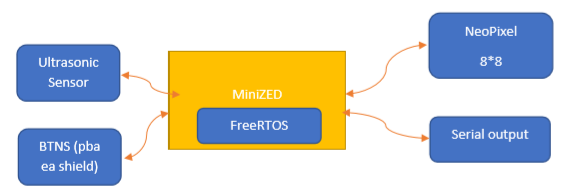
Final Exercise Part 1

# Overzicht:

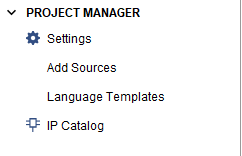
In dit project gaan we een Minized FPGA aansturen. De bedoeling is dat we als ingangen knoppen en een ultrasoon sensor gebruiken om zo een spelletje te programmeren op een Neonpixel (matrix 8\*8). En deze waarde ook serieel kunnen uitprinten op een seriële uitgang. Het programeren van deze software gebeurt in vivado en in SDK.



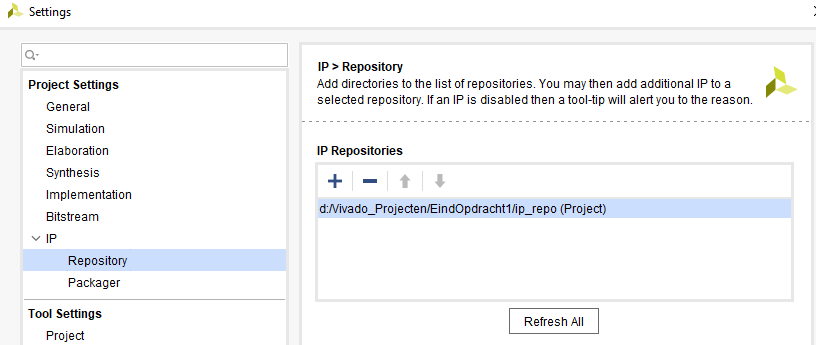
# Vivado:

Hier hebben we een project aangemaakt waar we de vhdl code geschreven hebben om de ultrasoon sensor en de neonpixel aan te sturen. Vervolgens hebben we van deze code aparte IP-blokken gemaakt die we vervolgens in een blokschema kunnen gebruiken om ze daarna te kunnen aansturen in SDK.

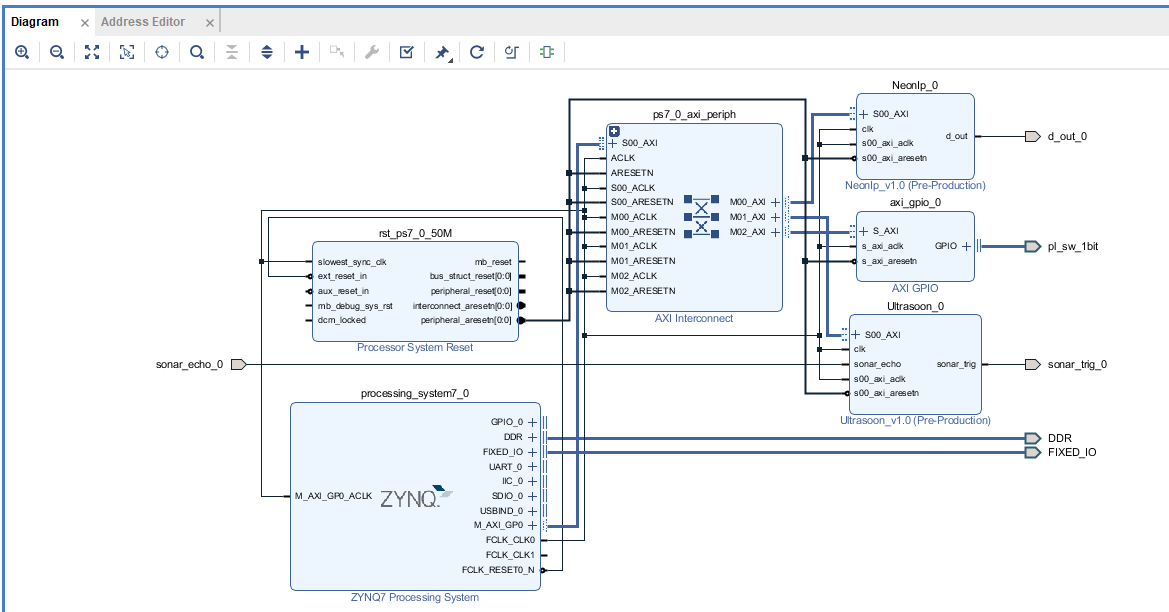
1. Open het programma dat het blokschema bevat om zo te controleren of alle blokken er in zitten.
   * Als ze er niet allemaal inzitten moet je ze toevoegen op de volgende manier.
     + Ga naar settings



* + - Klik op IP -> vervolgens op Respository
    - Klik op de **+** om de IP-blokken toe te voegen.

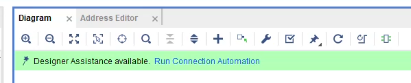


* + - Dan kan je de map selecteren waar je al je IP-blokken gemaakt hebt om ze daarna toe te voegen in je blokdesign.
    - In je blokdesign klik je op de **+** knop om alle nodige blokken toe te voegen.

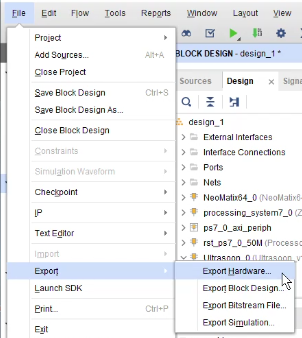


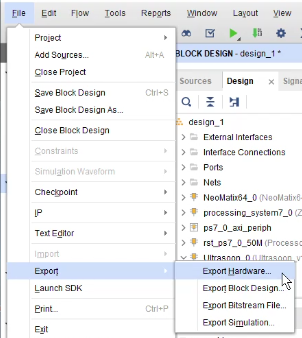
De blokken die nodig zijn: Zynq blok, al je eigen blokken en een axi gpio blok.

* + - Klik op run connection om alles met elkaar te verbinden



* + - Maak de nodige pinnen extern die nog niet verbonden zijn.

1. Als je ze allemaal hebt kan je een wrapper van je blokschema maken.
   * Rechtse muisknop op je blokschema design -> create hdl wrapper
2. Daarna maak je een bitstream van het programma.
   * 
   * Klik op File -> exporteer hardware :Vervolgens exporteer je de hardware samen met de bitstream naar SDK.

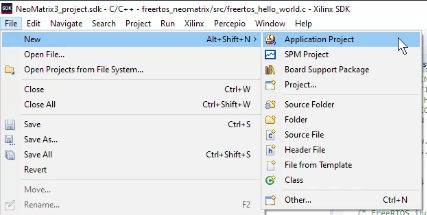


* + Klik op File -> launch SDK: SDK gaat automatisch open.

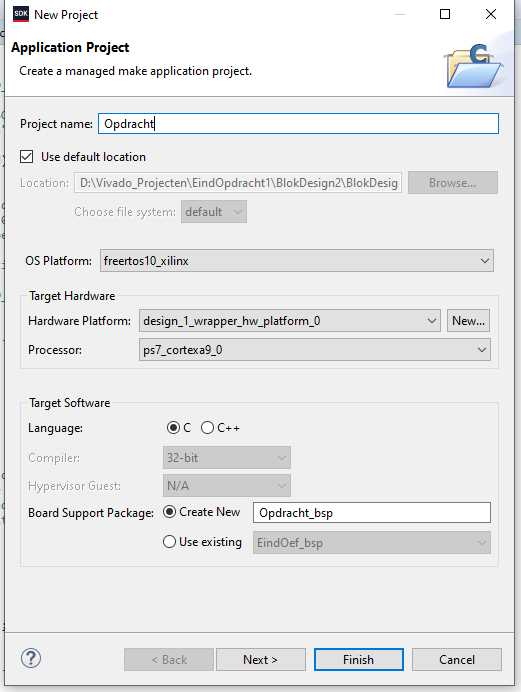
# SDK:

Hier is er een applicatie aangemaakt die in C-taal geschreven word, en die werkt op Xilinx-FreeRTOS zodat we kunnen werken met verschillende soorten queues, timers, etc. .

1. Maak een applicatie aan:
   * Klik op File -> New -> Application



* + Geef de applicatie een naam.
  + Zet het OS platform Freertos10\_xilinx



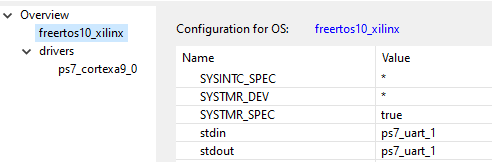
1. Ga naar de bsp file met dezelfde naam als uw project.



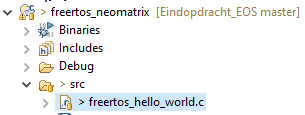
* + Klik hier met de rechtse muisknop en selecteer



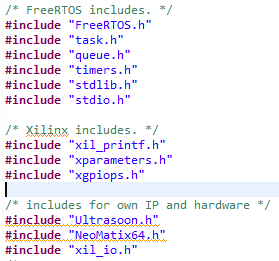
* + Dan ga je naar Overview -> freertos10\_xilinx en zet je de **stdin** en de **stdout** op ps7\_uart\_1.



1. Dan ga je naar de applicatie -> en klik je op de hello\_world.c file (dit is je hoofdprogramma)

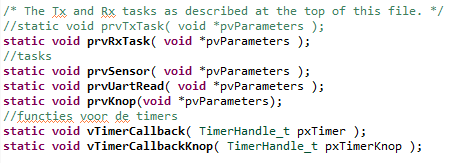


1. Importeer al de nodige libraries, ook die van de eigen aangemaakte IP-blokken.



1. Maak functies voor de sensor, Uart communicatie en de neonpixel.

Deze dienen om alle data te versturen en te ontvagen.

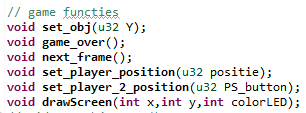


1. Functies voor de timers die we gaan gebruiken.

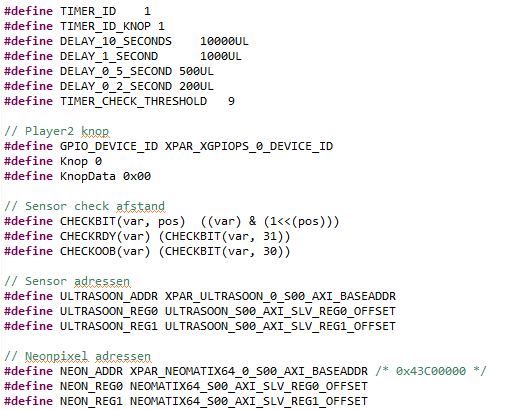
Worden gebruikt om de tijd te bepalen zodat het spel niet te snel of te traag is.



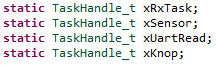
1. Ook zijn er functies voor de game zelf, deze zijn nodig om de game te spelen.



1. Maken verschillende define variabele aan, zodat we gemakkelijker kunnen werken in onze code, en zodat we maar een variabele moeten verander in plaats van diezelfde op meerdere plaatsen.



1. Definiëren de tasks voor de functies. Type waarnaar een taak verwezen wordt. (via een pointer-parameter)Nemen een variabele TaskHandle\_t die vervolgens kan worden gebruikt als parameter voor vTaskDelete om de taak te verwijderen.

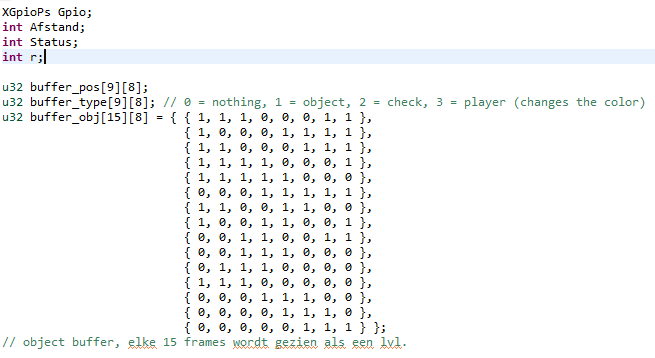


1. Definiëren de queue die gaan gebruiken om de data op te verzenden.



1. Definiëren de timers die we gebruiken om de delay’s in te stellen, zodat er niets te snel of te traag is.



1. Definiëren extra variabelen die we nodig hebben om het programma te runnen, deze staan globaal zodat we ze overal in ons programma kunnen gebruiken.
   * De GPIO staat voor "Algemene invoer / uitvoer". Het is een type pin op een geïntegreerde schakeling die geen specifieke functie heeft. De functie van een GPIO-pin is dat hij aanpasbaar is en bestuurd kan worden door software.
2. 

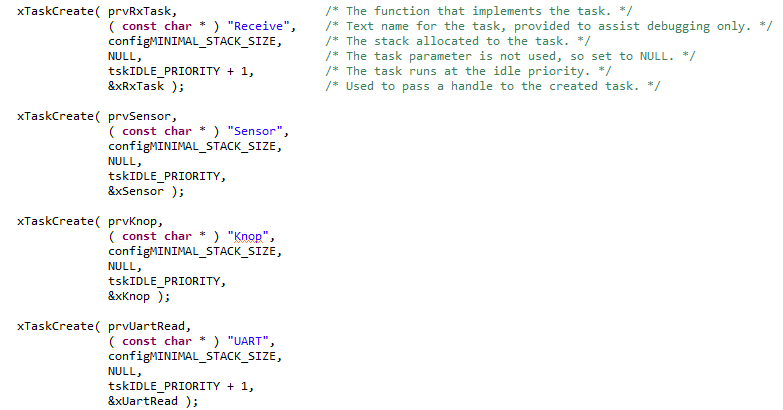
## Main() functie:

Hier schrijven we de belangrijkste onderdelen van onze code die we niet in een functie steken om ons programma te laten runnen zoals de tasks, timers en queue’s.

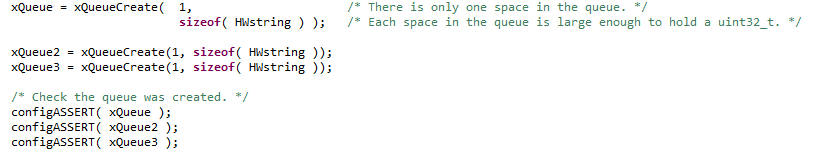
1. Maken constante timers die niet veranderer omdat er een vaste time delay nodig is voor bepaalde zaken in ons programma.



1. Maken de tasks aan die we koppelen aan de functies, die de codes bevatten om ons spel te kunnen spelen.



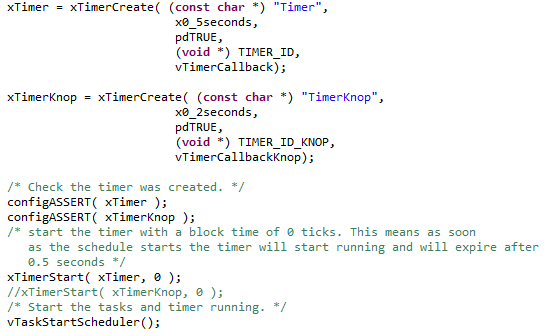
1. Maken de verschilende queues die nodig zijn voor de communicatie tussen de verschillende functies en de data die in elke functie verzonden word.



Controleren

* 1. xQueue: Is de verbinding tussen de ultrasoon sensor (speler1) en de neon pixel.
  2. xQueue2: Is de verbinding tussen de neon pixel die de kleur waarde verstuurd naar de UART.
  3. xQueue3: Is de verbinding tussen de knop( speler 2) en de neon pixel.

1. Definiëren we de timers die we gaan gebruiken om de delay in ons spel te maken zodat alles goed verloopt.



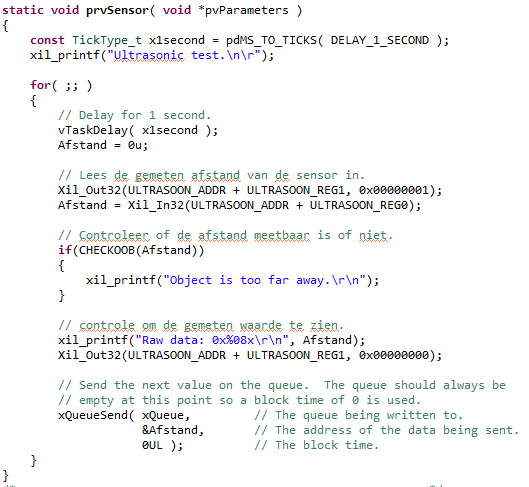
Welke timer doat wat.

* 1. xTimer:
  2. xTimerKnop:

## Ultrasoon sensor:

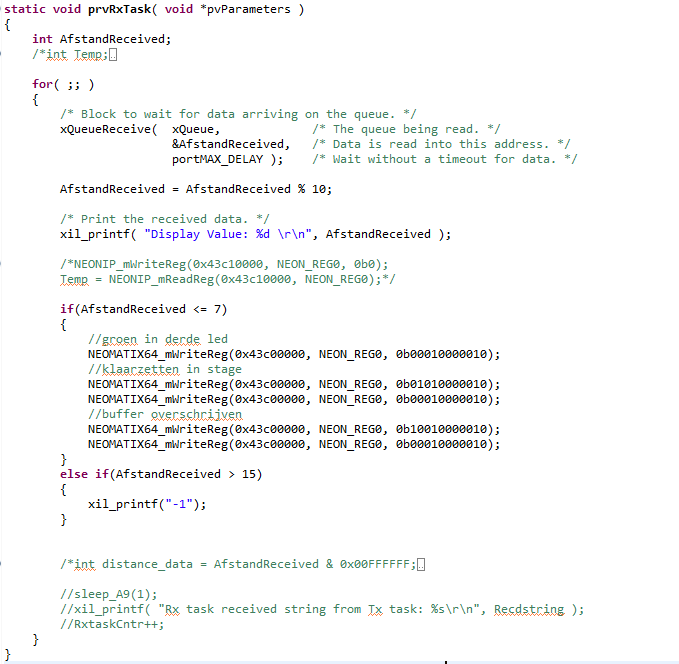
In deze functie staat alle code om de ultrasoon sensor in het programma te kunnen gebruiken.

1. De functie van de sensor
   1. Hier lezen we afstand in van de sensor door het basis adres en het slave register samen in te lezen en deze toe te kennen aan een variable samen met weergave van de waarde.
   2. Dan kijken we of de afstand meetbaar is of niet (licht de afstand binnen de range van de sensor).
   3. Vervolgens steken we het adres van de afstand in de queue om zo de waarde van de afstand te verzenden naar de ontvanger om zo een uitgang aan te sturen (neonpixel).



## NeonPixel:

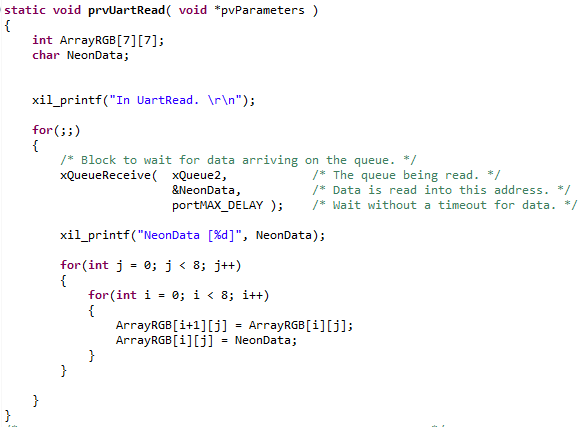
1. We lezen de afstand uit die via de queue ontvangen is.
2. We maken verschillende IF-state om zo te bepalen wat er in ons spelletje gaat gebeuren (dit gaat voor onze beweging van een speler gebruikt worden).
3. Via de mWriteReg functie van de neonpixel kunnen we zelf bepalen welke leds we gaan aansturen en welke kleur ze krijgen.



## UART Read:

Hier lezen we de RGB waardes van de neonpixel uit, die we vervolgens op dezelfde plaats als op de neonmatrix in een array gaan steken om ze te kunnen lezen in een terminal.

1. We halen de data uit een queue, en steken hem in een variabel.
2. Printen de RGB variabel uit om te controleren welke waarde er is meegegeven.
3. Vervolgens steken we deze waarde in een array die er uitziet zoals de neonmatrix.
4. Het resultaat van de array zou nu moeten overeenkomen met die van de neonmatrix.

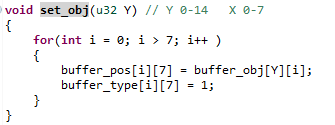


# Game:

Hier staan alle functies die betrekking hebben op het spelen van de game en de visualisatie daarvan.

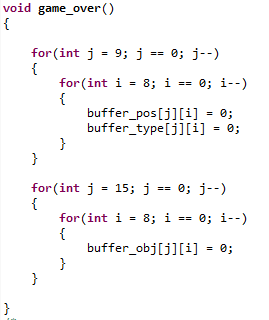
## set\_obj functie:

uitleg



## game\_over functie:

Hier word bepaald wanneer een speler de game verloren heeft.



## Next frame functie:

In deze functie gaan we kleuren van de neonmatrix updaten naar de nieuwste kleuren.

1. We maken de grootte van de neonpixel na, en zetten daarna de kleuren hierin, om ze weer te geven.
2. We maken ook een variable aan waar we de laatst gekende kleur insteken om die vervolgens in een queue te steken, die de waarde verstuurd naar de UART terminal.
3. De kleuren worden ook nog weggeschreven naar de juiste plaats op de neonpixel zelf met de mWriteReg functie van onze NeonMatix library.

