Onderwerp: Data

Leerdoelen

Aan het einde van dit onderwerp ben je in staat om:

- Pulsvorming toe te passen op een datastroom in de vorm van een RRC filter
- Grove frequentiesynchronisatie toe te passen in de vorm van een Frequency Locked Loop
- Fijne frequentiesynchronisatie toe te passen in de vorm van een Costas Loop
- Tijdssynchronisatie toe te passen met behulp van het *Symbol Sync* blok
- Stabiele smalbandige datacommunicatie op te zetten met behulp van de Pluto's

Voorbereiding op de les

- Lees de hoofdstukken van PySDR zoals vermeld in de planning op de wiki
- Bekijk de introductie video's op Brightspace.

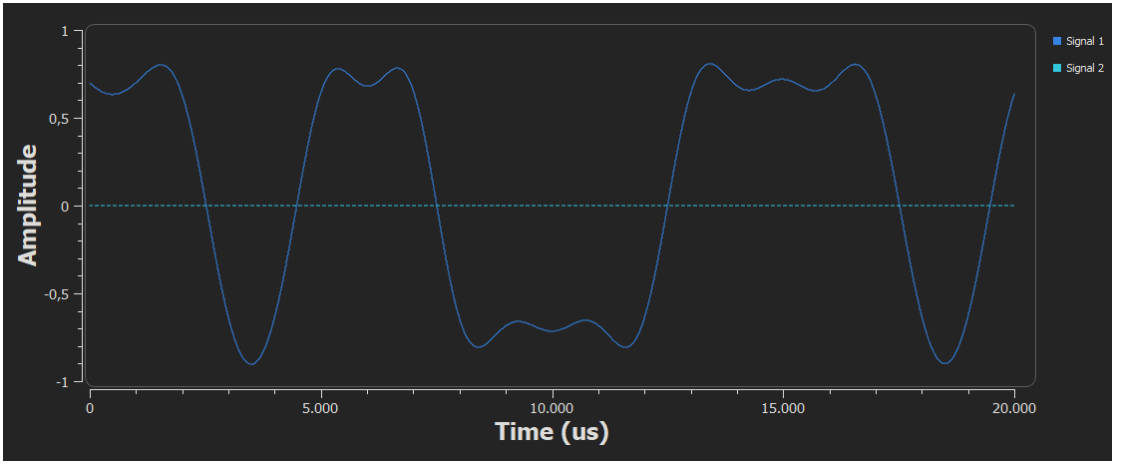
Beoordeling

Maak als groep een screencast waarin jullie de uitwerking van elke opdracht van dit onderwerp onderbouwen. Zorg ervoor dat beide personen aan woord komen en dat je aan de volgende punten aandacht besteedt:

1. Leg uit wat je in python hebt gedaan
2. Onderbouw voor de GNU Radio Companion schema's met name de instellingen van de blokken die je hebt gebruikt. Waarom heb je deze instellingen gekozen? Waarom deze volgorde van blokken?
3. Laat zien dat het werkt, dit mag een korte opname met je telefoon zijn waarin in duidelijk is dat jouw schema worden gebruikt.

De screencast wordt beoordeeld volgens de rubric en beschrijving die op brightspace is geplaatst. Hou de deadlines van de cursushandleiding aan.

Opdrachten 27 pt



Figuur 1: Datasignaal wat is gefilterd door een RRC filter in het tijddomein.

Opdracht 1: BPSK uitbreiding [4 pt]

Begin als basis met je schema's van de PSK opdracht uit Lab RDS. Zorg ervoor dat het ook pakketten gebruikt zoals we bij DSSS hadden gedaan. Tip 1: Het is handig om in beide schakelingen een variabele te maken `symbol_rate`, en een aantal andere variabelen waarvan je de waarde berekent op basis van de symbol en sample rate. Zoals bijvoorbeeld samples per symbool. Dit zijn dingen waar de blokjes vaak om vragen en het is handig om deze in één keer goed te hebben staan. Tip 2: Stel de draaggolffrequentie van de zender en ontvanger niet exact op de zelfde waarde in, maar bijvoorbeeld 50 kHz ernaast. Zie DC Piek op PySDR voor uitleg waarom dit handig is.

Pas de volgende dingen toe, zoals uitgelegd in de video's:

- Gebruik een *FLL Band-Edge* blok bij de ontvanger om een grove frequentiesynchronisatie toe te passen voordat het signaal naar een RRC filter gaat.
- Gebruik een *FFT Root Raised Cosine Filter* filter[a] bij de zender en ontvanger. Koppel een variabele of range aan α zodat je het in kunt stellen van 0 tot 1. Zet de standaard waarde op 0.35 voor een mooi signaal maar iets bredere band. Bekijk de uitgang in het tijddomein. Het zou een gefilterde en vertraagde versie

moeten zijn van jouw datasignaal. Verhoog eventueel het aantal taps en verlaag de gain om het binnen ±1 te houden. Zie figuur 1.

- Gebruik een *Costas Loop* blok bij de ontvanger om fijne frequentiesynchronisatie toe te passen. Zet er eventueel een *AGC* of *AGC3* blok ervoor om het vermogen van het signaal te regelen naar 1 voordat het in de costasloop gaat.

Nu het signaal gefilterd aankomt, zullen we de juiste samples eruit moeten pakken. Test eerst of je schema werkt met *Keep 1 in N* en een *Delay* ervoor om zelf de juiste timing te vinden. Waar is het bereik van deze delay van afhankelijk?

[a] In de video werd de non-fft versie getoond, maar de FFT-versie zou wat sneller moeten werken

Opdracht 2: Tijdsynchronisatie [4 pt]

Nu we een grove en fijne frequentiesynchronisatie hebben, kunnen we ook automatische tijdssynchronisatie proberen toe te passen bij de ontvanger. Gebruik hiervoor het *Symbol Sync* blok om de delay en keep 1-in-N te vervangen. N.B. In de video stelde ik dat dit blok al een matched filter heeft; Dit klopt alleen als je voor de `Interpolating Resampler` de `Polyphase Filterbank, MF` gebruikt, anders zul je jouw RRC moeten laten staan. Stel de timing error detector in op `Gardner` en de interpolator in op `MMSE 8 tap FIR`. Koppel een variabele of range aan de gain van de timing error detector zodat je deze kunt instellen. Kijk of het blok goed werkt en je stabiele communicatie hebt.

Om tijd te besparen zijn wat tips gevraagd aan AI voor het instellen van het Symbol Sync blok.

Opdracht 3: QPSK [5 pt]

Kijk of je jouw schema kunt uitbreiden naar QPSK. Wanneer het werkt, bepaal dan de SNR achter jouw RRC filter bij de ontvanger. Hiervoor kun je het vermogen meten als de zender aanstaat (signaal+ruis) en wanneer de zender uitstaat (ruis). Hier moet dan natuurlijk geen AGC voor staan, want die zou proberen het vermogen met of zonder signaal naar 1 te regelen. Op basis van de SNR, wat is het maximaal aantal niveaus wat je kunt gebruiken met jouw setup (gain en afstand en locatie van de

pluto's)? Zorg ervoor dat 16 niveaus theoretisch mogelijk zijn in jouw situatie voor de volgende opdracht.

Opdracht 4: QAM **[8 pt]**

Voor de volle punten de moeilijkste vorm. Kijk of je jouw schema kunt zelf kunt uitbreiden naar QAM-16 zonder extra instructies. Hiervoor zul je wel de aanpak moeten aanpassen omdat je nu de costasloop niet meer kunt toe passen en je waarschijnlijk een andere aanpak nodig hebt voor de tijdssynchronisatie. Dit is de aanpak die AI voorstelt. Wegens andere opzet van het vak is dit nooit eerder een opdracht geweest maar het moet mogelijk zijn! De docent heeft op moment van schrijven het zelf nog niet uitgevoerd, dus het is helemaal aan jullie om dit voor elkaar te krijgen.

Opdracht 5: Maximale data-rate **[6 pt]**

Wat is de maximale data-rate die je kunt behalen bij een sample-rate van 1 MHz op een afstand van 1 labtafel? Bereken de SNR en bepaal de theoretische limiet. Je zult dus moeten experimenteren met verschillende modulatieschema's en instellingen van de blokken om te kijken wat het beste werkt. Mocht QAM-16 niet gelukt zijn, probeer dan bijvoorbeeld 8-PSK.

Woordenlijst

GRC GNU Radio Companion. 1