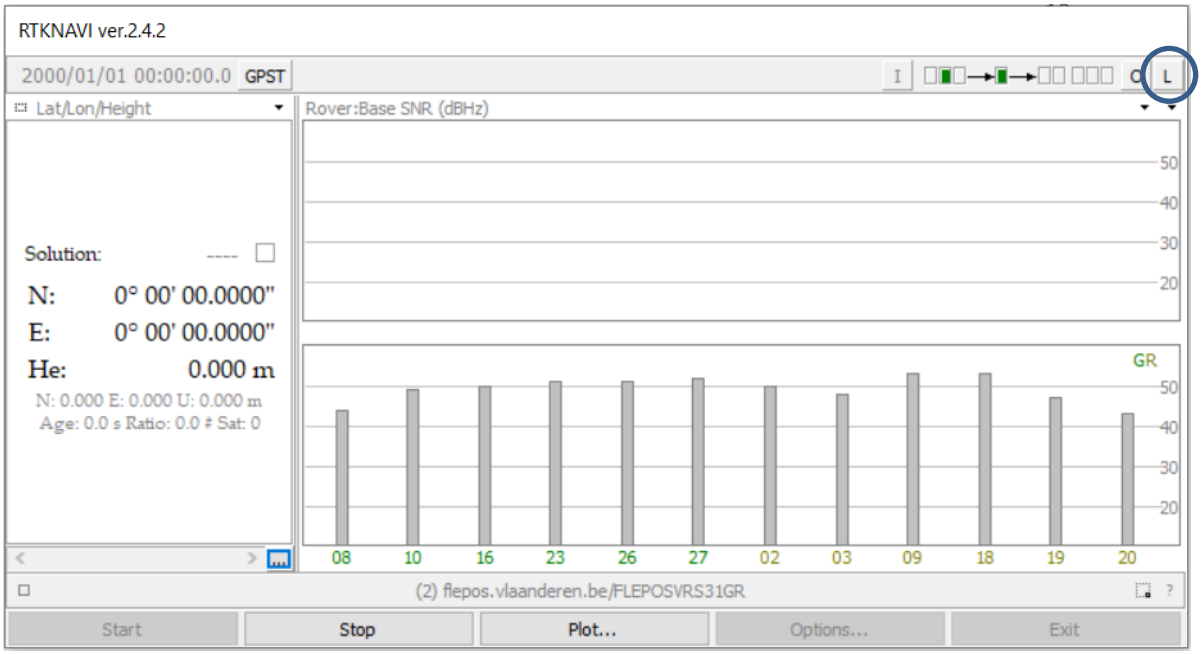
Figuur 48 Diagram RTK-Module Seriële verbinding

Deze kabel zorg ervoor dat we een seriële verbinding kunnen maken tussen de computer en het RTK-bordje. Op bovenstaande afbeelding ziet u hoe we de adapter verbinden met de module. Connecteer het RTK-bord ook met de ingebouwde micro usb aansluiting zodat we later de precieze locatie data kunnen visualiseren met u-center.

RTK-bord	USB-Serieel adapter
GND	GND
3v3	3v3
RX	TX

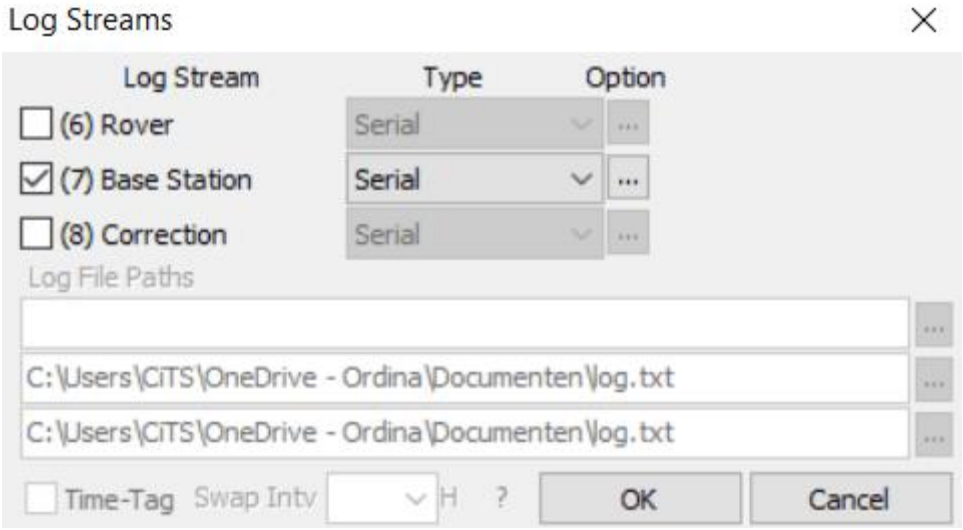
De TX-pin van het RTK-bord sluiten we niet aan. Dit omdat we alleen data moeten ontvangen van de USB-Serieel adapter. Als deze wel wordt aangesloten kan er een error voorkomen.

Nadat we deze verbonden hebben kunnen we de correctiedata naar het RTK-bord loggen doormiddel van de log streams in RTKNAVI. Dit doen we door op de L knop te duwen rechtsboven.



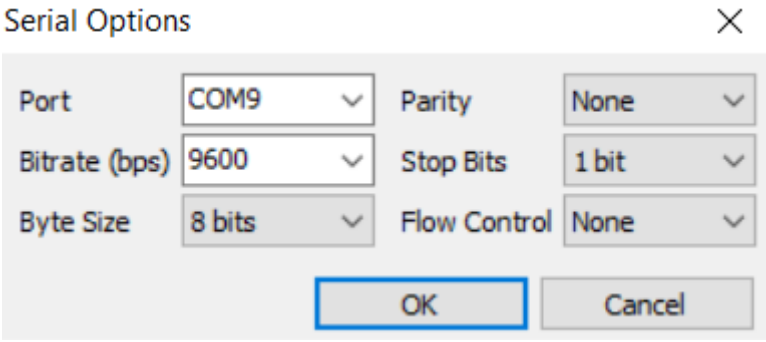
Figuur 49 Correctiedata RTKLIB

In het Log Streams venster duiden we aan dat we de correctiedata willen loggen naar de base station (RTK-bordje), dit wordt aanbevolen door SparkFun. Hier duiden we het type serial aan omdat we via seriële communicatie willen verbinden.



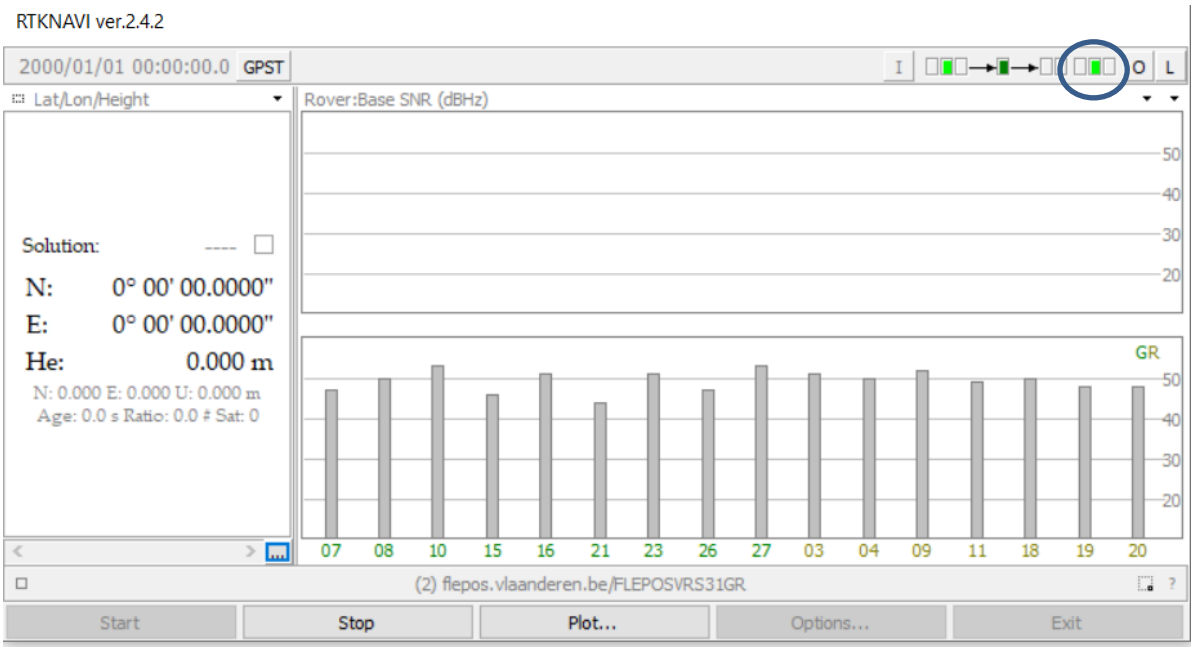
Figuur 50 Log Streams

Dan moeten we de seriële verbinding verder instellen. Eerst kies je de poort waar de USB to serieel adapter met verbonden is, bij mij is dit COM9. Dan stellen we de juiste Bitrate/baudrate in, deze stellen we in op 9600 omdat dit aangeraden wordt door SparkFun. De byte size is 8 bits lang, we hebben geen pariteit bit, 1 stop bit en ten slotte geen Flow Control. Als dit allemaal is ingesteld kunnen we op OK drukken om dit te sluiten.



Figuur 51 Serial Options

Het Log Streams venster mag je ook sluiten door op OK te drukken. Als we dan op start duwen zien we, doormiddel van de knipperende kleuren indicator rechtsboven, dat de correctiedata gelogd wordt naar het RTK-bord.



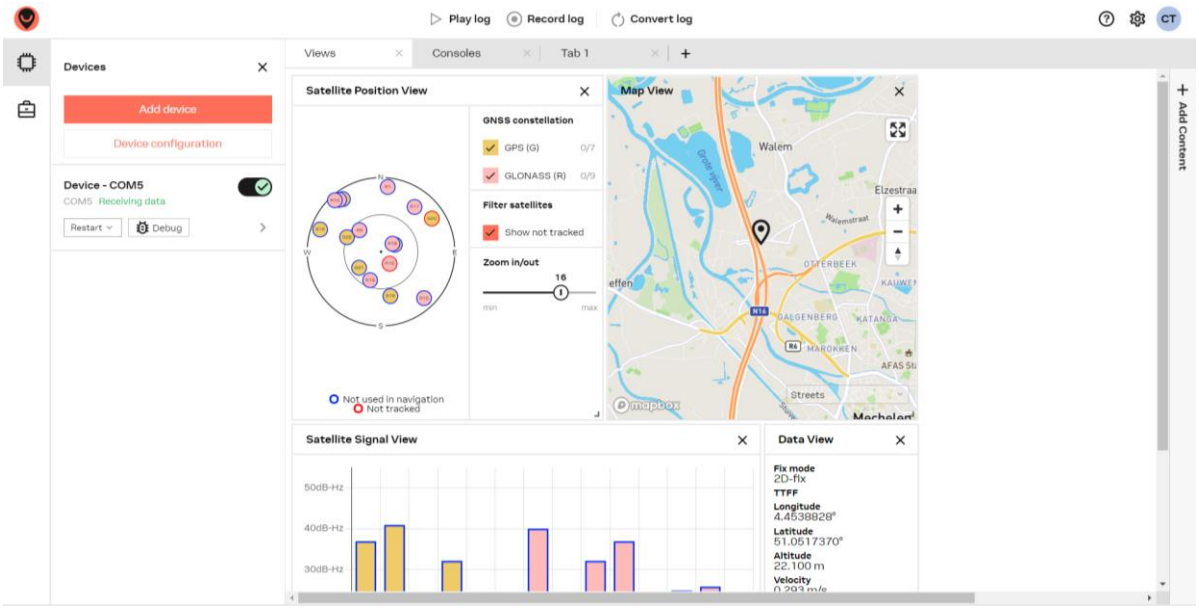
Figuur 52 Correctiedata log naar RTK-Bord

Betekenis kleuren indicator:

- Orange: wachten op connectie
- Donkergroen: geconnecteerd
- Lichtgroen: data actief
- Rood: error

Eenmaal wanneer het RTK-bord een geldige correctiedatastroom binnenkrijgt gaat de RTK-led op het RTK-bord beginnen knipperen. Dit betekent dat het RTK-bord in float modus staat. Wanneer de RTK-led uitgaat heeft het RTK-bord de hoge precisie locatie berekend en zend hij deze precieze locatie uit. Dit kunnen we, tijdens de test periode, visualiseren met u-center.

Dit doen we door te connecteren met u-center via de ingebouwde usb aansluiting. Zodra we verbonden zijn kunnen we een hele hoop parameters zien en een visualisatie op een geografische map.



Figuur 53 U-center visualisatie

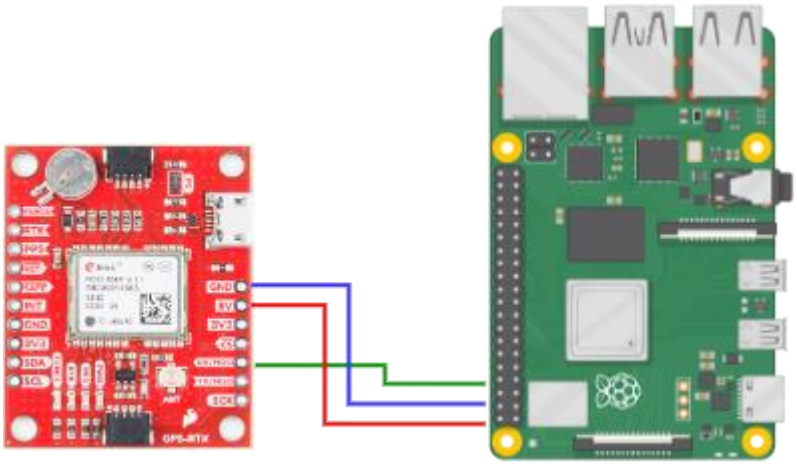
7.6.3 Stappenplan connecteren (Linux)

Het is hetzelfde principe als hiervoor alleen is er in plaats van een GUI een CUI die we gebruiken. Hoe je de CUI-apps en alle bijbehorende commando's bouwt, kun je vinden in de datasheet van (Takasu, 2007-2013).

De benamingen van de CUI-versie van RTKLIB zijn iets anders, maar doen in grote lijnen hetzelfde:

- RTKRCV, wordt gebruikt om RTK-server te starten en stoppen, opties te configureren, uitkomsten en statussen te printen en commando’s uit te voeren.
- RNX2RTKP, wordt gebruikt om RINEX OBS/NAV/GNAV/HNAV/CLK, SP3, SBAS-berichtlogbestanden te lezen en ontvanger (rover) posities en output positie oplossingen te berekenen.
- POS2KML, wordt gebruikt om positiebestand(en) te lezen en te converteren naar Google Earth KML-bestand.
- CONVBIN, wordt gebruikt om RTCM, ruwe data log ontvanger en RINEX-bestanden om te vormen tot een leesbaar, RINEX en SBAS/LEX bericht bestand.
- STR2STR, wordt gebruikt om input data van een input steam eventueel op te delen en uit te sturen naar één of meerdere ouput streams.

Stap 1: RTK-module aansluiten



Figuur 54 Aansluitschema RTK-bord Raspberry Pi

RTK-bord	Raspberry Pi
GND	GND
5V	5V
RX	TX GPIO 14

De TX-pin van het RTK-bord sluiten we niet aan. Dit omdat we alleen data moeten ontvangen van de Raspberry Pi over seriële communicatie. Als deze wel wordt aangesloten kan er een error voorkomen.

Figuur 55 Aansluiting Raspberry Pi RTK-bord

In de uiteindelijke versie van de stage worden alle apps automatisch gedownload en gestart. Volgende stappen worden gebruikt om alles handmatig te testen.

Stap 2: Apps installeren
Het is mogelijk om alle apps van deze tool te installeren, dit doen we via:

```
sudo git clone https://github.com/rtklibexplorer/RTKLIB.git
cd /home/pi/RTKLIB/app/consapp
sudo make
sudo make install
```

Of er kan een specifieke app geïnstalleerd worden zoals STR2STR. Deze ga ik gebruiken om data door te sturen naar het RTK-Bord:

```
sudo git clone https://github.com/rtklibexplorer/RTKLIB.git
cd /home/pi/RTKLIB/app/consapp/str2str/gcc
sudo make
sudo make install
```

Eenmaal dat de app geïnstalleerd is kunnen we het juiste commando invoegen. De lijst van commando's is te vinden op pagina 99 in de datasheet van (Takasu, 2007-2013).

Stap 3: uitvoeren commando
Dit is het commando dat we gaan gebruiken:

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -
out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Verduidelijking commando:
Eerst hebben we sudo, deze gebruiken we om commando's uit te voeren met root rechten.

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -
n msec  -out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Dan kiezen we welke app we willen gebruiken, hier is dat str2str.

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -
n msec -out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Dan declareren we de input stream (-in) hiermee maken we verbinding met FLEPOS om de correctiedata te ontvangen. In deze input stream declaratie geven we:

- Eerst het verbindingstype = NTRIP
- Dan de user dat we willen gebruiken van FLEPOS
- Dan het paswoord
- Dan het verbindingsadres van FLEPOS = 212.204.120.33
- Dan de verbindingspoort van FLEPOS = 2101
- ten slotte de mountpoint = FLEPOSVRS32GREC.

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -n msec -out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Dan geven we onze vaste positie (-p) weer (latitude, longitude en hoogte) en de nmea request cycle, zodat FLEPOS weet waar onze ontvanger zich in een straal van 15 km zich bevindt.

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -n msec  -out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Dan declareren we de ouput stream (-out), via seriële verbinding maken we connectie met het RTK-bord. In deze output stream declaratie geven we:

- Eerst het verbindingstype = serial
- Dan de poort om te verbinden= ttyAMA0
- Dan de baudrate = 9600
- De rest kunnen we leeg laten deze staan standaard juist.

```
sudo str2str -in ntrip://[user[:passwd]@]addr[:port][/mntpnt] -p lat lon hgt -n msec  -out serial://port[:brate[:bsize[:parity[:stopb[:fctr]]]]]
```

Ingevuld ziet dit er zo uit:

```
sudo str2str -in ntrip://69845a001:passwd@212.204.120.33:2101/FLEPOSVRS32GREC -p 51.05191220718147 4.909250985575229 19.0 -n 1 -out serial://ttyAMA0:9600
```

7.7 NMEA formaat

De verkregen locatie data wordt weergegeven in het National Marine Electronics Association (NMEA) formaat. Dit formaat wordt wereldwijd gebruikt voor plaatsbepalingen. Hieronder vindt u meer informatie over dit formaat.

7.7.1 Zinsopbouw

- Adresveld
- Gegevensvelden
- controlesom veld
- Afsluitend veld
- Alle zinnen bevatten alleen ASCII-tekens.
- De maximumlengte van een zin is 82 tekens.
- Alle velden worden gescheiden door scheidingstekens

7.7.2 Adresveld

Het adresveld begint met "$", gevolgd door de ID van de spreker en een zinsidentificatie. De gebruikte spreker ID's zijn:

- GP voor GPS-oplossingen
- GL voor oplossingen met alleen GLONASS
- GA voor alleen GALILEO-oplossingen
- GN voor Multi GNSS-oplossingen

De gebruikte zin-identificaties zijn:

- GGA - Vaste gegevens Globaal Positioneringssysteem
- VTG - Koers over grond en grondsnelheid
- GSA - GNSS DOP en actieve satellieten
- GSV - GNSS-satellieten in zicht
- RMC - Aanbevolen specifieke minimum GNSS-gegevens
- ZDA - tijd en datum
- PASHR - Houdingsgegevens

7.7.3 Gegevensvelden

Gegevensvelden moeten altijd gescheiden worden door ",". Zij kunnen alfanumerieke waarden bevatten, alle gecodeerd in ASCII-tekens. De lengte van een gegevensveld kan constant zijn, variabel of kan een vast en variabel deel bevatten. Dit verschilt voor elke zin.

7.7.4 Controlesom veld

Het controlesom veld begint met "*" gevolgd door de controlesom van de zin. De controlesom wordt gegenereerd door een bitwise exclusieve OR van alle velden inclusief de "," scheidingstekens, tussen maar niet inclusief de "$" en de "*" karakters. De hexadecimale waarde van de controlesom wordt vervolgens omgezet in twee ASCII-tekens.

7.7.5 Afsluitend veld

De afsluitende reeks bevat de twee ASCII-tekens <CR> en <LF> zonder enig scheidingsteken.

7.7.6 Satelliet Nummering

- GPS: 1-32
- GLONASS: 33-96
- GALILEO: 301-336*

*Er is momenteel geen standaardmanier om Galileo-satellieten te nummeren.

7.7.7 Zinsspecificatie

Ik ga alleen ingaan op de NMEA-zin die het meeste informatie bevat genaamd Recommended minimum specific GNSS-data ($GNRMC). Hierin staan de coördinaten, tijd, datum, grondsnelheid en -richting en of de data "Valid" is. In volgende afbeelding ziet u de opbouw van dit formaat.

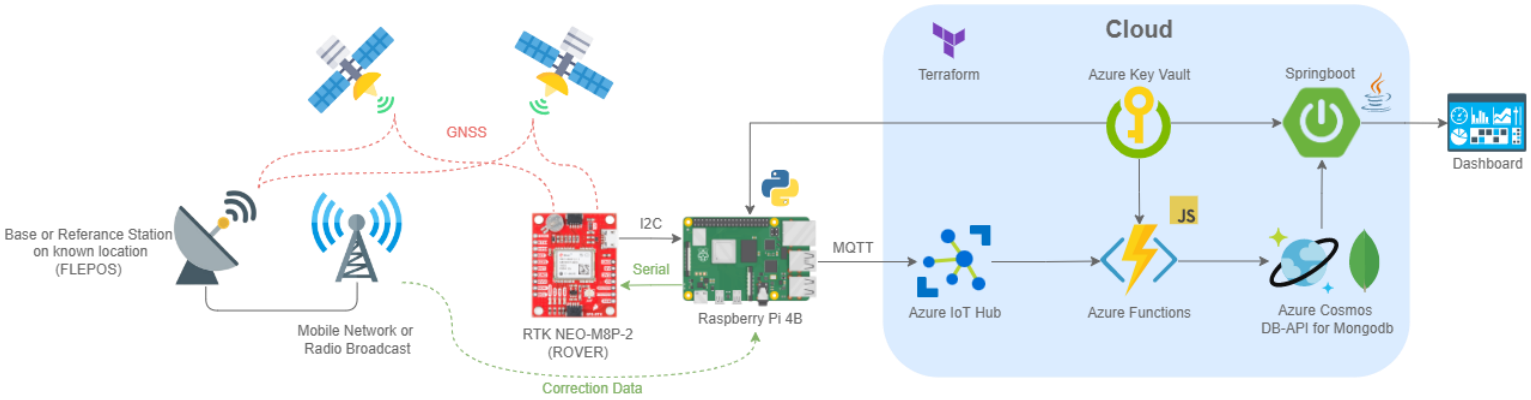
RMC – Recommended Minimum Specific GNSS Data
Time, date, position, course and speed data provided by a GNSS navigation receiver.
Format:
$--RMC,hhmmss.sss,x,llll.lll,a,yyyyy.yyy,a,x.x,u.u,xxxxxx,,,v*hh<CR><LF>

Field	Name	Description
hhmmss.sss	UTC time	UTC time in hhmmss.sss format (000000.000 ~ 235959.999)
x	Status	Status 'V' = Navigation receiver warning 'A' = Data Valid
llll.lll	Latitude	Latitude in dddmm.mmmm format. Leading zeros are inserted.
A	N/S indicator	'N' = North; 'S' = South
yyyyy.yyy	Longitude	Longitude in dddmm.mmmm format. Leading zeros are inserted.
A	E/W Indicator	'E' = East; 'W' = West
x.x	Speed over ground	Speed over ground in knots (000.0 ~ 999.9)
u.u	Course over ground	Course over ground in degrees (000.0 ~ 359.9)
xxxxxx	UTC Date	UTC date of position fix, ddmmyy format
v	Mode indicator	Mode indicator 'N' = Data not valid 'A' = Autonomous mode 'D' = Differential mode 'E' = Estimated (dead reckoning) mode
hh	checksum	

8 REALISATIE

Nu er onderzoek is gedaan naar de nodige technologieën en de nodige testen zijn gebeurd kunnen we de hele infrastructuur gaan ontwikkelen.

8.1 Infrastructuur



Figuur 56 Diagram infrastructuur

Het RTK-bord gaat verbinden met de GNSS-satellieten, deze stuurt zijn ongecorrigeerde locatie door. Deze ongecorrigeerde locatie van de rover (RTK-bord) wordt ontvangen door een base- of referentiestation. In de uiteindelijke opstelling van de stage wordt deze handmatig ingesteld en doorgestuurd naar het referentiestation, dit omdat FLEPOS een vaste locatie vraagt. De positie van dit base- of referentiestation staat vast bepaald en is bekend. Dan wordt er een correctie stroom gegenereerd die doorgestuurd wordt via een mobiel netwerk naar de Raspberry Pi.

De Raspberry Pi is verbonden, via I²C en seriële communicatie, met het RTK-bord. De seriële communicatie wordt gebruikt om de correctie stroom, dat de Raspberry Pi van FLEPOS ontvangt, door te sturen naar het RTK-bord. Door deze correctie stroom te voeden naar het RTK-bord kan deze zijn precieze locatie berekenen. De I²C connectie wordt gebruikt om deze precieze locatie door te sturen naar de Raspberry Pi.

Deze data worden via een Python programma op de Raspberry Pi (zie hoofdstuk: Uitlezen RTK-bord naar Cloud toelichting code) ontvangen, bewerkt en doorgestuurd via MQTT naar Azure IoT Hub (Cloud). Deze data worden dan via een Javascript functie (zie hoofdstuk: Azure functions toelichting) opgeslagen in de Mongodb database. De backend is gemaakt in Java met behulp van Spring Boot. Deze backend haalt alle data uit de database en visualiseert deze, voormalig nog in plain tekst omdat dit niet de grootste focus was tijdens de stage. Deze infrastructuur zou idealiter beveiligd moeten worden

via de Azure Key Vault. Tijdens het testen was dit niet de grootste prioriteit dus is er niet verder op ingegaan.

De infrastructuur van het Cloud gedeelte kan worden geïmporteerd via de Terraform file. Via een zelf geschreven installeer script kunnen alle acties op de Raspberry Pi geïnstalleerd en uitgevoerd worden.

8.1.1 Demo video

De volledige werking kan bekeken worden in volgende video:
https://youtu.be/GJu5fh_Wdxk

8.2 Installeer script toelichting code

Het zelf geschreven installeer script wordt gebruikt om alle acties op de Raspberry Pi te installeren en uit te voeren.

Eerst heb ik ASCII-art toegevoegd om de installatie visueel aantrekkelijker te maken.

```
#!/bin/sh

#ASCII-ART for RTK module
echo "\e[32m"
echo "  ____ _____ _  __      __  __  ___  ____  _   _ _     _____ "
echo " |  _ \_   _| |/ /     |  \/  |/ _ \|  _ \| | | | |   | ____|"
echo " | |_) || | | ' /_____| |\/| | | | | | | | | | | |   |  _| "
echo " |  _ < | | | . \_____| |  | | |_| | |_| | |_| | |___| |___"
echo " |_| \_\|_| |_|\_\    |_|  |_|\___/|____/ \___/|_____|_____|"
echo "\e[0m"
```

Daarna kijkt het script na of er een internet connectie is.

```
#checking network connection
if timeout 2 nc -z google.com 443; then
    echo "\e[32m"
    echo "Network connection OK"
    echo "\e[0m"
else
    echo "\e[31m "
    echo "NO NETWORK CONNECTION"
    echo "\e[0m"
    exit 1
fi
```

Hierna worden alle pakketten geïnstalleerd die nodig zijn om het proces te laten lopen.

```
#install packages

PACKAGES="python-pip"
#update took to long
#sudo apt-get update
#sudo apt-get upgrade -y
sudo apt-get install $PACKAGES -y
sudo pip3 install smbus
sudo pip3 install azure-iot-device

#Install RTKLIB
sudo git clone https://github.com/rtklibexplorer/RTKLIB.git
cd /home/pi/RTKLIB/app/consapp
sudo make
sudo make install
```

Vervolgens wordt er nagekeken of I²C aanstaat op de Raspberry Pi, zo niet gaat het script deze aanzetten en de Raspberry Pi heropstarten. De heropstart is nodig voor het aanzetten van I²C te voltooien.

```
#ENABLE I²C
CONFIG="/boot/config.txt"

#if dtparam=I²C_arm=on is commented out
if grep -Fq "#dtparam=I²C_arm=on" $CONFIG
then
    sudo sed -i "s/#dtparam=I²C_arm=on/dtparam=I²C_arm=on/" $CONFIG
    sudo reboot
else
    echo "I²C is on"
fi
```

Dan wordt er een map gecreëerd waarnaar de Git repository wordt gekopieerd vanuit Azure DevOps.

```
#make dir clone python file from git and run
sudo rm -rf /home/pi/automaticD
sudo mkdir -p /home/pi/automaticD
cd /home/pi/automaticD
sudo git clone
https://5mao46tazegrut7dmtt42qwgjoh3ro7gtsz3tb4pz2ybaujrgica@dev.azure.com/or
dina-jworks/SparkFun-gps-rtk-stage/_git/SparkFun-rtk-read
cd /home/pi/automaticD/SparkFun-rtk-read
```

Zodra deze gekopieerd is worden er 3 parameters gevraagd die van belang zijn om de initiële locatie in te stellen van het RTK-bord zodat FLEPOS weet waar deze zich bij startup bevindt.

```
echo Enter latitude of rover:
read lat
echo Enter longitude of rover:
read lon
echo Enter height of rover:
read hgt
```

Als deze 3 parameters zijn ingevuld wordt de server gestart om correctiedata te ontvangen. Ook wordt er gecheckt of deze server wel degelijk loopt.

```
until sudo str2str -in
ntrip://69845a001:Ordina2022\!@212.204.120.33:2101/FLEPOSVRS32GREC -p $lat
$lon $hgt -n 1 -out  serial://ttyAMA0:9600:8 &
do
    echo "\e[31m"
    echo "CORRECTION DATA NOT RUNNING"
    echo "\e[0m"
    sleep 1s
done
```

Zodra deze server lopende is wordt het Python programma gestart zodat de rest van de infrastructuur kan doorlopen worden.

```
until sudo python3 Gnss_I²C_to_Cloud_batches.py
do
    echo "\e[31m"
    echo "GNSS PROGRAM NOT RUNNING"
    echo "\e[0m"
    sleep 1s
done
```

8.3 Uitlezen RTK-bord naar Cloud toelichting code

Het uitlezen van het RTK-bord gebeurt via een Python programma. Deze data worden dan doorgestuurd naar de Cloud via MQTT. Als volgt wordt het programma toegelicht hoe dit in zijn werk gaat.

Eerst worden de libraries geïmporteerd: re is een library die ik gebruik om een I²C bug te maken. De time library wordt gebruikt voor een delay toe te voegen. Smbus is een library voor het uitlezen van de I²C poort. Traceback wordt gebruikt om eventuele fouten weer te geven. Sys is het systeem dat wordt geïmporteerd. Httplib wordt gebruikt om de internet connectie te testen. Azure IoT device wordt gebruikt om te verbinden met de IoT Hub van Azure en het formateren van berichten.

```
import re
import time
import smbus
```

```
import traceback
import sys
import http.client as httplib
from azure.iot.device import IoTHubDeviceClient, Message
```

De connectie string wordt gebruikt om te connecteren met de IoT Hub. Deze zou het best verborgen zijn maar tijdens het testen is dit oké als deze niet verborgen staat.

```
CONNECTION_STRING = "[Fill in connection string] "
```

De volgende functie gaat de 7de bit van de I²C bericht leegmaken omdat hier een fout in kan zitten waardoor het bericht onbruikbaar wordt.

```
def fix(m):
    b = chr(ord(m.group(0)) & 0x7F) # clear bit 7.
    #print(f'Replaced {a} with {b}')
    return b
```

Dan wordt het JSON bericht gedefinieerd zodat we deze later kunnen gebruiken om door te sturen naar de IoT Hub.

```
# Define the JSON message to send to IoT Hub.
MSG_TXT = '{{"Data": {msgString}}}'
```

In deze functie wordt er een IoT Hub client aangemaakt.

```
def iothub_client_init():
    # Create an IoT Hub client
    client =
IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    return client
```

Hierna wordt in de volgende functie de internet connectie nagekeken. Als deze geen internet connectie heeft gaat hij nooit de berichten kunnen doorsturen naar de Cloud.

```
def have_internet():
    conn = httplib.HTTPSConnection("8.8.8.8", timeout=5)
    try:
        conn.request("HEAD", "/")
        print("Connected to the Internet")
        return True
    except Exception:
        print("No internet connection.")
        return False
    finally:
        conn.close()
```

Als volgt wordt er connectie gemaakt met de I²C bus, dan worden alle variabelen gedefinieerd. Het I²C adres waarop het RTK-bord is aangesloten wordt ingesteld. Hierna wordt er een lege array aangemaakt voor de data maar ook een array om later de data te gaan ordenen.

```
def iothub_client_telemetry_sample_run():
    while True:
        try:
            I²Cbus = smbus.SMBus(1)
            emptyspaceCounter = 0
            NoSignalCounter = 0
            address = 0x42
            data = []
            dataBegin = [36,71,78,82,77,67] # = $GNRMC
            msgString=""
```

Dan wordt de eerste byte van de I²C bus uitgelezen en nagekeken of dat deze een dollarteken is. Door dit dollarteken kan de datalijn per lijn gesplist worden. Daarna wordt dit teken in de array geplaats.

```
while True:
                msg= I²Cbus.read_byte(address) #reads the first byte
                if msg == 36: #checks if it is a $ sign
                    data.append(msg)
```

Dan wordt de bus opnieuw uitgelezen en toegevoegd aan de array tot dat de eerste 6 tekens gelijk zijn aan de dataBegin array. Hierdoor worden alle lijnen samengevoegd en wordt de data later in de code doorgestuurd in groepen om dataverkeer uit te sparen naar de Cloud toe, dit kan extra kosten als dit dataverkeer te groot wordt. Als de data niet overeenkomt met de eerste 6 tekens van de dataBegin array wordt de data array leeg gemaakt. Dit omdat er dan een leeg teken in zit of het eerste teken niet correct was.

```
while True:
     msg= I²Cbus.read_byte(address) #reads the first byte
     data.append(msg)
     if len(data) >= len(dataBegin) and data[:6]==dataBegin[:6]:

     elif msg == 255:
           data = []
     elif len(data) >= len(dataBegin) and data[:4]!=dataBegin[:4]:
           break
```

Hier wordt de rest van de data ingelezen, dan wordt deze opgedeeld in een groep en wordt deze opgeschoond door alle lege karakters te verwijderen. Dan wordt het laatste 6 tekens van het bericht verwijderd omdat deze gebruikt werd voor het bericht te sorteren. Dan wordt de data nagekeken of deze wel gebruikt kan worden. Als deze bruikbaar is wordt deze data verstuurd naar de Azure IoT Hub. Als de data niet bruikbaar is wordt er om de 10 seconden een bericht naar de IoT Hub gestuurd zodat er remote feedback is. Nadat de data zijn verstuurd wordt alles leeggemaakt.

```
while True:
     msg= I²Cbus.read_byte(address) #reads the first byte
     data.append(msg)
```

```
    if data[-6:]==dataBegin[:6]:
      cleandata=[]
    for i in data: #remove all the 255 in data list clean up for debugging
      if i != 255:
        cleandata.append(i)
    data=cleandata
    del data[-6:] #deletelast GNRMC

    checkdata=[]
    for i in data[1:]:  #checks the first line GNRMC for emptyspaces
      if i == 36:
        break
      checkdata.append(i)

    if 86 in checkdata: #check if the data is invalid
        print("Data invalid")
        NoSignalCounter += 1

    else: #Send gps signal to iot hub
        msgString=''.join(map(chr,data)) #converts it into text
        msgString= re.sub(r'[\x80-\xff]',fix,msgString) # fix I²C bug
        message = Message(msgString,message_id="Data")
        print( "Sending message: {}".format(message) )
        client.send_message(message) #Send to IoTHub
        print ( "Message successfully sent" )
        NoSignalCounter = 0

    if NoSignalCounter == 10: #Every 10 seconds it sends a message to iot
hub for some info
      print( "Sending message: {}".format("Data invalid") )
      client.send_message("GPS DATA NOT VALID, NO SIGNAL") #Send to IoTHub
      print ( "Message successfully sent" )
       NoSignalCounter = 0

    time.sleep(1)
    data = []
    data.extend(dataBegin)
    emptyspaceCounter =0
```

Tot slot worden de fouten doorgestuurd naar de IoT Hub dit kan handig zijn voor het oplossen van fouten op afstand. Daarna is er een stuk code die de functies oproept als er internet connectie is gaat deze pas het volledige programma starten.

```
#if error occures it keeps trying until Keyboardinterrupt
       except Exception as err:
           ErrorMessage=traceback.format_exc()
           print(err,ErrorMessage)
           message=Message(ErrorMessage,message_id="Error")
           client.send_message(message) #Send Error message to IoTHub
```

```
            time.sleep(1)
            continue

        except KeyboardInterrupt:
            print ( "IoTHubClient sample stopped" )
            break

if __name__ == '__main__':
    print ( "Press Ctrl-C to exit" )
    if have_internet(): #check internet connection before running the code
        client = iothub_client_init() #make connection with the iot hub
        iothub_client_telemetry_sample_run() #if there is connection run the
code
    else:
        exit
```

8.4 Azure functions toelichting code

De Javascript functie neemt de data van de IoT Hub en slaagt deze op in de database.

Eerst worden alle benodigdheden toegevoegd. Dan wordt de connectie string van de database, die buiten development doeleinden secure moet staan, toegevoegd. Tot slot wordt er een nieuwe Mongoclient aangemaakt om verbinding te maken met de database.

```
const { MongoClient } = require("mongodb");
const { v4:uuidv4 } = require("uuid")

const uri = "CONNECTION STRING"
const client = new MongoClient(uri)
```

Dan wordt per elk bericht een ID gelogd, hierna wordt er connectie gemaakt met de data base om de verkregen data in op te slagen. Eerst wordt de data gesorteerd zodat deze op een bruikbare manier kan opgeslagen worden in de database.

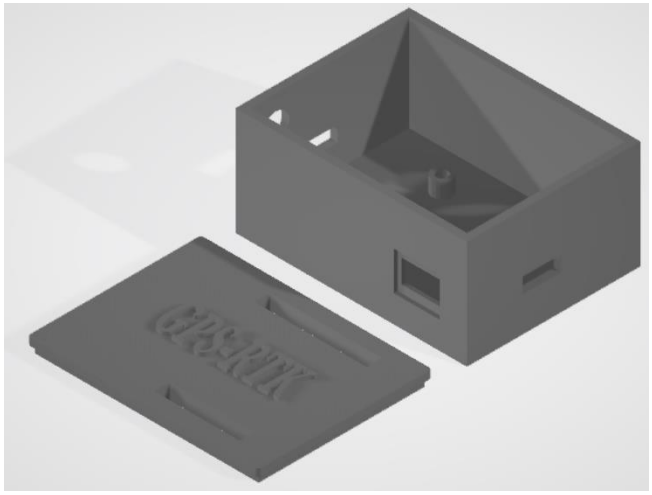
```
module.exports = async function (context, eventHubMessages) {
    context.log(`JavaScript eventhub trigger function called for message
array ${eventHubMessages}`);

    eventHubMessages.forEach((message, index) => {
        context.log(message._id);
    });

    try {
        await client.connect();
        await client.db("rtk").command({ ping: 1 });
        context.log("Connected successfully to server");
        /*const database = client.db("rtk")*/
        const gpsData = client.db("rtk").collection("gpsData")
        //await gpsData.deleteMany({}) //EMPTY THE DATABASE FOR TESTING

        //ORGANIZING THE DATA
        let foo =
eventHubMessages.toString().split("\n").reduce(function(obj, str, index) {
            //data = data.replace(/\r+/g,"") // delete \r because we don't
need them
            let strParts = str.slice(1,6)
            if (strParts && str) { //<-- Make sure the key & value are not
undefined
            obj[strParts.replace(/\s+/g, '').trim()] = str.trim(); //<-- Get
rid of extra spaces at beginning of value strings
            }
            return obj;

        }, {});

        const currentDate = new Date();
        const timestamp = currentDate.toISOString()

        let gpsdata = [
            {
                _id : uuidv4(),
                data : foo,
                timestamp: timestamp
            }
        ]

        await gpsData.insertMany(gpsdata)

        } finally {
            await client.close();
        }
    context.done();
};
```

8.5 Spring Boot toelichting code

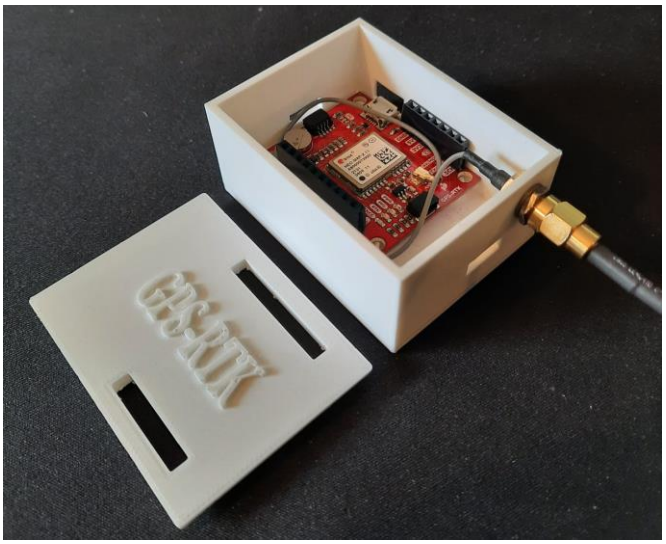
De Spring Boot REST API gaat de data uit de database nemen door middel van een connectie string. Deze data worden in een array geplaatst en krijgt een ID en een timestamp. Daarna wordt deze in data in tekstvorm weergegeven op de interface. De gehele code is te groot om in de documentatie te verwerken, deze kan teruggevonden worden op Azure DevOps. In de demo video (zie hoofdstuk: Demo video) kan je de REST API in werking zien.

8.6 3D geprinte behuizing

Ik heb een behuizing getekend en geprint zodat het RTK-bord niet beschadigd raakt wanneer ik deze buiten ging testen. Deze behuizing is getekend in Tinkercad en geprint in PLA. In de case is genoeg plek voorzien om alle connecties ordelijk te maken.



Figuur 57 3D behuizing design



Figuur 58 3D behuizing

9 PROBLEMEN

Hier haal ik de grootste problemen aan die tijdens mijn stage zijn voorgekomen. Er zijn een aantal kleinere problemen die relatief snel waren opgelost en niet de moeite zijn om aan te halen.

9.1 Correctiedata RTK-bord

Het grootste probleem van mijn stage waar ik heel veel en moeite tijd heb aan besteed is dat het RTK-bord de correctiedata niet wil verwerken. Normaal zou deze standaard als er correctiedata via de seriële poort wordt gevoed deze moeten verwerken, dat deed hij niet.

Ik dacht eerst dat dit aan de correctiedata zelf lag van FLEPOS. Dit omdat er een probleem was dat FLEPOS standaard een vaste start locatie moet hebben van de rover. Dit stond nergens beschreven in de documentatie. Deze documentatie is ook niet meer up-to-date volgens de klantendienst van FLEPOS. Hier ben ik zelf achter gekomen dat FLEPOS deze start locatie nodig heeft om te werken. Na dit werd de correctiedata nog steeds niet verwerkt. Toen heb ik voor de zekerheid naar de hulpdienst van FLEPOS gebeld om uit te sluiten dat het probleem bij FLEPOS ligt. Met deze hulpdienst heb ik zeker 2 uur aan het bellen geweest. Ze hebben kunnen bevestigen dat de fout niet bij hen lag.

Dan heb ik zelf blijven doorzoeken en in de tussentijd naar de fabrikant SparkFun een mail gestuurd met mijn probleem. Zij hebben geantwoord dat ze op de vraag een antwoord hebben maar alleen via hun forum. Toen heb ik mij aangemeld op hun forum en een topic aangemaakt maar tot op de dag van vandaag is deze topic nog steeds niet goedgekeurd. Daarna ben ik blijven zoeken maar uiteindelijk tot op heden heb ik het probleem nog steeds niet gevonden.

9.2 Antenne

Een van de eerste problemen was de onnauwkeurige antenne waardoor ik niet nauwkeurig en moeilijk kon testen. Dit probleem is opgelost door een nieuwe antenne te bestellen. Dit duurde ongeveer 1 week waardoor ik in deze week moeilijk kon testen en verder onderzoek heb verricht naar de verschillende systemen.

9.3 I²C uitlezen

Tijdens het uitlezen van de I²C bus was er een probleem dat er willekeurige karakters in de data werden toegevoegd, hierdoor werd de data onbruikbaar. Dit was een hardware error en ik probeerde deze ook zo op te lossen. Dit had ik gemeld tijdens mijn dagelijkse stand up meeting waardoor Bart Coppens me heeft geholpen met een software fix.

9.4 Azure function

Het eerste plan was om de functie te schrijven in TypeScript dit omdat deze verwacht wordt dat deze in de toekomst meer gebruikt gaat worden dan Javascript. Bij het aanmaken wilde ik deze aanmaken met een Linux besturingssysteem. Dit ging niet omdat op die moment Azure dit niet ondersteunde waardoor ik de functie heb moeten aanmaken op een Windows besturingssysteem en hierdoor Javascript heb gebruikt.

Ook had ik tijdens het schrijven van deze functie een connectie fout gemaakt waardoor ik toch wat kostbare tijd verloren heb.

9.5 Visualstudio code Ordina laptop

Ik gebruikte altijd Visualstudio code om de programma’s in de Raspberry Pi te programmeren omdat je remote verbinding kan maken tussen Visualstudio code en de Raspberry Pi. Dit is zeer handig omdat alles automatisch wordt opgeslagen en overgezet wordt naar de Raspberry Pi waardoor ik dit veel sneller kon testen. Dit verliep altijd heel soepel tot week 7 van de stage. Door een onbekende reden kon ik niet meer verbinden met de Raspberry Pi. Deze fout heb ik proberen op te lossen maar is niet gelukt. Hierdoor moest ik dit op een omslachtigere manier uitvoeren waardoor dit langer duurde.

9.6 Connectie Raspberry Pi

In dezelfde week als het verbindingsprobleem met Visualstudio code was er ook een verbindingsprobleem met de Raspberry Pi. Ik kon op geen enkele manier meer verbinden, uiteindelijk na veel testen kwam ik tot de conclusie dat de SD kaart corrupt was geworden. Daarna heb ik een nieuwe SD kaart in de Raspberry Pi gestoken en was het probleem opgelost. Hierna is het nog een aantal keer voorgevallen dat ik niet kon verbinden door een andere onbekende reden.

Tijdens het demonstreren van mijn project op een terugkomdag op Thomas More, was dit voorval er weer. Dit gebeurde enkel toen ik de beamer aansloot.

9.7 Docker blog post

Om een visualisatie te krijgen van de blog post tijdens dat deze gemaakt wordt is er een Docker container voorzien vanuit Ordina. Dit maakt het aanpassen van de blog post gemakkelijk zodat je direct feedback krijgt over wat je typt. Tijdens het lopen van deze container liep het altijd mis voor een onbekende reden. Deze error heb ik proberen op te lossen maar lukte niet, hierdoor ging het maken van de blog post wat moeizamer en is er wat tijd verloren.

10 BESLUIT

Het doel van de stage was GNSS en de daarbij passende correctiemethodes te onderzoeken. Bijkomend was het de bedoeling deze data naar de Cloud te sturen en automatisch te deployen. Ik heb met dit document als bewijsstuk bewezen dat er onderzoek is verricht naar deze systemen, de locatie data kan uitgelezen en verstuurd worden naar de Cloud.

In het eindproduct is er een onopgelost probleem dat het RTK-bord de correctiedata niet correct verwerkt. Waardoor de correctiemethode waar een grote focus op lag, RTK, niet volledig werkend is. Maar door dit onderzoek heb ik wel een goede basis gelegd om hierop verder te werken. Dit onderzoek is een goede start om projecten op verder te bouwen met oneindige doeleinden.

Verder zijn er nog een aantal kleine werkpunten die verbeterd kunnen worden zoals de automatische deployment en beveiliging van de infrastructuur maar hierop lag niet de grootste focus tijdens mijn opdracht van de stage.

Ik ben zeer tevreden met het eindproduct en dat ik dit onderzoek zelfstandig heb kunnen verwezenlijken.

11 FIGUURLIJST

12 BIBLIOGRAFIE

ANAVS. (sd). *NMEA Solution Output Format*. Opgehaald van ANAVS: https://anavs.com/knowledgebase/nmea-format/

EUSPA (© EUSPA 2021). (2021, Mei 17). *EUSPA .* Opgehaald van EUSPA : https://www.euspa.europa.eu/european-space/eu-space-programme/what-sbas

Microsoft. (sd). *Azure documentation*. Opgehaald van Microsoft: https://docs.microsoft.com/en-us/azure/?product=popular

Nate. (sd). *GPS-RTK Hookup Guide*. Opgehaald van Sparkfun: https://learn.sparkfun.com/tutorials/gps-rtk-hookup-guide?_ga=2.165808823.806341776.1654247501-499731255.1646122031

Takasu, T. (2007-2013). *RTKLIB ver. 2.4.2 Manual.* Opgehaald van http://www.rtklib.com/prog/manual_2.4.2.pdf

Vlaanderen, D. (sd). *DIGITAAL VLAANDEREN*. Opgehaald van Vlaanderen: https://www.vlaanderen.be/digitaal-vlaanderen/onze-oplossingen/flepos-centimeternauwkeurige-positiebepaling