Signaalverwerking van een radar-systeem voor spoorbewaking

Bachelor Afstudeerproject

A.M. Flipse
J.P. Coenen

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

TU Delft

Bachelor Afstudeerproject

Signaalverwerking van een RADAR-systeem voor spoorbewaking

Auteurs:
A.M. Flipse
J.P. Coenen

Docent-begeleider:
prof. dr. O. Yarovoy

Opdrachtgever:
ir. R.I.A. Harmanny

18 juni 2014

TU Delft Delft University of Technology

Goedkeuring Stage-/Afstudeerverslag van:

A.M. Flipse en J.P. Coenen

Titel verslag: Signaalverwerking van een RADAR-systeem voor spoorbewaking

Opleidingsinstelling: Technische Universiteit Delft

Stage-/Afstudeerperiode: april-juni 2014

Vestiging/Afdeling: Delft

Stagebegeleider Thales: R. I. A. Harmanny

Dit verslag (zowel de papieren als de elektronische versie) is door de begeleider van Thales Nederland B.V. gelezen en becommentarieerd. Hierbij heeft de begeleider, naast inhoudelijke zaken, tevens gelet op gegevens die niet naar buiten mogen, zoals plattegronden, confidentiële informatie en organisatieschema's waarin namen staan vermeld.

In principe worden drie categorieën van verslagen onderscheiden, dit verslag valt onder de volgende categorie*:

- ☐ 1. Verslagen en/of samenvattingen hiervan, die vanwege veiligheidsredenen/commerciële aspecten intern (Thales) moeten blijven.
  Het verslag blijft te allen tijde in het archief van Thales. Indien nodig kan dit verslag door een afgevaardigde van de opleidingsinstelling bij Thales worden ingezien, mits deze afgevaardigde een geheimhoudingsverklaring heeft ondertekend.
- ☑ 2. Verslagen en/of samenvattingen hiervan, die beperkt openbaar zijn (binnen eigen hogeschool/universiteit of studierichting).
- ☐ 3. Verslagen en/of samenvattingen hiervan, die publiek en dus volledig openbaar zijn en dus ook op internet gepubliceerd mogen worden.

In voorkomende gevallen moet een aangepast verslag voor de opleidingsinstelling worden gemaakt.

* De stagebegeleider van Thales geeft aan in welke categorie het verslag wordt ingedeeld. Het verslag wordt door de stagebegeleider afgetekend.

Akkoord:	Akkoord: (alleen bij categorie 1)
Harmanny	
(Stagebegeleider Thales)	(Opleidingsinstelling)

Delft, 18 juni 2014

(plaats/datum)

THALES

Samenvatting

Spoorwandelaars zijn een groot maatschappelijk probleem, zo zijn er elk jaar 200 zelfdodingen [1]. Door spoorwandelaars lopen treinreizigers elke dag veel vertraging op en daarnaast zorgen spoorwandelaars voor veel directe en indirecte kostenposten. Een goede oplossing is er op dit moment niet. Een RADAR-systeem lijkt een mogelijke oplossing.

Het doel van dit onderzoek is dan ook om tot een prototype van een RADAR-systeem voor spoorbewaking te komen. Deze thesis beschrijft de signaalverwerking die nodig is voor dit prototype. In de thesis wordt gekeken naar de nodige voorbewerking van het signaal uit de gebruikte RADAR-module, vervolgens de acquisitie van data die te verkrijgen is uit dit signaal en tot slot de verwerking van de verkregen data met behulp van zowel hardware als software.

De voorbewerking is gerealiseerd door middel van filters en stelt het systeem in staat om objecten die zich 10 m tot 1500 m van het systeem bevinden waar te nemen. Verder is de data-acquisitie geïmplementeerd met een *Innovative P25M* kaart en verwerking met behulp van een CPU en een stuk software in C++.

Er kan geconcludeerd worden dat het gerealiseerde prototype in de huidige staat nog verre van af is. Er is nog geen detectie of classificatie algoritme geïmplementeerd. Met ongeveer 46 % processorkracht en 884 ms aan tijd beschikbaar is het hoogstwaarschijnlijk wel mogelijk om in ieder geval detectie binnen de gestelde eis van 1 seconde te implementeren met de huidige opstelling. Voor classificatie is het met het huidige resultaat niet mogelijk om te bepalen of dit gerealiseerd kan worden binnen de gestelde eis van 5 seconden.

Een belangrijke vervolgstap zou uiteraard het implementeren van een detectie en classificatie algoritme zijn. Daarna kan het systeem in een relevante omgeving getest worden.

Inhoudsopgave

Inleiding **6**

1 Probleemstelling **8**
1.1 Type RADAR . . . 8
1.2 RADAR-module . . . 9
1.3 Gebruikte parameters . . . 10
1.4 Detectie en classificatie . . . 11
1.5 Programma van Eisen . . . 11

2 Ontwerp **13**
2.1 Systeemoverzicht . . . 13
2.2 Voorbewerking signaal . . . 14
2.3 Data-acquisitie . . . 17
2.4 Verwerkingshardware . . . 20
2.5 Verwerkingssoftware . . . 22

3 Resultaten **25**

4 Discussie **27**
4.1 Discussie resultaten . . . 27
4.2 Verbeterpunten . . . 27

5 Conclusie **29**
5.1 Beoogd gebruik . . . 29
5.2 Syteemontwerp . . . 29
5.3 Aanbevelingen . . . 30

Afkortingen **30**

Literatuur **30**

A Datasheet RFBeam KOR-001 **33**

B Datasheet Innovative P25M **40**

Lijst van figuren

1.1 Een situatieschets van een mogelijke opstelling van de RADAR-module langs het spoor . . . 8
1.2 Een overzicht van het uitgezonden, ontvangen en het resulterende beatsignaal voor een FMCW-RADAR . . . 9
1.3 Vooraanzicht van de gebruikte KOR-001 RADAR-module . . . 10
1.4 µdopplerspectrum van een persoon die richting de RADAR-module loopt . . . 11

2.1 Totaal systeemoverzicht verwerking . . . 13
2.2 Blokschema van de voorbewerking . . . 15
2.3 Het circuit zoals het is gebruikt voor de voorbewerking van het RADAR-signaal . 15
2.4 Frequentiekarakteristiek van het voorbewerkingscircuit zoals gesimuleerd met LT-Spice. De ononderbroken lijn geeft de vermogensoverdracht weer en de gestippelde lijn de fasedraaiing. . . . 17
2.5 Implementatie van het voorbewerkingscircuit (voor één kanaal) op een breadboard 18
2.6 Bovenaanzicht van de P25M data-acquisitiekaart *(foto door Innovative Integration)* 19
2.7 Overzicht van de lengte in samples van de frequentiesweeps en daaronder het bijbehorende triggersignaal . . . 20
2.8 De gedeelde buffer zoals hij wordt gebruikt voor tussentijdse data-opslag. . . . 23
2.9 Blokschema van de dataflow in de verwerkingssoftware . . . 24
2.10 Interliniëring van de data zoals die wordt ontvangen op de computer. Elk vakje stelt een sample voor . . . 24

Lijst van tabellen

1.1 De gebruikte paramters voor de FMCW-modus [2] 10
1.2 De limieten voor het systeem die volgen uit de gegeven parameters [2] 11

2.1 De componentwaarden van het voorbewerkingscircuit zoals zij uiteindelijk zijn gebruikt 16

3.1 Specificaties van de computer gebruikt voor het draaien van de testsoftware . . . 25
3.2 Processorbelasting bij de verschillende testsituaties 25

Inleiding

"Videobewaking tegen spoorwandelaars", was de kop van een groot artikel in de krant Metro op 23 april 2014 [3]. Het artikel in de krant gaat in op een groot maatschappelijk probleem: spoorwandelaars. Elk jaar zijn er ongeveer 200 zelfdodingen op het spoor [1]. Dit zorgt voor een grote directe kostenpost, voor bijvoorbeeld het schoonmaken van de trein en het opruimen van de resten van de betreffende persoon. Maar nog belangrijker zijn de grote vertragingen die dit dagelijks oplevert, waardoor reizigers veel te laat op hun bestemming arriveren. En natuurlijk de verschrikkelijke impact die een frontale botsing met een persoon maakt op een machinist en de reizigers in een trein.

Naast zelfdoding leveren mensen in de buurt van het spoor nog meer problemen op. Zo moet een trein uit voorzorg al langzamer rijden wanneer zich personen langs het spoor bevinden. Dit zorgt al voor de nodige vertragingen. Een ander groot probleem is koperdiefstal. Met de stijgende koperprijzen is koperdiefstal steeds lucratiever geworden voor criminelen. De diefstal van koper langs het spoor is niet alleen een grote kostenpost, maar zorgt ook voor gevaarlijke situaties door bijvoorbeeld niet meer functionerende wissels of seinen. Zo is in 2011 bij Zevenaar een trein ontspoort als gevolg van koperdiefstal [4].

State-of-the-art

Zoals de hierboven genoemde krantenkop wel laat zien wil ook ProRail graag het probleem van mensen langs het spoor oplossen. Op dit moment wordt dit vooral getracht door hekken te plaatsen langs het spoor [3], dit is eigenlijk niet meer dan een eerste barrière tegen spoorwandelaars. Verder worden er vlak na grote incidenten soms mensen ingezet om delen van het spoor te bewaken. Doordat met de inzet van mensen per persoon maar een klein gebied bewaakt kan worden zijn de arbeidskosten hiervoor hoog.

Een andere mogelijke oplossing is videobewaking. Door langs heel het spoor camera's te plaatsen kan in principe heel het spoor in de gaten gehouden worden. Doordat het voor iemand in de meldkamer niet mogelijk is om alle beelden die worden gemaakt tegelijk te bekijken is men wellicht te laat om in te grijpen in het geval van spoorwandelaars. Dit is deels op te lossen door gebruik te maken van software die de camerabeelden analyseert en zo bepaalt of zich mensen langs het spoor bevinden. Het grootste nadeel van het realiseren van spoorbewaking met behulp van camera's is de weersafhankelijkheid. Sowieso is met een camera slechts een gebied van enkele tientallen meters te overzien, maar het weer kan hier in negatieve zin nog veel invloed op hebben. Zo wordt het zicht van een camerabeeld ernstig verstoord bij zware regenval, mist, felle zon(reflectie) of gewoon wanneer het nacht is.

RADAR

Zoals hierboven beschreven kleven er nog grote nadelen aan de huidige manieren om het aantal mensen op en langs het spoor terug te dringen. Radio Detection And Ranging (RADAR) technologie zou hiervoor een uitkomst kunnen bieden. Zelfs met relatief goedkope RADAR-modules is een veel groter gebied te overzien (tot een paar kilometer) dan met camera's. Ook hebben RADAR-modules niet tot nauwelijks last van de weersomstandigheden en functioneren bijvoorbeeld prima bij nacht of dichte mist, waardoor een gebied ook dan bewaakt kan worden. Daarnaast heeft een RADAR-systeem de potentie om vrijwel autonoom te opereren. Ideaal gezien stelt het systeem zelf vast of zich een persoon (of ook een ander gevaarlijk object) in het te bewaken gebied bevindt en wat vervolgens een gepaste actie is die moet worden uitgevoerd.

Het is dus zeker interessant om onderzoek te doen naar de mogelijkheden voor spoorbewaking met behulp van RADAR-technologie. Het doel van dit bachelor afstudeerproject is dan ook om een prototype te maken van een RADAR-systeem dat ingezet kan worden om het spoor te bewaken. Dit is in drie delen opgedeeld. Het eerste deel bestaat uit de algoritmiek die nodig is voor het prototype [2]. Daarnaast is er nog een deelonderzoek gedaan naar de hardware en benodigde randapparatuur [5]. Het laatste onderdeel wordt in deze thesis behandelt en gaat over de signaalverwerking voor het systeem.

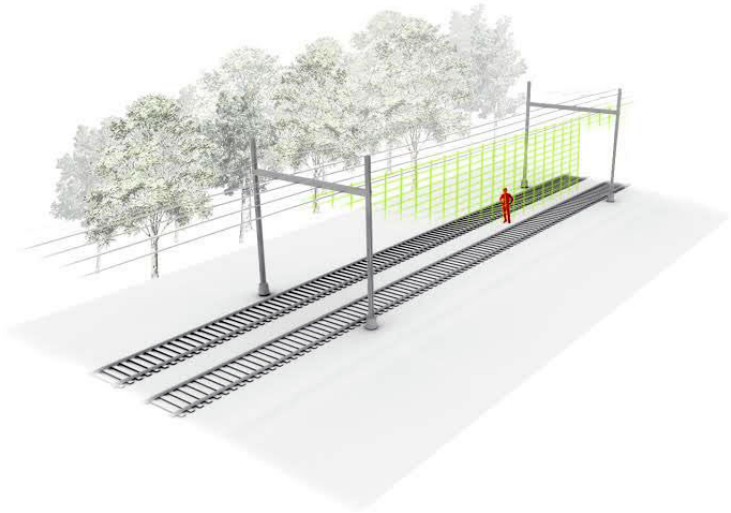
Opbouw thesis

Zoals gezegd behandelt deze thesis de verwerking van het signaal van een RADAR-systeem voor spoorbewaking. In hoofdstuk 1 wordt de probleemstelling van dit project uiteengezet, evenals de eisen die aan het ontwerp gesteld moeten worden. In hoofdstuk 2 een beschrijving van het systeem gegeven en daarna de ontwerpkeuzes voor de verschillende onderdelen van het systeem gemotiveerd. Vervolgens worden de resultaten besproken in hoofdstuk 3. De discussie volgt in hoofdstuk 4, hier worden verbeterpunten en aanbevelingen voor een vervolg onderzoek aangehaald. Tot slot wordt de thesis in hoofdstuk 5 afgesloten met de conclusie, waar terug wordt gekeken op de gestelde eisen uit hoofdstuk 1.

Hoofdstuk 1

Probleemstelling

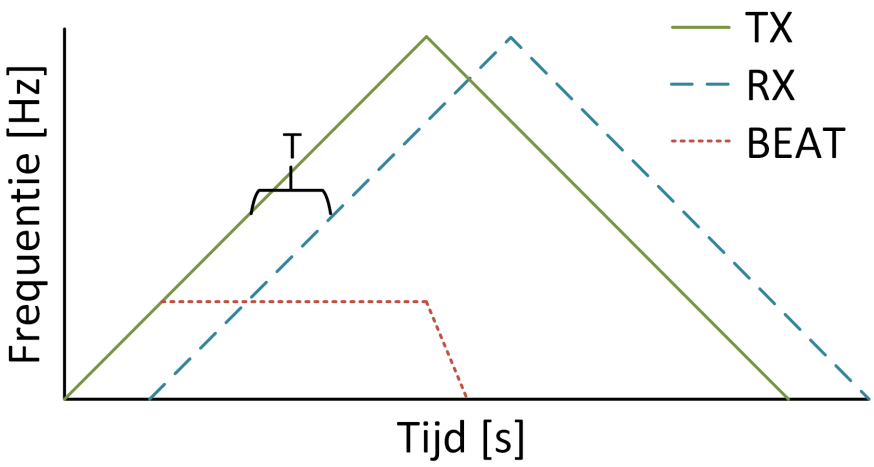
Om het genoemde probleem van indringers rond het spoor te verhelpen, wordt een RADAR-systeem ontworpen dat objecten kan detecteren en mensen hieruit kan herkennen. Door dergelijke modules aan de steunen voor bovenleidingen te bevestigen, kan het hele spoornetwerk worden bewaakt (zie figuur 1.1).



Figuur 1.1: Een situatieschets van een mogelijke opstelling van de RADAR-module langs het spoor

1.1 Type RADAR

Er wordt gebruik gemaakt van FMCW-RADAR. Bij dit type RADAR wordt er een continue golf uitgezonden waarvan de frequentie lineair sweept. Het ontvangen signaal wordt vervolgens gemixt met het verzonden signaal. Het signaal dat dan ontstaat is het beatsignaal. Een overzicht is te vinden in figuur 1.2. Aangezien de frequentie van het beatsignaal lineair afhankelijk is van de tijd die het RADAR-signaal onderweg is, welke dus ook afhankelijk van de afstand tot een object (vergelijking 1.1). Afstandsinformatie kan dus uit het beatsignaal gewonnen worden door een FFT uit te voeren op het beatsignaal (vergelijking 1.2).



Figuur 1.2: Een overzicht van het uitgezonden, ontvangen en het resulterende beatsignaal voor een FMCW-RADAR

$$f_{beat} = f_{TX} - f_{RX} = T \cdot \frac{df_{TX}}{dt} = 2\frac{s}{c} \cdot \frac{df_{TX}}{dt} \tag{1.1}$$

$$s = \frac{c \cdot f_{beat}}{2 \cdot \frac{df_{TX}}{dt}} \tag{1.2}$$

Uit de data van een FMCW-RADAR kan ook worden gehaald wat de dopplerverschuiving van het ontvangen signaal is. De dopplerverschuiving heeft een directe relatie tot de snelheid van het object waar het signaal tegen weerkaatst is (vergelijking 1.3) [6].

$$f_d = 2v\frac{f_{TX}}{c - v} \approx 2v\frac{f_{TX}}{c} \qquad (v \ll c) \tag{1.3}$$

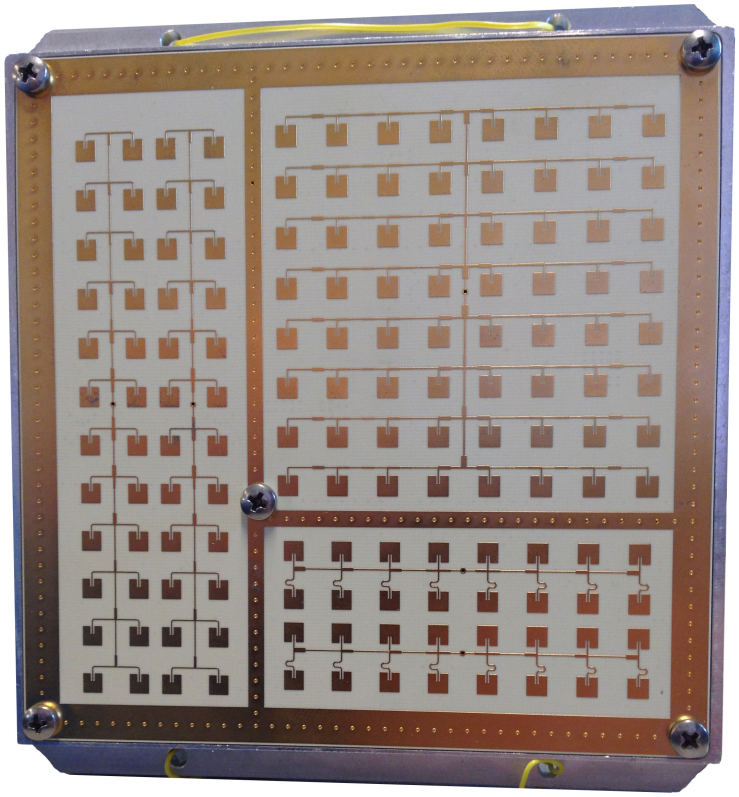
Deze dopplerinformatie kan worden verkregen door een tweede FFT uit te voeren, maar nu over hetzelfde sample, uit meerdere opeenvolgende frequentiesweeps [7].

1.2 RADAR-module

Als RADAR-module is de KOR-001 van RFBeam Microwave GmbH gebruikt (figuur 1.3). Dit is een module die op maat is gemaakt voor een klant van het bedrijf, maar vervolgens wel op kleine schaal verder is verkocht. Nadeel hiervan is dat de documentatie gebrekkig is. Het enige wat beschikbare is een datasheet [8] en contact met de technische afdeling van RFBeam.

Om in FMCW-modus te kunnen werken, heeft de KOR-001 een ramp-up signaal nodig. Zodra dit signaal hoog is, zal de frequentie van het uitgezonden signaal toenemen. Wanneer het laag is, neemt de frequentie af. Dit signaal kan in de verwerking ook gebruikt worden om te bepalen wanneer een frequentiesweep gestart is.

De module heeft vier antennes. Allemaal hebben ze als uitgang een zogenaamd I-signaal. Twee antennes voeren bovendien ook een Q-signaal (complex geconjungeerde van I) uit. Voor het systeem worden de Q-signalen buiten beschouwing gelaten, omdat implementatie hiervan te



Figuur 1.3: Vooraanzicht van de gebruikte KOR-001 RADAR-module

complex is. De overige signalen worden meegenomen in het ontwerp, zodat ze eventueel bij de detectie en classificatie gebruikt kunnen worden.

1.3 Gebruikte parameters

In het aanverwante deelonderzoek [2] zijn de parameters voor de FMCW-modus vastgelegd. Deze zijn te vinden in tabel 1.1.

Tabel 1.1: De gebruikte paramters voor de FMCW-modus [2]

Parameter	**Waarde**
Sweepfrequentie	$250\,\mathrm{GHz\,s^{-1}}$
Sweeptijd	$100\,\mu\mathrm{s}$
Sweeps per 2^e FFT	800

De consequenties van deze parameters voor de limieten van het systeem zijn te vinden in tabel 1.2.

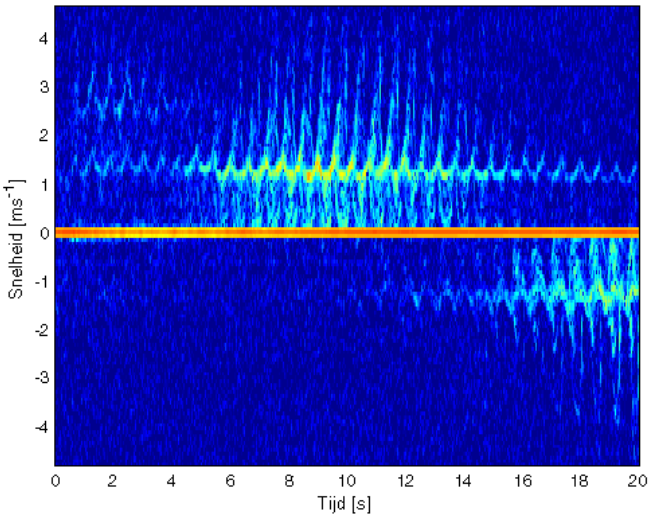
Tabel 1.2: De limieten voor het systeem die volgen uit de gegeven parameters [2]

Limiet	**Waarde**
Maximaal te meten afstand	1.8 km
Maximale afstandsresolutie	6.00 m
Maximaal te meten snelheid	15.6 $\mathrm{m\,s^{-1}}$
Maximale snelheidsresolutie	0.0781 $\mathrm{m\,s^{-1}}$

1.4 Detectie en classificatie

Het algoritme voor het detecteren van objecten en het herkennen van mensen staat beschreven in [2]. Voor detectie wordt gebruik gemaakt van een drempelmap. Voor elk afstands-snelheidpaar wordt een drempelwaarde opgesteld waarboven de gemeten waarde moet liggen om waarschijnlijk een reflectie van een object te zijn.

Classificatie gebeurt door naar µdoppler te kijken. Aangezien armen en benen tijdens het lopen met een andere snelheid bewegen dan de rest van het lichaam, geeft dit een karakteristiek resultaat in het dopplerspectrum [9].



Figuur 1.4: µdopplerspectrum van een persoon die richting de RADAR-module loopt

Voor zowel detectie als classificatie is het van belang dat dit niet te lang duurt. Een persoon loopt immers maar een beperkte tijd in het zicht van de RADAR en moet dus gedetecteerd worden in die tijdspanne. Bovendien geldt dat een snellere detectie het ook beter mogelijk maakt om te reageren op de situatie.

1.5 Programma van Eisen

Om het probleem zoals beschreven in dit hoofdstuk op te kunnen lossen zijn er een aantal eisen opgesteld waaraan voldaan moet worden. Aan deze eisen moet worden voldaan om uiteindelijk tot een (goed) werkend prototype te komen, dat ook daadwerkelijk een oplossing biedt voor het beschreven probleem.

1.5.1 Eisen vanuit beoogd gebruik

1.5.1.1 Het systeem moet een digitale uitvoer hebben die aangeeft of er een object door de sensoren wordt gedetecteerd.
1.5.1.2 Het systeem moet kunnen aangeven of het gedetecteerde object een mens is of niet.
1.5.1.3 Het systeem moet een afstandsbereik hebben tussen de 10 m en 1500 m.

1.5.2 Eisen met betrekking tot het te ontwerpen systeem zelf

1.5.2.1 Het systeem maakt gebruik van de signalen zoals zij uit de RFBeam KOR-001 RADAR-module komen.
1.5.2.2 De reactietijd voor het detecteren van objecten moet minder zijn dan 1 seconde.
1.5.2.3 De reactietijd voor het herkennen van personen moet minder zijn dan 5 seconden.
1.5.2.4 Het systeem implementeert het algoritme voor detectie en classificatie zoals beschreven in [2].
1.5.2.5 Het formaat van het systeem moet beperkt blijven, zodat het systeem eenvoudig langs het spoor gemonteerd kan worden.

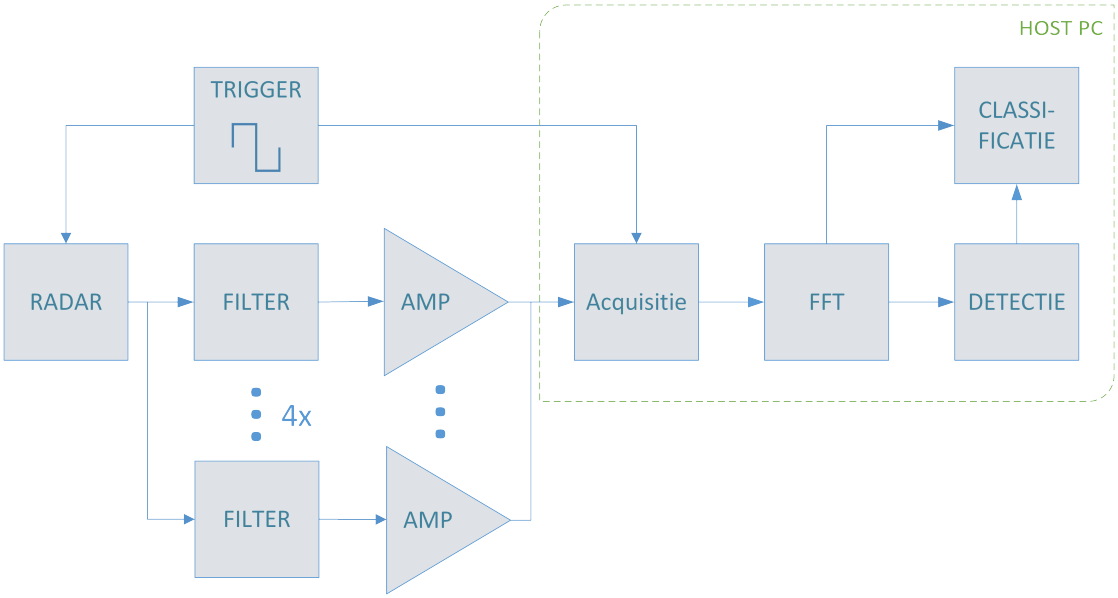
Hoofdstuk 2

Ontwerp

In dit hoofdstuk worden er eisen gesteld aan het ontwerp van verschillende onderdelen van het systeem. Aan de hand van deze eisen worden er verschillende opties behandeld en vervolgens de gemaakte keuze gemotiveerd. Tenslotte wordt de implementatie van de keuze in het prototype besproken en wordt er behandeld wat voor resultaten dit oplevert.

2.1 Systeemoverzicht

De verwerking van het signaal is uiteraard onder te brengen in verschillende stappen en daarnaast ook een onderdeel van het grotere systeem. In deze sectie wordt daarom gekeken hoe dit systeem globaal is opgebouwd.



Figuur 2.1: Totaal systeemoverzicht verwerking

In figuur 2.1 is te zien hoe het verwerking gedeelte en alles dat hierbij hoort is opgedeeld in dit project. Zoals gezegd is de RADAR-module al gegeven. Het *Trigger* blok valt buiten de scope

van deze thesis en wordt verder behandeld in [5]. Net als de blokken *Classificatie* en *Detectie*, deze worden verder behandeld in [2].

Zoals is aangegeven valt een deel van de blokken (in de gekozen implementatie) onder een *Host PC*, de overige blokken zijn geïmplementeerd in de vorm van een stuk hardware die aan de *Host PC* wordt verbonden.

De overige blokken die wel binnen de scope van deze thesis vallen zijn op te delen in een aantal onderwerpen en dat is dan ook gedaan. Zo wordt in sectie 2.2 (Voorbewerking signaal) de filters en versterkers behandeld die het signaal uit de RADAR-module geschikt maken voor de data-acquisitie. Vervolgens wordt in sectie 2.3 (Data-acquisitie) uitgelegd hoe de (analoge) signalen die dan overblijven worden omgezet naar bruikbare data. Tenslotte wordt deze data verwerkt zodat er uiteindelijk detectie en/of classificatie kan plaatsvinden, dit is voornamelijk een FFT en is door middel van secties 2.4 (Verwerkingshardware) en 2.5 (Verwerkingssoftware) uitgesplitst in een gedeelte betreffende de hardware en een gedeelte over de software.

2.2 Voorbewerking signaal

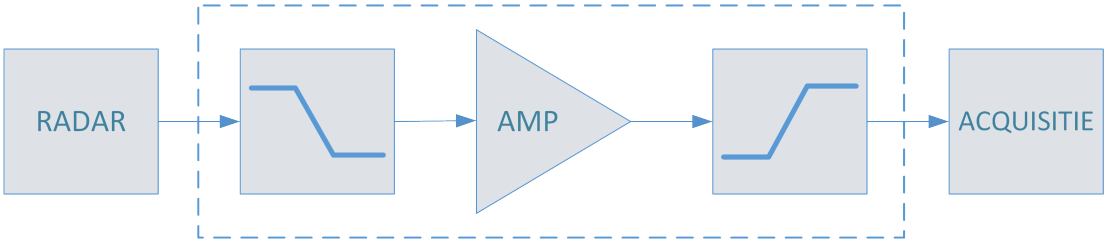
Zoals vermeld in sectie 1.2 is de RADAR-module gegeven in de vorm van de RFBeam KOR-001. Uit deze RADAR-module komt een analoog signaal. Voordat dit signaal naar de door ons gebruikte ADC kan (waarover meer in sectie 2.3) moet het worden voorbewerkt, zodat het signaal aan de eigenschappen voldoet die de ADC verwacht. In deze sectie wordt besproken wat de eisen zijn waaraan de voorbewerking van het signaal moet voldoen en welke opties hiervoor beschikbaar waren. Uiteindelijk wordt besproken voor welke optie er is gekozen, hoe deze is geïmplementeerd en tenslotte wat voor resultaten dit opleverde.

2.2.1 Eisen

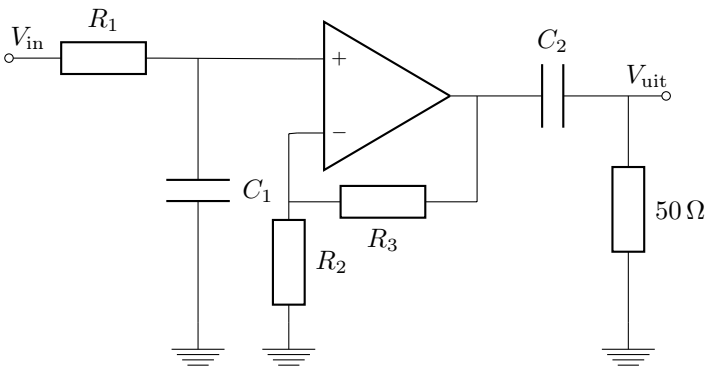
- Het signaal uit de gebruikte RADAR-module (KOR-001) bevat frequentiecomponenten tot ten hoogste 3 MHz [8]. Dus moet de voorbewerking in ieder geval frequentiecomponenten tot 3 MHz doorlaten. Hogere frequentiecomponenten kunnen gefilterd worden, want die bevinden zich niet op het signaal uit de RADAR-module, dus zullen volledige uit (ongewenste) ruis bestaan.
- De belasting op de uitgang van de RADAR-module moet minimaal 1 kΩ zijn, dit volgt uit de datasheet van de gebruikte RADAR-module [8].
- Het signaal dat binnenkomt op de ADC moet ten alle tijden binnen de ±2 V liggen, dit volgt uit de datasheet van de gebruikte ADC [10].
- Het signaal dat wordt uitgestuurd door de gebruikte RADAR-module bevat een DC-offset. Om het bereik van de ADC optimaal te benutten, dient deze er uitgehaald te worden.
- Het systeem moet uitgaan van een belastingsweerstand van 50 Ω, aangezien dit de ingangsweerstand van de gebruikte data-acquisitiekaart is (zie sectie 2.3.3).

2.2.2 Ontwerp

Voor het behalen van de verschillende eisen, is gebruik gemaakt van het circuit uit figuur 2.3. Dit circuit bestaat, van links naar rechts bekeken, uit drie onderdelen: een laagdoorlaatfilter, een versterker en een hoogdoorlaatfilter, wat ook schematisch is weergegeven in een blokschema in figuur 2.2. Deze onderdelen zullen hieronder een voor een besproken worden.



Figuur 2.2: Blokschema van de voorbewerking



Figuur 2.3: Het circuit zoals het is gebruikt voor de voorbewerking van het RADAR-signaal

Laagdoorlaatfilter

Het laagdoorlaatfilter, bestaande uit R_1 en C_1, heeft als doel om alle signalen boven de samplefrequentie weg te filteren. Dit voorkomt aliasing. De kantelfrequentie (f_c) wordt dan ook ingesteld op 3 MHz.

$$f_c = \frac{1}{2\,\pi\,R_1\,C_1} = 3\,\text{MHz} \tag{2.1}$$

Voor hoge frequenties geldt dat de ingangsimpedantie van het circuit gegeven wordt door R_1. Aangezien deze impedantie altijd hoger dan 1 kΩ moet zijn, wordt R_1 gekozen op 1 kΩ. Uit vergelijking 2.1 volgt dan dat C_1 gelijk moet zijn aan 53 pF.

Hoogdoolaatfilter

Het hoogdoorlaatfilter, bestaande uit C_2 en de ingangsimpedantie (50 Ω) van de ADC, heeft als doel om de laagfrequente signalen te verzwakken. Deze signalen komen namelijk van objecten die dichtbij staan en hebben dus een grote amplitude. Door deze signalen weg te filteren, kan heel het overige signaal versterkt worden, waardoor het beter binnen het bereik van de ADC valt. Bovendien haalt dit filter de DC-offset die aanwezig is op het signaal weg.

De frequentie van het beatsignaal is volgens vergelijking 2.2 gelinkt aan de afstand tot een object. Als dus alle objecten met een afstand kleiner dan 10 m van het systeem moeten worden weggefilterd, resulteert dit in een kantelfrequentie van 16.7 kHz.

$$f_{beat} = 2\,\frac{s}{c}\,\frac{df_{TX}}{dt} \tag{2.2}$$

Uitgaande van een weerstand van 50 Ω, kan C_2 dan uit vergelijking 2.1 bepaald worden. De waarde van C_2 komt hiermee uit op 191 nF.

Versterker

Het versterkerdeel, bestaande uit de opamp, R_2 en R_3, dient ervoor om de maximale amplitude van het signaal te vergroten naar 2 V. Om de gewenste versterking te bepalen, is de versterker eerst vervangen door een spanningsvolger. Vervolgens is de RADAR aangezet en is hier een hoekreflector voor bewogen op verschillende afstanden. Meer reflectie dan van deze reflector zal in de praktijk niet voorkomen, dus dit simuleert goed een extreme situatie. Vervolgens is de maximale amplitude met een oscilloscoop gemeten op 1.3 V. Hiermee komt de gewenste versterkingsfactor dus op $\frac{2.0\,\mathrm{V}}{1.3\,\mathrm{V}} = 1,5$.

Als opamp is de AD828 gekozen. Dit is een videoversterker, wat ervoor zorgt dat hij kan werken in het gewenste frequentiebereik. Tevens zorgt de beschikbaarheid in een DIL-package voor goede mogelijkheid om te testen.

De versterking van het gekozen circuit wordt gegeven door vergelijking 2.3. In de datasheet van de opamp wordt aangeraden om weerstanden kleiner dan 1 kΩ te gebruiken [11]. Als voor R_2 800 Ω wordt gekozen, volgt hieruit dat R_3 400 Ω moet zijn.

$$A = 1 + \frac{R_3}{R_2} \tag{2.3}$$

Voeding

Als voeding voor de opamp wordt ±12 V gebruikt. Dit omdat de positieve spanning toch al aanwezig is voor de RADAR-module en er dus alleen nog een −12 V aansluiting toegevoegd hoeft te worden. Verder raadt de datasheet van de opamp aan om 0.1 µF ontkoppelcondensatoren bij de voedingsingang te plaatsen [11]. Dit is dan ook in het ontwerp verwerkt.

2.2.3 Simulatie

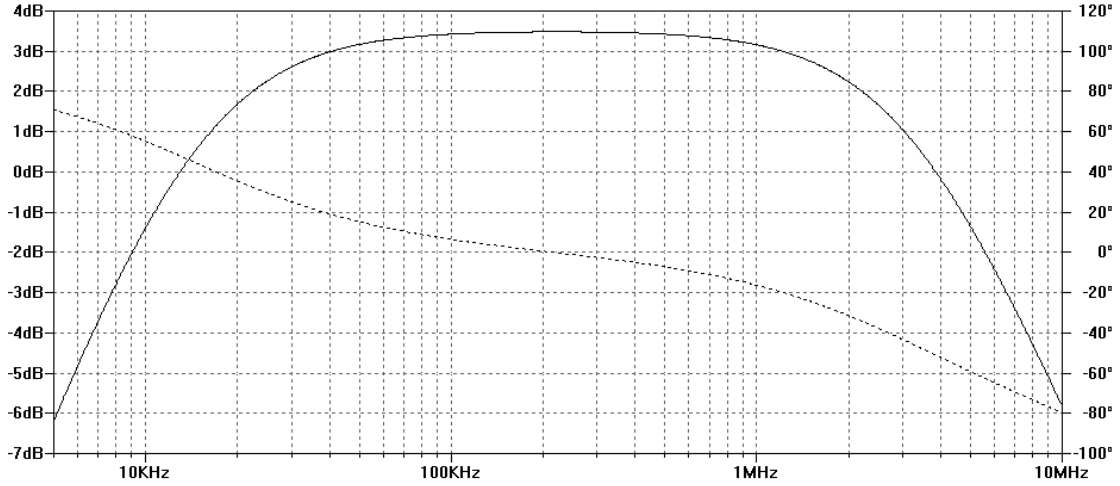
Het ontworpen circuit is in LTSpice gezet en vervolgens gesimuleerd. Het resultaat is te vinden in figuur 2.4. De componentwaarden zijn ingevoerd zoals zij zijn gebruikt bij de implementatie (zie tabel 2.1).Uit de simulatie blijkt dat de gekozen componentwaarden tot de juiste overdracht leiden. Voor de opamp is het AD828 SPICE-model van Analog Devices gebruikt [12].

2.2.4 Implementatie

Het deelsysteem is uiteindelijk uitgevoerd op een breadboard. Het resultaat hiervan is te zien in figuur 2.5. Voor de componentwaarden zijn de waarden gepakt die zo dicht mogelijk bij de berekende waarden liggen, te vinden in tabel 2.1.

Tabel 2.1: De componentwaarden van het voorbewerkingscircuit zoals zij uiteindelijk zijn gebruikt

Component	**Waarde**
R_1	1 kΩ
R_2	800 Ω
R_3	400 Ω
C_1	47 pF
C_2	220 nF



Figuur 2.4: Frequentiekarakteristiek van het voorbewerkingscircuit zoals gesimuleerd met LT-Spice. De ononderbroken lijn geeft de vermogensoverdracht weer en de gestippelde lijn de fase-draaiing.

Tevens is er een begin gemaakt aan het ontwerpen van een Printed Circuit Board (PCB). Het voordeel van een PCB ten opzichte van het gebruik van een breadboard is dat er veel minder losse draden zijn die voor storing kunnen zorgen. Daarnaast wordt de gehele opstelling steviger omdat alle onderdelen permanent gesoldeerd worden. Er is echter nog te weinig vordering gemaakt met dit PCB ontwerp om het hier te tonen.

2.3 Data-acquisitie

Nadat de signalen uit de RADAR-module zijn gefilterd en gebufferd, moeten de signalen naar het digitale domein worden gebracht. Deze sectie gaat over die stap in het systeem.

2.3.1 Eisen

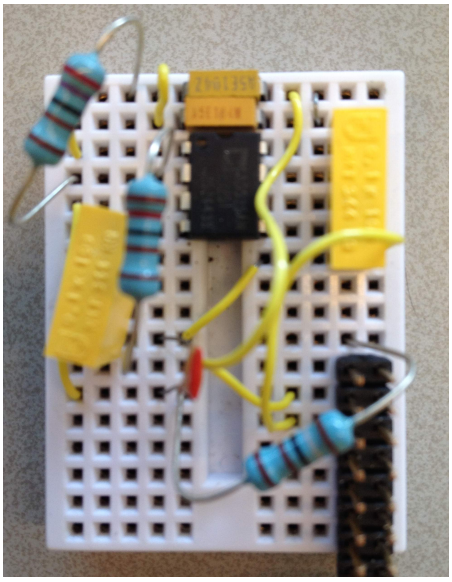
Samplefrequentie

De maximale frequentie van het signaal dat uit de RADAR-module komt is 3 MHz. Volgens Nyquist-Shannon theorema moet de minimaal samplefrequentie dan ook 6 MHz zijn om het signaal volledig te samplen.

Dynamisch bereik

Het vermogen van het signaal dat uit de RADAR-module komt, neemt met een vierde macht af met de afstand waarop het gereflecteerd is (zie 2.4). Als al het signaal dat reflecteert van een afstand van minder dan 10 m wordt weggefilterd, dan verschilt het vermogen van het sterkste en zwakste signaal dat wordt ontvangen 87 dB van elkaar. Om het signaal niet in de ruis te laten verdrinken, dient de signaal-ruisverhouding van de ADC dus groter te zijn dan 87 dB.

$$P_{rx} = \frac{P_{tx}\, G^2\, \sigma\, \lambda^2}{(4\,\pi)^3\, R^4} \propto \frac{1}{R^4} \tag{2.4}$$



Figuur 2.5: Implementatie van het voorbewerkingscircuit (voor één kanaal) op een breadboard

$$\frac{P_{rx,10m}}{P_{rx,1,5km}} = 10 \log \frac{1500^4}{10^4} = 87.0\,\text{dB} \tag{2.5}$$

Datalink

De gesamplede data moet kunnen worden verwerkt. Het is dus een vereiste dat de data moet kunnen worden doorgestuurd naar de software die de verwerking doet. Uitgaande van een samplefrequentie van 6 MHz, het gebruik van 16 bits per sample en het uitlezen van 4 kanalen tegelijk, is een minimale bussnelheid van 48 MBps benodigd.

2.3.2 Opties

ADC

Wat betreft de ADC is er een aantal mogelijkheden beschikbaar. De twee voornaamste opties voor de te behalen samplefrequentie zijn SAR en pipelined ADC's [13]. Het voornaamste voordeel van de tweede optie is dat deze hogere samplefrequenties kan halen. Hier staat echter tegenover dat deze optie wel een hogere latency heeft. Dit is voor deze toepassing echter geen probleem. Een hogere samplefrequentie geeft de mogelijkheid om te oversamplen, wat de SNR kan verbeteren [14].

Datalink

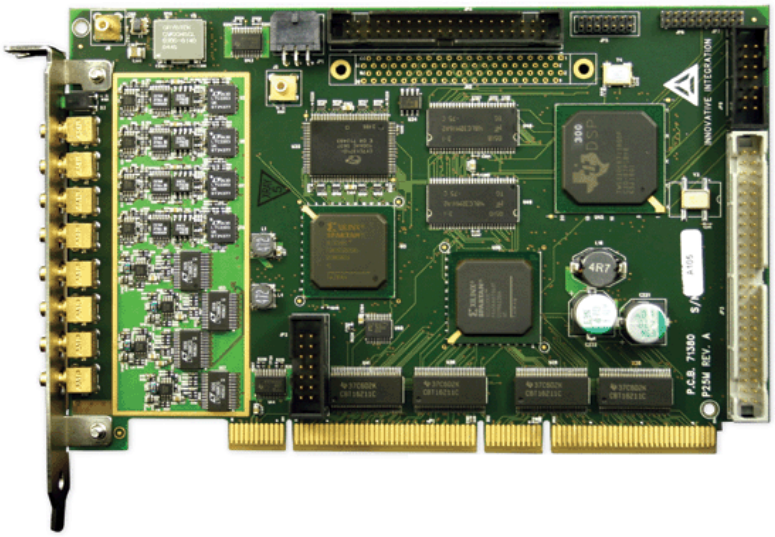
Voor het doorsturen van de data naar de software zijn meerdere opties beschikbaar. Wat betreft de te behalen overdrachtsnelheid, zijn er de volgende algemeen gebruikte opties: USB 3.0, Firewire b en PCI [15]. Een groot voordeel van de derde optie, is dat deze DMA (via bus-mastering) ondersteunt [16]. Gebruik hiervan zorgt ervoor dat de belasting voor de CPU vermindert op het moment dat er data wordt doorgestuurd.

Geheel

Voor het totaalplaatje van dit onderdeel is het natuurlijk mogelijk om eigenhandig iets te ontwerpen of om voor een off-the-shelf oplossing te gaan. Onder andere het bedrijf Innovative Integration maakt hardware die op de gestelde hoge frequentie kan samplen.

2.3.3 Keuze

Uiteindelijke is gekozen om gebruik te maken van de *Innovative P25M* (figuur 2.6) voor data-acquisitie. Deze kaart voldoet aan de gestelde eisen, met bijvoorbeeld een signaal-ruisverhouding van meer dan 90 dB [10]. Deze kaart bevat naast 4 analoge ingangen ook een externe triggerpoort. Deze poort is handig om te kunnen beginnen met samplen bij het begin van een frequentiesweep. Bovendien bevat deze kaart een DSP en FPGA die gebruikt kunnen worden voor de signaalverwerking (sectie 2.4).



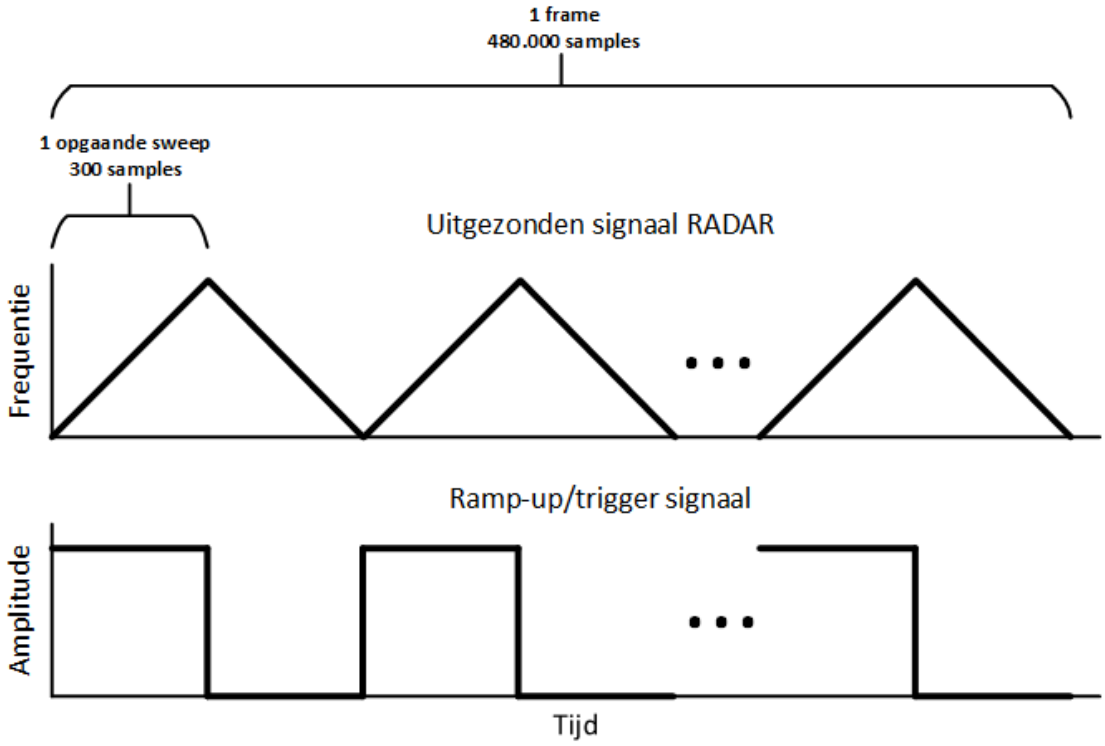
Figuur 2.6: Bovenaanzicht van de P25M data-acquisitiekaart *(foto door Innovative Integration)*

2.3.4 Implementatie

De signalen die uit de voorbewerking komen, worden met SMB-kabels aangesloten op de P25M. Daarnaast wordt ook het triggersignaal dat door de RADAR-module wordt gebruikt aangesloten op de triggerpoort.

Er is uiteindelijk voor gekozen om te samplen op de Nyquist-frequentie van 6 MHz[1]. Hiermee kan dan ook het aantal samples dat een opgaande frequentiesweep in beslag neemt worden berekend (vergelijking 2.6). Er worden in het algoritme 800 opgaande sweeps samengenomen om een tweede FFT over uit te voeren. Dit gecombineerd met het feit dat de helft van de sweeps aflopend is, betekent dat er $(300 \cdot 2 \cdot 800 =)$ 480.000 samples nodig zijn om deze tweede FFT te

[1] In de discussie (hoofdstuk 4) wordt toegelicht waarom er niet wordt overgesampled.



Figuur 2.7: Overzicht van de lengte in samples van de frequentiesweeps en daaronder het bijbehorende triggersignaal

kunnen uitvoeren. Een overzicht hiervan is te vinden in figuur 2.7.

$$n_{\text{samples per sweep}} = T_{\text{sweep}} \cdot f_s = 50\,\mu\text{s} \cdot 6\,\text{MHz} = 300 \text{ samples} \tag{2.6}$$

De acquisitiekaart kan op twee manieren samplen: continous en framed [17]. Bij de eerste modus wordt er continu gesampled en wordt de data vervolgens in pakketjes van variabele grootte naar de computers gestuurd. In de andere modus wordt er gestart met samplen zodra het triggersignaal hoog is en wordt er vervolgens een vooraf bepaald aantal samples opgenomen. Dit is één frame en dit wordt vervolgens in zijn geheel naar de computer gestuurd.

Voor de verdere bewerking van het signaal, moet het bekend zijn waar in het gesamplede signaal een nieuwe frequentiesweep begint. Dit is opgelost door de P25M in de framed modus te laten triggeren op het ramp-up-signaal (zie hoofdstuk 1) van de RADAR-module. Vervolgens wordt er 480.000 keer gesampled voordat er weer wordt gewacht op een nieuwe trigger. Hierdoor komt dus alle benodigde informatie voor het uitvoeren van beide FFT's in een pakketje aan bij de software.

2.4 Verwerkingshardware

Voordat er detectie en/of classificatie kan plaatsvinden moet de data die de data-acquisitie heeft opgeleverd nog worden verwerkt. Hiervoor is in ieder geval een stuk hardware nodig. In deze

sectie worden de hardware mogelijkheden voor deze verwerking, die voornamelijk bestaat uit FFT's besproken.

2.4.1 Eisen

Een eis die aan deze verwerkingshardware gesteld moeten worden is dat de verwerking kan plaatsvinden zonder grote vertragingen en bij voorkeur dat de verwerking realtime gebeurt.

Verder moet het formaat van de hardware zo veel mogelijk beperkt blijven. Immers geldt dat hoe kleiner, hoe makkelijker te monteren. Klein formaat betekent gelijk ook dat de warmteproductie beperkt moet blijven, omdat in een kleine behuizing nou eenmaal eerder oververhitting plaatsvindt.

2.4.2 Opties

Er zijn meerdere opties beschikbaar om de verwerking uit te voeren. Sommige volledig bestaand uit hardware, maar ook een combinatie van hardware en software is mogelijk.

DSP

Voor de verwerking van de RADAR-data zou een Digital Signal Processor (DSP) gebruikt kunnen worden. Deze zijn erg energie-efficiënt en nemen bovendien relatief weinig ruimte in. Vergeleken met andere opties heeft een DSP echter wel weinig rekenkracht [10].

FPGA

Ook kan een Field-Programmable Gate Array (FPGA) gebruikt worden. Een FPGA is ook bijzonder energie-efficiënt en is ook erg klein. Een FPGA kan ook veel rekenkracht bezitten en kan relatief snel zijn in het uitvoeren van berekeningen. Verder is een FPGA erg flexibel, doordat er zeer uiteenlopende hardware mee geïmplementeerd kan worden en deze kan ook relatief snel worden aangepast. [18]

CPU

De Central Processing Unit (CPU) van een computer is ook een goede optie voor het verwerken van de data die een RADAR genereert. Een voordeel van een CPU is dat de verwerking makkelijk te implementeren is met behulp van een stuk software. Nadelen zijn dat de bijbehorende computerkast over het algemeen meer ruimte inneemt dan andere opties en ook een stuk minder energie-efficiënt is dan bijvoorbeeld een FPGA of DSP [18].

GPU

Tenslotte kan de verwerking ook worden gedaan met behulp van de Graphics Processing unit (GPU). De GPU is bij uitstek geschikt om instructies parallel uit te voeren door de vele kleine processors (*streaming processors* genaamd) die een GPU bezit. Doordat een FFT parallel kan worden uitgevoerd heeft een GPU de potentie om de prestaties van de verwerkingshardware aanzienlijk te verbeteren, zeker ten opzichte van een CPU waarmee het slechts zeer beperkt mogelijk is om instructies parallel uit te voeren. De energie en ruimte argumenten komen overeen met die voor een CPU [18].

2.4.3 Keuze

Kijkend naar de opties hierboven lijkt een GPU misschien wel de beste optie. Echter heeft een GPU als nadeel dat deze alleen met data kan werken die in zijn eigen geheugen staat. Dit betekent dat voor en na berekeningen data van en naar het werkgeheugen gekopiëerd moet worden. Deze communicatie zorgt voor overhead die een bottleneck vormt en ervoor zorgt dat het uitvoeren van een FFT op een GPU pas een prestatiewinst oplevert bij FFT's van een paar honderdduizend punten [18]. Voor RADAR verwerking hebben de benodigde FFT's echter veel minder punten. Zo zijn er met de gebruikte hardware voor dit project bij de eerste FFT 512 punten en voor de tweede FFT 1024 punten. Deze getallen zijn gebaseerd op het benodigde aantal punten afgerond naar boven tot de dichtsbijzijnde macht van twee, het FFT algoritme werkt namelijk stukken efficiënter met machten van twee [19]. Met deze hoeveelheid aan punten valt het gebruik van een GPU dus af.

De meeste DSP's hebben simpelweg te weinig rekenkracht om de verwerking van de data zonder grote vertragingen uit te voeren. De data realtime verwerken zou natuurlijk ideaal zijn. Dus blijven een FPGA of CPU over als opties.

Een FPGA, zeker een moderne, heeft genoeg rekenkracht voor de verwerking van de RADAR-data. FPGA's zijn namelijk ook heel geschikt voor het parallel uitvoeren van instructies. Op een FPGA zullen wel veel minder cores passen dan een GPU heeft. Toch is een FPGA in vergelijking met een GPU een stuk sneller voor FFT's met minder datapunten. Dit komt door de hogere throughput die een FPGA heeft, doordat een FPGA de data een stuk sneller kan wegschrijven naar geheugen [18]. Daarnaast verbruikt een FPGA veel minder vermogen voor het uitvoeren van een specifieke taak dan een GPU of CPU, wat een FPGA geschikter maakt om in een kleine behuizing te stoppen.

Zoals vermeld in sectie 2.3.3 is er gebruik gemaakt van een *Innovative P25M* voor data-acquisitie. Deze kaart bevat een DSP en FPGA. Toch is er bij de implementatie van de verwerkingshardware, ondanks de hierboven beschreven voordelen, geen gebruik gemaakt van deze FPGA en ervoor gekozen om de verwerking op een CPU uit te voeren. Dit is gedaan vanwege de gebrekkige documentatie van de FPGA en met name ook de inmiddels zeer verouderde ondersteuning van de beschikbare FPGA. Aangezien de *Innovative P25M* kaart door middel van een PCI connectie al was verbonden met de CPU van een computer was de implementatie hiervan een stuk makkelijker en sneller te realiseren, wat een groot voordeel is gezien de korte tijdsduur van dit project. De verdere verwerking op de CPU is gedaan met een stuk software, dit wordt beschreven in sectie 2.5.

2.5 Verwerkingssoftware

Na de data-acuisitiestap is erg alleen nog een digitaal signaal. Om hier een object in te kunnen detecteren en classificeren, is er verdere verwerking nodig. Dit gebeurt volledig softwarematig. De benodigde stappen hiervoor worden beschreven in de probleemstelling (hoofdstuk 1).

2.5.1 Gemaakte afwegingen

Programmeertaal

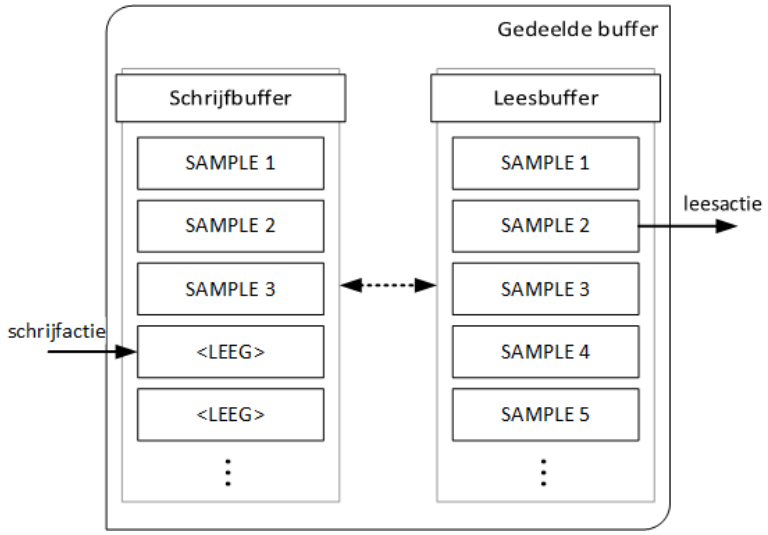
Omdat het om veel data gaat die de software moet verwerken, is ervoor gekozen om een low-level programmeertaal te gebruiken. Bij dit soort talen heb je als programmeur namelijk zelf in de hand hoe het geheugen gebruikt wordt, wat de efficiëntie ten goede komt. Er is uiteindelijk voor C++ gekozen omdat de software behorende bij de P25M daar ook in is geschreven en dit dus de integratie vergemakkelijkt.

ASnap

Bij de P25M wordt standaard een aantal voorbeeldprogramma's geleverd. Een hiervan (ASnap) kan gebruikt worden om de data die wordt gesampled op een computer te loggen. Omdat dit programma de basiscommunicatie met de P25M al bevat, is dit gebruikt als basis. Het programma is zo aangepast dat het de datapakketjes die het binnenkrijgt, doorstuurt naar de in dit hoofdstuk beschreven software.

Multithreading

Bij het ontwerpen van de software is getracht om zoveel mogelijk multithreaded te werken. Voordeel hiervan is dat de software gebruik kan maken van het hedendaagse multicore ontwerp van processoren. Om dit mogelijk te maken, wordt alle data tussentijds opgeslagen in een speciale, gedeelde buffer (zie figuur 2.8). Deze buffer maakt het mogelijk om vanuit verschillende threads gelijktijdig te lezen en te schrijven. Er is een aparte lees- en schrijfbuffer aanwezig. Als de data in de leesbuffer volledig verwerkt is, worden (nadat de huidige schrijfoperatie is afgerond) de twee buffers omgewisseld.



Figuur 2.8: De gedeelde buffer zoals hij wordt gebruikt voor tussentijdse data-opslag.

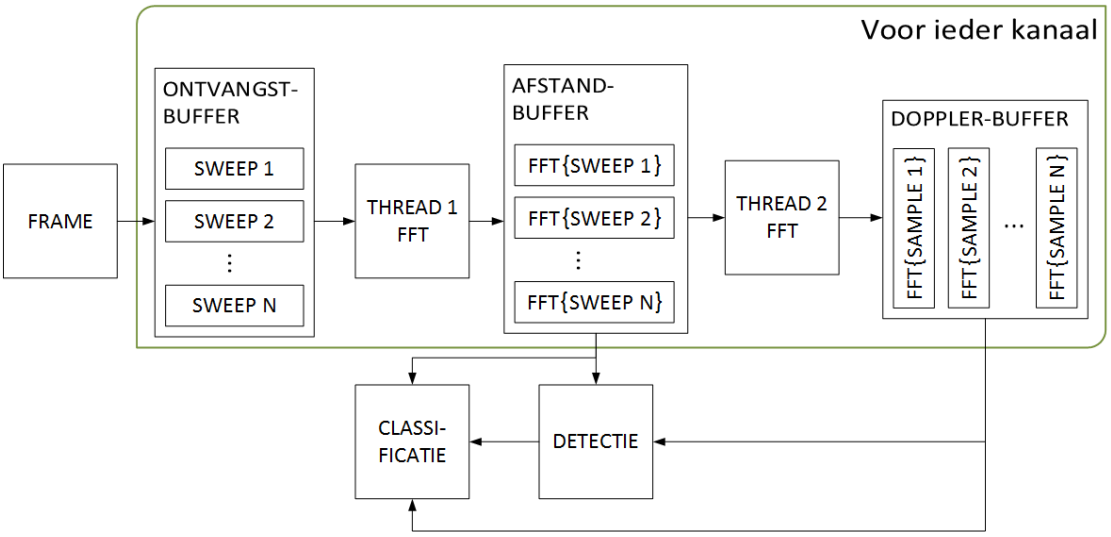
FFTW

In de software wordt in twee stappen een FFT toegepast. Voor het uitvoeren van FFT's zijn verschillende softwarebibliotheken beschikbaar. Er is gekozen om gebruik te maken van FFTW, aangezien deze in bijna alle gevallen het snelst is en bovendien eenvoudig te verkrijgen is [19].

2.5.2 Implementatie

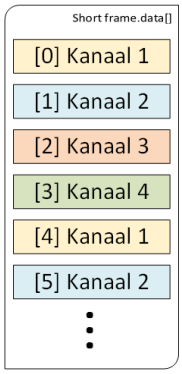
In figuur 2.9 is een overzicht te vinden van de dataflow zoals die plaatsvindt in de software.

De data komt binnen in frames, bestaande uit de sampledata van 800 frequentiesweeps (sectie 2.3.4). Het ontvangen datapakket bevat een array met daarin de data van alle gesamplede kanalen geïnterliniëerd (zie figuur 2.10). In de huidige situatie wordt alle informatie van de verschillende



Figuur 2.9: Blokschema van de dataflow in de verwerkingssoftware

kanalen bij elkaar opgeteld om de signaal-ruisverhouding te verbeteren. De kanalen kunnen echter ook los van elkaar verwerkt worden. In beide gevallen wordt de data opgeslagen in een buffer (ontvangstbuffer). In het geval van meerdere kanalen, heeft elk kanaal zijn eigen buffer en worden de volgende stappen ook los uitgevoerd.



Figuur 2.10: Interliniëring van de data zoals die wordt ontvangen op de computer. Elk vakje stelt een sample voor

Vanuit deze eerste buffer leest thread 1 alle beschikbare data uit. Deze thread voert over alle frequentiesweeps een FFT uit en plaatst deze data in de afstandbuffer. Vanuit deze buffer wordt data uitgelezen door thread 2, die een FFT in de andere richting uitvoert (hij gebruikt alle n^e samples van een sweep). Het resultaat wordt in de dopplerbuffer geplaatst.

Na deze stap is zowel de afstands- als dopplerinformatie beschikbaar. Beide worden gebruikt voor zowel de detectie als classificatie van objecten. Deze stappen zijn echter nog niet geïmplementeerd.

Hoofdstuk 3

Resultaten

Aangezien nog niet heel het systeem geïmplementeerd is, kan het systeem in zijn geheel ook nog niet worden getest. Op het moment van schrijven is alleen de eerste stap van de FFT ver genoeg af om te testen. Wel is er een aantal tests gedaan om de prestaties van de software een eerste beoordeling te kunnen geven. Voor het testen is door de computer gebruik gemaakt van het systeem zoals gespecificeerd in tabel 3.1.

Tabel 3.1: Specificaties van de computer gebruikt voor het draaien van de testsoftware

Component	**Specificatie**
Besturingssysteem	Microsoft Windows XP SP3
Processor	Intel Xeon 3060 @ 2.4 GHz
Werkgeheugen	2 GB

Als eerste test is er gekeken naar de benodigde tijd voor het uitvoeren van de FFT's. De tijdregistratie is gedaan met behulp van de functie zoals beschreven in [20]. Hiermee is vervolgens de tijd geregistreerd die nodig is voor het uitvoeren van de eerste set van 800 FFT's. Dit duurde gemiddeld over meer dan 100 iteraties 18 ms.

Vervolgens is er nog gekeken naar de belasting op de processor. Deze is gemeten door naar de processorbelasting in Windows taakbeheer te kijken. Dit is in drie situaties gedaan. In de eerste situatie stond alleen de data-acquisitie aan en werd de data van twee kanalen naar de computer gestuurd, opgeslagen en vervolgens weggegooid. In de tweede situatie werd ook de eerste FFT-stap uitgevoerd op de data van één kanaal. In de derde situatie werd dit voor beide kanalen uitgevoerd. De bijbehorende belastingen zijn te vinden in tabel 3.2. Analyse van deze data leidt tot de conclusie dat het uitvoeren van de eerste stap FFT op één kanaal ongeveer 6 % processorbelasting oplevert.

Tabel 3.2: Processorbelasting bij de verschillende testsituaties

	Situatie	**Processorbelasting**
1	acquisitie	6 %
2	acquisitie + 1 kanaal FFT	13 %
3	acquisitie + 2 kanaal FFT	18 %

Wat betreft de resultaten van de voorbewerking valt te zeggen dat er resultaat is geboekt, maar dat dit nog niet goed gekwantificeerd is. Er is een duidelijk verschil te zien in de maximale

amplitude van het signaal met en zonder hoogdoorlaatfilter bij metingen. Echter, vanwege de snel variërende aard van de gemeten signalen, is het moeilijk om een precies resultaat te meten. Eigenlijk zou over een bepaalde tijdspanne het frequentiespectrum van het signaal met en zonder filter gemeten moeten worden. Dan kan het effect van het filter goed beoordeeld worden. Op dit moment is er alleen in het tijddomein naar het signaal gekeken en kan bevestigd worden dat het uitgangssignaal ten alle tijden tussen $\pm 2\,\mathrm{V}$ blijft.

Verder geldt natuurlijk dat er als resultaat een prototype is. Dit prototype is echter verre van af. De volgende functies zijn inmiddels geïmplementeerd:

- Filteren van de signalen van de KOR-001 RADAR-module.
- Versterken van de signalen naar een bereik van $\pm 2\,\mathrm{V}$.
- Samplen van de signalen.
- Het bufferen van de data op de computer.
- Het uitvoeren van de eerste FFT om afstandsinformatie te verkrijgen.

Mogelijk wordt de werking van het prototype nog uitgebreid nadat deze thesis is ingeleverd.

Hoofdstuk 4

Discussie

In dit hoofdstuk worden de resultaten zoals beschreven in hoofdstuk 3 bediscussieerd en enkele verbeterpunten aangedragen.

4.1 Discussie resultaten

De gemeten resultaten kunnen vooral gebruikt worden om een inschatting te maken van de prestaties van het afgeronde systeem.

Het samplen van één frame bestaande uit 480.000 samples duurt ($\frac{480000}{6\,\mathrm{MHz}}$ =)80 ms. Het doen van de eerste FFT op deze data duurt ongeveer 18 ms. Er vanuitgaande dat de tweede FFT-stap ongeveer net zo lang duurt (het totaal aantal punten is nagenoeg gelijk), nemen deze twee stappen 36 ms in beslag. Dit betekent dat er $(1000 - 80 - 36 =)$ 884 ms over is voor het detectie-algoritme. Het is in dit stadium moeilijk om dezelfde analyse voor classificatie te maken, omdat voor classificatie de data van meer dan één frame nodig is.

Extrapolatie van de processorbelasting geeft de verwachting dat beide FFT's op één kanaal een processorbelasting van 18 % oplevert. Als er wordt gekeken naar de situatie waarin alle vier de kanalen worden gebruikt, komt de verwachte processorbelasting te liggen op $6 + 4 \times 2 \times 6 =$ 54 %. Dit zou betekenen dat nog bijna de helft processorkracht beschikbaar is voor detectie en classificatie. Deze analyse neemt de beschikbare geheugenbandbreedte echter niet mee, omdat deze lastiger in te schatten is. Mogelijk is dit dan ook de bottleneck.

4.2 Verbeterpunten

Uiteraard zijn er gedurende het project keuzes en beslissingen geweest die beter of idealiter anders genomen hadden kunnen worden. Hieronder worden enkele van deze verbeterpunten besproken.

FPGA

Zoals beschreven in sectie 2.4 was er voor de verwerkingshardware idealiter voor gekozen om (voor een deel) gebruik te maken van een FPGA. Naast dat de verwerking hiermee waarschijnlijk nog wat sneller zou zijn geweest [18], zou een FPGA vooral de grootte van het prototype ten goede komen. Op dit moment is de behuizing van het prototype ongeveer 18 cm bij 15 cm bij 13 cm. Terwijl de gebruikte computerkast, met daarin ook de ADC en verwerkingshardware ongeveer 44 cm bij 43 cm bij 16 cm is. Wanneer de verwerkingshardware dus op een FPGA was

gerealiseerd, die geïntegreerd kan worden in de behuizing van het prototype, zou het systeem een stuk kleiner worden.

Filter voorbewerking

Het ontvangen signaalvermogen bij de RADAR neemt voor afnemende frequenties toe met 40 dB per decade (zie vergelijking 2.4). Om de invloed van lage frequenties in amplitude te beperken, is een hoogdoorlaatfilter gebruikt. Dit is echter een eerste orde butterworth filter met een vermogensafname van 20 dB per decade. Om de vermogenstoename bij de lage frequenties volledig te compenseren, had dus beter een tweede orde filter gebruikt kunnen worden. Een dergelijk filter neemt namelijk met 40 dB per decade af, wat voor een vlakke overdracht bij de lage frequenties zou zorgen. Dit is echter te laat bedacht om nog in het ontwerp te kunnen worden opgenomen.

Oversamplen

Oversamplen is het samplen op een hogere frequentie dan strikt noodzakelijk is volgens het Nyquist-Shannon theorema. Dit zorgt in principe voor een betere signaal-ruisverhouding [14] [21]. De gebruikte ADC is ook in staat om met twee keer de nyquist-frequentie (12 MHz) te samplen. Er werd echter te laat gedacht aan oversamplen om dit nog in het ontwerp te verwerken.

Als dit wel wordt geïmplementeerd, betekent dit dat er een stap bij komt in de software. Hier moet namelijk een extra laagdoorlaatfilter bij (voor het uitvoeren van de FFT's) met een kantelfrequentie van 3 MHz. Dit filter zorgt dat de kwantisatieruis die in de band van 3 MHz tot 6 MHz wordt weggehaald.

Licentie FFTW

Voor het uitvoeren van de FFT's in de software wordt gebruik gemaakt van FFTW. Bij eventuele in productie name van het systeem zou de GPL-licentie van deze bibliotheek echter een probleem kunnen opleveren. Deze licentie vereist namelijk dat de broncode van software die gebruik maakt van FFTW openbaar gemaakt wordt voordat deze mag worden verspreid. In dat geval kan er echter prima overgestapt worden naar een (commercieel) alternatief, zoals Intel IPP.

Hoofdstuk 5

Conclusie

In dit hoofdstuk wordt teruggekeken op het resultaat uit hoofdstuk 3 en de doelen die vooraf zijn gesteld, zoals te vinden in sectie 1.5, aan het prototype. In het algemeen kan gezegd worden dat niet alle doelen zijn gehaald in de beperkte tijd van het project en dus is helaas het prototype niet zover gevorderd als vooraf gewenst.

5.1 Beoogd gebruik

Allereerst wordt gereflecteerd op de eisen vanuit beoogd gebruik, zoals genoemd in sectie 1.5.1.

Aan de in sectie 1.5 gestelde eis dat er digitale uitvoer moet zijn die aangeeft of er een object is gedetecteerd (eis 1.5.1.1) is niet voldaan. Het oorspronkelijke idee was om de algoritmiek die uit een ander deelonderzoek [2] zou komen te implementeren in software (C++), om dit te realiseren. Doordat het gereed maken van de algoritmiek voor de uiteindelijk gebruikte RADAR-module (KOR-001) langer duurde dan verwacht, was er geen tijd meer over om deze vervolgens te implementeren.

De eis dat het systeem moet kunnen aangeven of het gedetecteerde object een mens is of niet (eis 1.5.1.2) is niet gerealiseerd. Zoals hierboven gezegd is het niet gelukt om de detectie te implementeren, daardoor was het ook niet mogelijk om de daaropvolgende classificatie te realiseren.

Aan de eis dat het systeem een afstandsbereik tussen de 10 m en 1500 m heeft (eis 1.5.1.3), wordt waarschijnlijk voldaan. Hoewel er geen meting kon worden gedaan over een recht stuk van minimaal 1500 m, dus het is nu niet te zeggen of het systeem ook daadwerkelijk nog signaal kan meten op dezet afstand. Zoals gezegd in hoofdstuk 3 bleek bij metingen wel dat het signaal van dichtbijzijnde objecten wordt weggefilterd door het hoogdoorlaatfilter van de voorbewerking (sectie 2.2), ook al is er nog niet gekwantificeerd of dit daadwerkelijk voor objecten dichterbij dan 10 m is. Daarnaast bleek een signaal-ruisverhouding van 87 dB nodig (sectie 2.3) en dat volgens de datasheet de gebruikte data-acquisitiekaart (*Innovative P25M*) deze signaal-ruisverhouding ook haalt [10].

5.2 Syteemontwerp

In sectie 1.5.2 waren er eisen gesteld aan het ontwerp van het systeem zelf. Hieronder zal er worden gereflecteerd op deze eisen.

Aan de eis dat het systeem gebruik maakt van de signalen zoals zij uit de RFBeam KOR-001 RADAR-module komen (eis 1.5.2.1) is voldaan. Alle ontwerpkeuzes zijn gebaseerd op een prototype opgebouwd rondom de RFBeam KOR-001 RADAR-module. En bij het maken van het systeem is dan ook volledig gebruik gemaakt van de signalen zoals die uit de RFBeam KOR-001 komen.

Hoewel de detectie niet is geïmplementeerd en dus niet geconcludeerd kan worden dat de eis om objecten in minder dan 1 seconde te detecteren (eis 1.5.2.2) is gehaald. Blijkt uit de resultaten en de discussie daarvan, hoofdstuk 3 en sectie 4.1, wel dat met het huidige prototype de eis waarschijnlijk met gemak gehaald kan worden. Er is immers nog aanzienlijk wat rekenkracht beschikbaar voor de detectie (ongeveer 46% van de processorkracht) en ook tijd, zoals vermeld ongeveer 884 ms, oftewel ruim 88% van de tijd die in totaal beschikbaar was gesteld in de eis.

Zoals ook is vermeld in sectie 4.1, is het lastig om te zeggen of personen binnen 5 seconden gedetecteerd kunnen worden (eis 1.5.2.3). Nu is dat in ieder geval nog niet het geval, omdat de classificatie nog niet is geïmplementeerd, zoals vermeld in sectie 4.1. Aangezien niet met zekerheid gezegd kan worden of eis 1.5.2.2 gehaald kan worden en er nog niks van de classificatie is geïmplementeerd, kan niet geconcludeerd worden of eis eis 1.5.2.3 haalbaar is met het huidige prototype.

Zoals hierboven en bij de discussie (hoofdstuk 4) al is genoemd is er niks van het algoritme voor detectie of classificatie geïmplementeerd. Aan eis 1.5.2.4 is dus niet voldaan.

Aan de eis dat het systeem een beperkt formaat moet hebben, zodat het systeem langs het spoor geplaatst kan worden (eis 1.5.2.5) is met de huidige verwerkingshardware niet voldaan. Door de noodzaak van een (externe) computerkast voor de verwerking met de huidige implementatie, past het prototype niet in een compacte behuizing voor langs het spoor.

5.3 Aanbevelingen

Zoals gezegd is dit onderzoek nog niet af, dat blijkt ook wel uit het feit dat veel van de aan het prototype gestelde eisen nog niet behaald zijn. De belangrijkste vervolgstap is dan ook om een algoritme voor detectie en classificatie te implementeren. Als dat is gebeurd kan er beter getest worden of het prototype doet wat het zou moeten doen.

Verder zou, zoals genoemd in hoofdstuk 4, het goed zijn om de verwerking op een FPGA te implementeren. Naast dat dit mogelijk betere prestaties oplevert komt tot vooral ten goede aan grote en ook het stroomverbruik. Door de compactere implementatie die wordt gerealiseerd met een FPGA is het waarschijnlijk mogelijk om heel het prototype in de behuizing te integreren, in plaats van dat er nog een externe computerkast (die ook vele malen groter is dan de behuizing zelf) aangesloten moet worden. Eventueel zou er zelfs gebruik kunnen gemaakt van een tussenoplossing, waarbij een deel van de verwerking op een FPGA wordt uitgevoerd en een deel op een normale processor.

Tot slot zou, zoals gezegd in sectie 2.2.4, de voorbewerking beter op een PCB geïmplementeerd kunnen worden. Dit voorkomt verstoringen op het signaal en zorgt voor een compacter ontwerp dat makkelijker geïntegreerd kan worden in de behuizing van het prototype.

Bibliografie

[1] *ProRail jaarverslag.* Prorail, 2012.

[2] R. Bootsman and P. Lagerweij, "Fmcw radar algoritme ontwikkeling voor de detectie en classificatie van mensen gebruik makend van het micro doppler-effect."

[3] R. Eg, "Videobewaking tegen spoorwandelaars," *Metro*, p. 2, 2014.

[4] Rijksoverheid. Risico's van koperdiefstal. [Online]. Available: http://www.rijksoverheid.nl/onderwerpen/koperdiefstal/risico-s-van-koperdiefstal

[5] S. Gosseling and R. Wilmer, "Aansturing van fmcw-radar voor de detectie en classificatie van mensen."

[6] Ridenour, "Radar system engineering," *MIT Radiation Lab series*, vol. 1, p. 629, 1947.

[7] A. Wojtkiewicz, J. Misiurewicz, M. Naiecz, K. Jedrzejewski, and K. Kulpa, "Two-dimensional signal processing in fmcw radars," 2007.

[8] *KOR-001 3D RADAR transceiver. Customer specific datasheet*, RFBeam Microwave GmbH, version 1.1.

[9] S. Gürbüz, W. Melvin, and D. Williams, "Detection and identification of human targets in radar data," *Proc. SPIE*, vol. 6567, 2007.

[10] *P25M datasheet*, Innovative Integration.

[11] *AD828 datasheet*, Analog Devices, 2002, rev. c.

[12] A. Devices, "Ad828 spice model." [Online]. Available: http://www.analog.com/en/high-speed-op-amps/high-voltage-amplifiers/ad828/products/mod-spice-models/resources.html

[13] W. Kester, "Which adc architecture is right for your application?" *Analog Dialogue*, vol. 39, no. 06, 2005.

[14] M. W. Hauser, "Principles of oversampling A/D conversion," *J. Audio Eng. Soc.*, vol. 39, no. 1/2, januari/februari 1991.

[15] P. Brady, "Connection speeds compared," september 2006. [Online]. Available: http://www.pixelbeat.org/speeds.html

[16] Ranjith. (2009, oktober) Introduction to PCI protocol. [Online]. Available: http://electrofriends.com/articles/computer-science/protocol/introduction-to-pci-protocol/

[17] *P25M User's Manual*, Innovative Integration, 2390-A Ward Ave Simi Valley, California 93065, februari 2009.

[18] *Radar Processing: FPGAs or GPUs?*, Altera, version 2.0.

[19] FFT benchmark results. [Online]. Available: http://www.fftw.org/speed/

[20] (2012, april) C/c++ tip: How to measure elapsed real time for benchmarking. [Online]. Available: http://nadeausoftware.com/articles/2012/04/c_c_tip_how_measure_elapsed_real_time_benchmarking

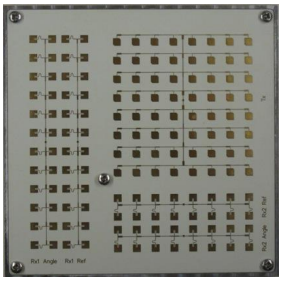
[21] J. G. Proakis and D. G. Manolakis, *Digital signal processing.* Pearson, 2007.

Bijlage A

Datasheet RFBeam KOR-001

Features

- This is not an RFbeam Standard product
- 24 GHz K-band Superhet Transceiver
- Highest Sensitivity Receiver
- Measure of Speed, Distance and Angle
- Monopulse, 1xTX-Antenna, 4xRX-Antenna
- PLL Controlled Frequency Generation
- I/Q IF Doppler output with 3MHz Bandwidth
- Integrated FSK and FMCW mode
- Configuration over Serial Interface
- 20dBm EIRP output power



Applications

- Traffic monitoring
- Industrial Applications
- Security systems

Description

KOR-001 is a Doppler transceiver for measuring speed, distance and angle of an object. The superhet structure gives best sensitivity for enhanced measurement distance. An easy interface allows the use of FSK and FMCW modulation. Modulation bandwidth can be programmed over a serial line. All frequencies are PLL controlled to ensure a low temperature drift. High bandwidth I/Q outputs allow use of high FSK switching frequencies.

Blockdiagram

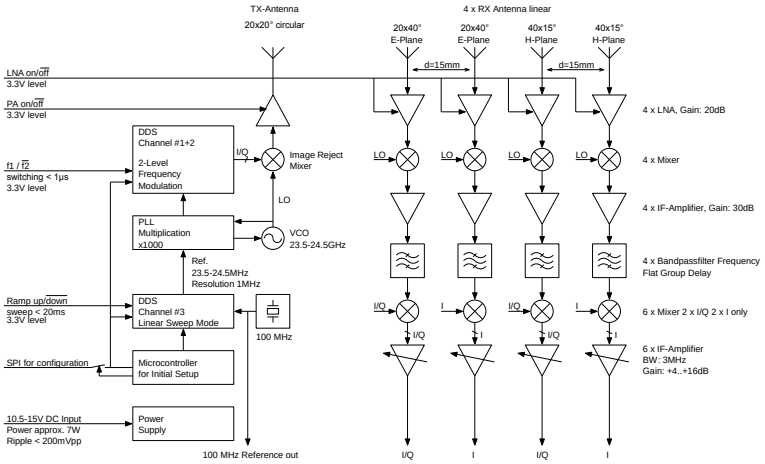


Fig. 1: Block diagram

Characteristics

Parameter	Min	Typ	Max	Unit	Testing Method
Operating Conditions					
Supply Voltage	10.5	12.0	15.0	V	
Voltage Ripple			200	mV_{pp}	
Supply Current		600		mA	Input Voltage = 12V
Logic ‚1' Level	2.4			V	(3.3V) f1/f2, Ramp up/down, PA/LNA on/off, SPI
Logic '0' Level			0.8	V	(3.3V) f1/f2, Ramp up/down, PA/LNA on/off, SPI
Operating Temperature	-20		+65	°C	
Storage Temperature	-40		+85	°C	
Transmitter					
LO Frequency	23.500		24.500	GHz	Full specification only for 24.000 – 24.250GHz
Transmitter Frequency	23.540		24.540	GHz	TX frequency = LO frequency + 40MHz
Frequency Step Width		1		kHz	
FMCW Sweep Time	20			ms	Sweeping from 24.000 – 24.250GHz
FSK Switching Time			1	µs	Step from 24.100GHz to 24.110GHz
PLL Settling Time			20	µs	Step from 24.100GHz to 24.110GHz
Output Power		+17	+20	dBm	EIRP power, 24.000 – 24.250GHz
Output Power Deviation		+/- 1		dB	Frequency sweeping from 24.000 – 24.250GHz
Frequency Drift vs. Temperature			+/- 50	ppm	-20°C - +65°C
Maximum Frequency Error			100	ppm	Absolute deviation of 24.100GHz Carrier
Phase Noise		-85		dBc	@100kHz, 24.125GHz
Spurious		-30		dBc	24.000 – 24.250GHz, not LO or other Sideband
Out of Band Spurious		-40		dBc	Non-harmonic or Subharmonic frequencies
LO Suppression		15		dBc	
Suppression of other Sideband		20		dBc	
TX-Antenna					
Antenna Gain		19		dBi	
Polarisation		V			Linear, vertical
Horizontal -10dB Beamwidth	18	20	22	°	
Vertical -10dB Beamwidth	18	20	22	°	
Sidelobe Suppression		25		dB	
RX-Antenna 'RX1'					
Antenna Gain		14		dBi	
Polarisation		H			
Horizontal -10dB Beamwidth	13	15	17	°	
Vertical -10dB Beamwidth			40	°	
Sidelobe Suppression		25		dB	

RX-Antenna 'RX2'					
Antenna Gain		14		dBi	
Polarisation		V			
Horizontal -10dB Beamwidth			40	°	
Vertical -10dB Beamwidth	18	20	22	°	
Sidelobe Suppression		25		dB	
Receiver					
LNA Gain		18		dB	
LNA Noisefigure		4.0		dB	
Mixer Conversion Loss		-10		dB	
IF Phase Error		15		°	1MHz FSK Switching at IF Center Frequency
IF Amplifier Gain		32		dB	
IF Bandwidth		10		MHz	Passband: 30 – 40MHz
Receiver Sensitivity		-141		dBm	f=500Hz, B=1kHz, S/N=6dB
Overall Sensitivity		158		dBc	f=500Hz, B=1kHz, S/N=6dB
Doppler Output					
Frequency Range	3		3M	Hz	-3dB
IF Buffer Gain	4	10	16	dB	Not user adjustable
Noise Floor		-96		dBV/Hz	RX-Antenna covered, B=10kHz
DC Offset	2.0	2.5	3.0	V	RX-Antenna covered
I/Q Amplitude Balance		+/- 1		dB	Will be factory adjusted for all receivers
I/Q Phase Shift	80	90	100	°	
Phase Offset RX1			10	°	Max. error for object approaching orthogonally
Phase Offset RX2			10	°	Max. error for object approaching orthogonally
Body					
Outline Dimensions		110x110x10		mm	
Weight		250		g	

Antenna System Diagram

The diagrams are separated for Transmitter and bot Receiving channels. To find overall sensitivity both diagrams need to be considered.

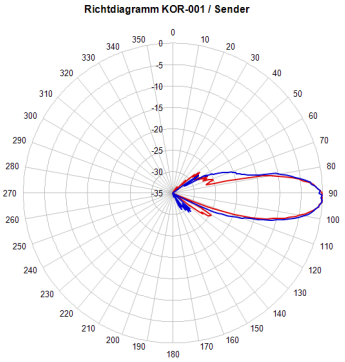


Fig. 2: Antenna diagram 'Transmitter'

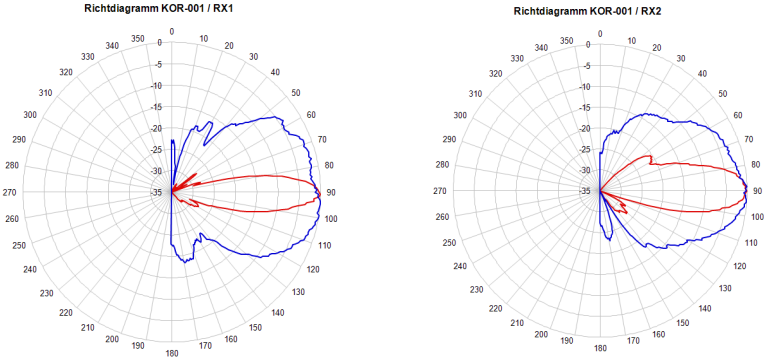


Fig. 3: Antenna diagram 'RX1' (left) and 'RX2' (right)

Pin Configuration

Pin	Name	Description
1	GND	ground
2	GND	ground
3	RX1, Ref I-Channel	load >=1kOhm
4	RX1, Ref Q-Channel	load >=1kOhm
5	RX1, Angle I-Channel	load >=1kOhm
6	RX2, Ref I-Channel	load >=1kOhm
7	RX2, Ref Q-Channel	load >=1kOhm
8	RX2, Angle I-Channel	load >=1kOhm
9	GND	ground
10	GND	ground

Fig. 4: Connector 1 (Analog)

Pin	Name	Description
1	+12V Input	Power Supply +
2	GND In	Power Supply -
3	GND In	Power Supply -
4	DDS MOSI	3.3V SPI to AD9959 DDS
5	DDS MISO	3.3V SPI to AD9959 DDS
6	DDS SCLK	3.3V SPI to AD9959 DDS
7	DDS nCS	3.3V SPI to AD9959 DDS
8	DDS I/O_Update	3.3V SPI to AD9959 DDS
9	DDS Switch f1/f2	Switch FSK Frequency
10	DDS Ramp up/down	FMCW Ramp control
11	DDS P3	not used
12	PA on/off	0V = TX PA on
13	CPU UART RX	3.3V Serial Configuration input
14	CPU UART TX	3.3V Serial Configuration output
15	GND	Signal Ground
16	Oscillator Switch	0V = internal oscillator
17	Ext. Oscillator Input	External Reference Input (100MHz)
18	GND	Signal Ground
19	GND	Signal Ground
20	Oscillator Output	DDS Sync Output (125MHz)

Fig. 5: Connector 2 (Power / Control)

Outline Dimensions

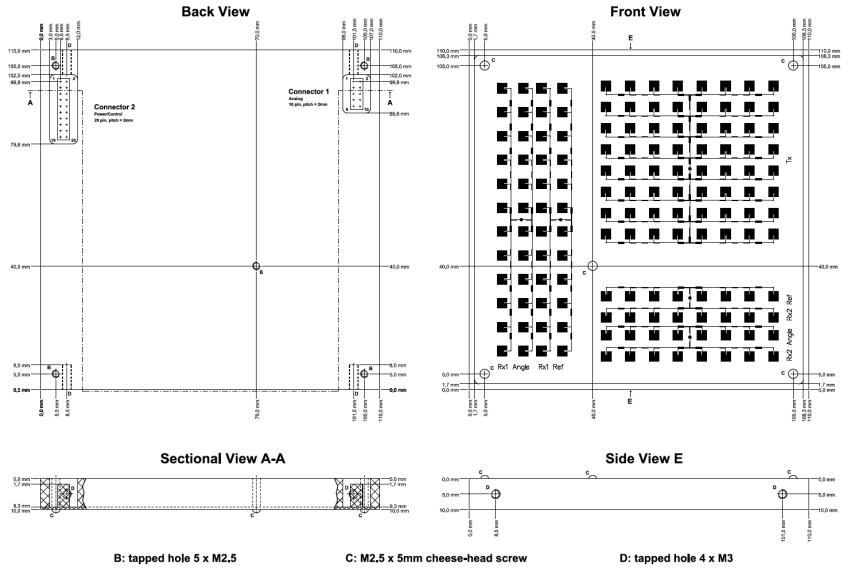


Fig. 6: Mechanical data

Datasheet Revision History

Version	Date	Changes
1.0	22-Nov-2013	initial release
1.1	12-Dec-2013	minor changes in text

Bijlage B

Datasheet Innovative P25M

P25M

DSP/FPGA PCI Card with Analog I/O

300MHz C6713 DSP (Floating Point)
User-configurable Spartan-3 FPGA
Four 25MSPS • 16-bit A/D
Four 50MSPS • 16-bit D/A

Features

- PCI 64-bit/66MHz
- >85 db SFDR analog IO
- Xilinx 1M gate Spartan3 FPGA
- 65 bits digital IO
- DSP Memory: 128MB DRAM
- FPGA Memory: 2MB SRAM
- Extensive software support in source form
- Custom logic development supported for FPGA
- DSP/BIOS peripheral drivers

Applications

- PCI based real-time control
- High-end data capture & playback
- Industrial high-speed controls
- Stimulus- response measurements
- OEM instruments

Hardware Options

- DSP debugger
- I/O cables

Software Development Tools

- C/C++ Dev Sys using TI Code Composer Studio
- Real-time OS: TI DSP/BIOS w/drivers for DSP periph
- MS Visual C++ Host support
- Turnkey applications with source code
- C++ components for analysis, graphing, logging
- MATLAB and VHDL FrameWork Logic Dev Tools

XILINX®
The MathWorks

Ordering Information

P25M	80169-0

RoHS 2002/95/EC Compliant

P25M

Overview

P25M is a powerful and flexible DSP + FPGA for signal processing and real-time control in PCI systems. The P25M PCI card features a 300 MHz floating point DSP and 1M gate FPGA with 4 channels of 25MSPS, 16-bit A/D and 4 channels of 50MSPS, 16-bit D/A. Supporting this open-architecture hardware platform, the Pismo software tool set provides target and host libraries, practical utilities and numerous example programs that illustrate the use of all board peripherals. The FrameWork Logic development tools provide comprehensive support for adding signal processing to the FPGA. The full-featured logic and DSP programming tools greatly accelerate all facets of the system development.

DSP Core

The P25M is built around the powerful, C-friendly floating point Texas Instruments TMS320C6713® DSP. The '6713 has 256KB of on-chip memory coupled with an efficient cache controller. The CPU clock is 300MHz while the 32-bit External Memory Interface (EMIF) bus is clocked at 75MHz.

The P25M main memory is 128MBytes of synchronous SDRAM clocked at 75MHz. The enhanced DMA controller handles 16 independent channels, which assists the CPU with data movement and preserves its bandwidth for application-specific code. Both McBSP sync serial ports are pinned out to connectors.

The DSP is programmed using Texas Instruments DSP/BIOS, a high performance DSP RTOS. The software libraries provided with the card include DSP/BIOS peripheral drivers for all peripherals. The toolset includes turn-key data acquisition and playback application with source code that illustrates the best utilization of all hardware resources. In particular, communicating with the host is simple with a powerful packet-based PCI transfer supporting both chained busmastering for high-speed data transfer as well as mailbox message for command and controls.

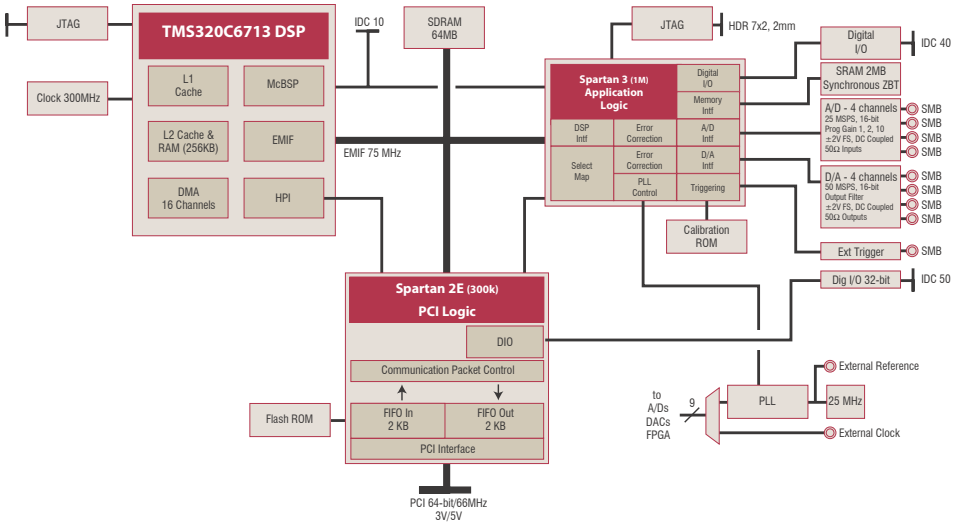
FPGA Core

The FPGA logic is reconfigurable on the P25M and may be customized using the FrameWork Logic development tools. The Xilinx XC3S1000 FPGA featured on the P25M provides 1M system gates, multipliers and embedded memory blocks. The P25M exposes over 33 of these I/O pins to end user on the digital I/O. The FGPA has 2MB SRAM memory, useful for computational or buffer memory.

The FrameWork Logic for the P25M supports VHDL and MATLAB development tools for IP development. The logic is configured from the PCI host through select MAP. The standard logic consumes about 20% of the logic. Further details of the logic development process are in the FrameWork Logic User Guide, online at Innovative's website.

Analog I/O

The P25M integrates 4 A/D and 4 D/A channels with the FPGA core. The channels are independent, simultaneously sampling with >90dB dynamic range - ideal for stimulus response measurements and servo applications. The A/Ds and D/As have a direct connection to the FPGA for high-speed DSP applications in the FPGA.



The A/Ds and D/As may be clocked externally or from a low jitter PLL. Triggering features in the FPGA include framed capture, external or software triggers.

All analog I/O is 100% calibrated on each card. Digital error correction in the logic is provided and uses an on-card non-volatile memory for calibration coefficients

System-Level Peripherals For Fast, Simple Integration

Digital IO Ports - Port 1: 32 bits implemented as memory-mapped, 32-bit latch for general-purpose LVTTL digital I/O, with ESD and overvoltage protection. Direction is software-configurable in bytes. The port may be software or externally clocked at rates to 40 MHz. Port 2: 33 bits direct to FPGA.

Accurate PLL Time Base - An AD9510 PLL and clock distribution device provides a low-jitter, stable clock that may be used as a sample clock. The PLL outputs have a range of 4.765 MHz to 50 MHz with a 150 MHz /N (N=1..32) resolution.

External Clock and External Interrupt input are available on SMB co-ax connectors with 50-ohm termination. Ext INT is used to trigger the execution of a particular processing routine on an external hardware event, while external clock signal is typically used to slave on-card analog conversion to an external time base.

Software Support

The P25M Pismo tool kit is a powerful collection of software libraries, utilities, examples projects and interactive help file that allow developers to be very productive from the start. Numerous program examples - with source code- demonstrate the usage every peripheral on the board.

Target Side Tools - Extensive C/C++ libraries, example source code, DSP/BIOS peripheral drivers, mailbox messaging and bulk data transfer to/from PC are fully integrated in Texas Instruments' Code Composer Studio. Pismo makes extensive use of DSP/BIOS, TI's DSP operating system included in CCStudio, which simplifies code development of complex, multi-thread applications by providing drivers, scheduling services and run-time management of all DSP resources to optimize bandwidth. Users do not need to understand hardware-specific low-level code to set up and control IO peripherals.

Host Side Tools – Pismo includes two groups of Host Side support tools: extensive C++ libraries -with sophisticated program examples- and turnkey utilities. Libraries include methods to control the P25M (Reset DSP, Download code) and to communicate with the board via mailbox-style messages and bulk data transfers. Turnkey utilities - provided as executables - provide logic and DSP loading and assist wth program debugging at the beginning of a project. A simplified set-up for RTDX terminal in-

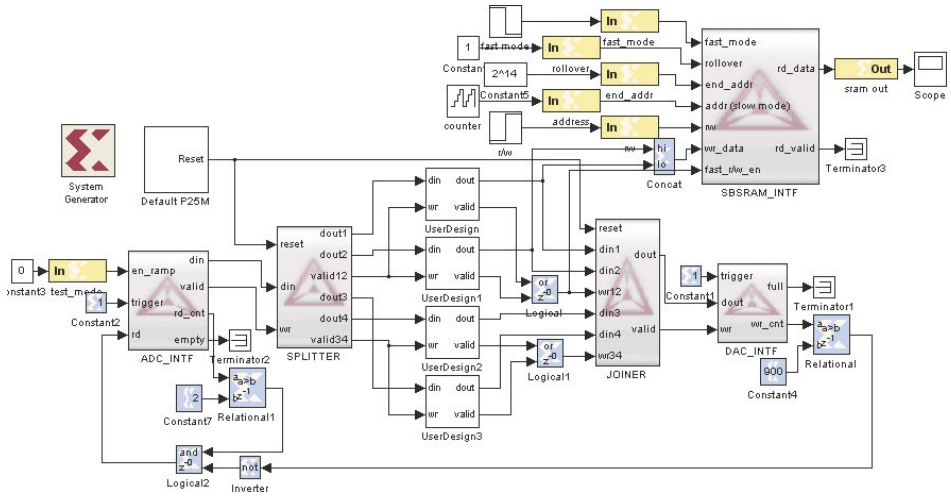
terfaces provided for Code Composer Studio, and practical examples showing messages and data transfer between DSPs and CCStudio. Additional provided tools for data viewing (BinView) and target communications (UniTerminal) can be used with Code Composer Studio Debugger to aid in DSP application development.

FPGA Logic Development

Logic development using VHDL and MATLAB is supported. The FrameWork Logic User's Guide fully describes the VHDL logic, which includes peripheral interfaces for all analog I/O and the DSP.

The VHDL FrameWork Logic is used as a hardware interface layer that can be modified to include new IP cores and add unique features such as signal processing and triggering. The logic components for each hardware interface provide a simple data stream interface that is used to add new functionality to the design either in VHDL, IP cores from MATLAB, or third party IP cores. Complete source code, constraints and Xilinx ISE project files are provided.

The board support package for MATLAB SimuLink and Xilinx System Generator provides a comprehensive development and simulation toolset for DSP development and testing.



Block sets for the P25M include A/D, DAC, DSP and memory interfaces that provide full hardware access to MATLAB users.

The P25M board support library is integrated with MATLAB and Xilinx System Generator providing full access to the P25M peripherals and DSP. Gateways between MATLAB and the hardware allow hardware-in-the-loop testing for IP development and verification. Examples illustrate the use of the analog IO, memory and DSP interface from within the MATLAB environment. When the development in MATLAB is complete, the logic is easily integrated with the FrameWork Logic for fully embedded operation by including the IP core into the VHDL project and compiling the system.

OEM Configurations

The P25M board can be configured or modified to fit your specific requirements and provide an optimal mix of performance, cost and features. Contact Innovative Integration with your specific OEM requirements.

Digital Signal Processor

300MHz Texas Instruments TMS320C6713 DSP
32-bit floating-point
L1 cache 4KB data/4KB program
L2 cache + Unified RAM 256 KB
Two multichannel buffered serial ports (McBSP)
Two 32-bit timers
Sixteen DMA channels
32-bit external memory interface, clocked at 75MHz
Two McBSP serial ports on external headers

Memory

128 MB SDRAM clocked at 75MHz
16 kbit serial EEPROM (used for calibration)

User-reconfigurable FPGA

Xilinx Spartan3 XC3S1000-4FG456C (1,000K gates, -4 speed grade is standard)
Supports all I/O functions
IO pins to FPGA : 33 DIO connections
SelectMAP loading from PCI bus
MATLAB Simulink Board Support Package
Framework Logic provides standard functionality
VHDL source code with ModelSim test benches
JTAG port for ChipScope and JTAG downloads

Clocks and Triggering

PLL with post-scaling for Low jitter: <1ps RMS
4.68MHz to 50 MHz
Dual External SMBs (optional 50 ohm) for external clocks, triggers or interrupts input/output
External, software and framed trigger modes

Digital IO ports

Port 1

32 bidirectional bits, direction selected on a byte basis
external clock input
LVTTL, direct from FPGA (0-3.3V signaling)
5V tolerant

Port 2

32 bidirectional bits, direction selected on a byte basis
external clock input
LVTTL, (0-3.3V signaling)
5V tolerant
ESD and overrange protection

A/D Channels

Channels	4 Independent
Sample Rate	1 to 25 MSPS
Resolution	16-bit
Input	±2V, DC Coupled
Input Impedance	50ohm
Program. Gain	1, 2, 10
Analog In Bandwidth	-3dB @ 12 MHz
A/D Devices	LTC2203

DC Specifications

Offset Error	Calibrated to <1mV
Gain Error	Calibrated to <0.05%fs
Linearty Error	<1.5 LSB

AC Specifications

SFDR	>88 dB, 100kHz, 5 MSPS
ENOB	>11.7 bits, 100kHz, 5 MSPS
S/N	>90 dB, 100kHz, 5 MSPS

D/A Channels

Channels	4 Independent
Sample Rate	0 to 50 MSPS
Resolution	16-bit
Input	±2V, DC Coupled
Input Impedance	50ohm
Output Bandwidth	-3dB @ 12MHz
DAC Devices	LTC1668

DC Specifications

Offset Error	Calibrated to <1mV
Gain Error	Calibrated to <0.05%fs
Linearty Error	<±4 LSB typ

AC Specifications

SFDR	84 dB, 14kHz, 25 MSPS update
ENOB	11.7, 14kHz, 25 MSPS update
S/N	82 dB, 14kHz, 25 MSPS update
Glitch Impulse	<1mV peak, <5ns

PCI Interface

64-bit, 66 MHz, 3V or 5V
Down-selects to 32 bit and 33MHz if needed
Packet mode interface:
- Efficient packet transmission mechanism
- Supports high rate continuous PCI data transfers
  - 80 MB/s on many 33 MHz, 32-bit platforms
  - 350 MB/s on many 66 MHz, 64-bit platforms
- DMA interface to DSP with auto-pacing
- Interrupts to DSP on packet transactions

HPI interface to DSP for code loading and communications
Field-programmable, in-situ logic updating using supplied applet

Debug Ports

JTAG 1149.1 compliant emulation port for DSP
Compatible with Innovative Code Hammer, TI XDS-510/560 compliant debugger
JTAG for FPGA (compatible with Xilinx cables such as Parallel Cable IV and platform USB)

Connectors

SMB: Ext Clocks (2), A/D inputs (4), D/A outputs (4)
IDC-50 polarized male, Digital IO
IDC-14 polarized male, DSP JTAG
IDC-40 polarized male, FPGA Digital I/O
IDC-10 polarized male, FPGA JTAG
2x5 header, McBSP's

Physical Data

1/2 Size PCI (short card)
64-bit 3V/5V connector
Weight: 160 grams

Power

4 W typical (varies with DSP & FPGA application)
3.3V, 5V supply from PCI bus

Operating Conditions

0-70 degrees C (noncondensing)
Some configurations may require forced air
Designed to meet ETS 300 019-1.1 Class 1.2, ETS 300 019-1.2 Class 2.3, ETS 300 019-1.3 Class 3.3

Development Languages

DSP: C++ under Code Composer Studio, MATLAB/Simulink for modelling and code generation
Pismo Toolset includes libraries, projects, utilities, help files
Host PC: MS Visual C++, .NET, Borland C++ Builder,
Logic : VHDL with Xilinx ISE, Mentor Graphics ModelSim, MATLAB and Xilinx System Generator

Accessories

SMB to BNC cable, 1M 67021

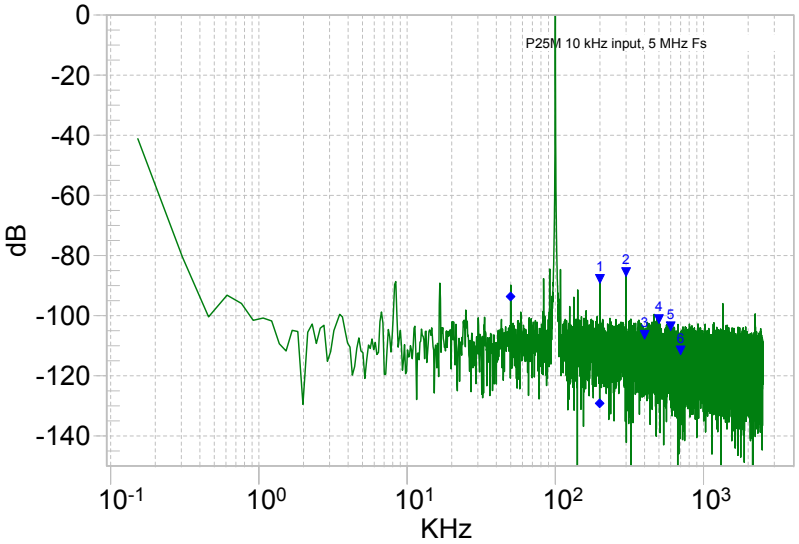
Documentation for P25M

P25M Hardware user Guide
FrameWork Logic User Guide
Malibu Software Development Manual

Environmental Data

Lead Free
ROHS Compliant

Frequency Response



Max S/N (dB)	S/N (dB)	SINAD (dB)	ENOB	SFDR (dB)	THD (%/dB)
98.1	74.9	72.4	11.7	88.7	0.006051 (84.4dB)

DAC Output Spectrum