Testplan

Ik heb mijn applicatie en mijn library getest op fouten. Ik heb geprobeerd alle mogelijke waarden mee te geven met mijn applicatie. In dit testplan laat ik weten wat er faalde en wat niet. Ik beschrijf per functie/classe welke tests er zijn gedaan. Catch is op heel veel plekken niet mogelijk omdat ik geen return krijg van mijn ledstrip. De ledstrip heeft alleen maar een input.

# Arduino

Elke keer voordat ik een sensor wilde aansluiten aan de arduino heb ik getest of de port het wel deed met een led + weerstand. Door due::pin\_out te gebruiken en de correcte pin aan te zetten controleerde ik of de pin niet dood was. Zo kon ik met 100% zekerheid een sensor aansluiten.

Verder heb ik voor de allereerste flash een OOPC-example geüpload. Puur om te testen of het flashsen werkt en dat het ledje van de arduino knipperde.

# Applicatie

## Int getafstand(due::pin\_out pinout, due::pin\_in pinin)

Deze functie heb ik zelfgeschreven omdat ik moeite had om een andere library te importen. Arduino.h was niet aanwezig. Om deze functie te testen heb ik 4 verschillende Afstandsensoren (HC-SR04) aangesloten. Door deze functie in een for loopje te laten lopen + hwlib::cout te gebruiken kon ik erachter komen of de waardes veranderde.

Waar kwam ik achter?: Het resultaat was altijd 0. Terwijl de Logic analyser wel pulsen aangaf. Ik zag in de grafiek dat tussen de aanpuls van de trigger pin en de echo pin een korte tijd niks aanwas. Ik heb ik de code een loop geschreven waar hij inblijt waneer hij 0 is. En pas gaat meten wanneer echo net op 1 springt.

Het volgende probleem was dat ik heel erg lang moet wachten als de afstand super ver is. Daarom heb ik een limiet ingesteld.

## Void showcolor(bool bgreen, bool bred, bool byellow, bool bblue, ledstripport ledstrippie)

Deze functie heb ik getest door de verschillende boolean variabele aan te passen. Wanneer meerdere variabele true zijn moeten er meerdere kleuren branden. Deze functie werkte helemaal goed in 1 keer

## bool checkcolor(int duration, int color, ledstripport ledstrippie, int endwait = 50)

Deze functie heb ik getest door elke kleur te testen. Ik heb de duration heel hoog gezet waardoor er heel veel tijd is om deze functie te testen. Ik heb getest of de afstand sensor op de goede plek staat en dat de correcte kleur aanging toen ik met mijn hand voor de sensor bewoog. Langzamerhand heb ik de distance steeds veranderd naar een getal waar het programma het beste op werkt. De functie returnt true wanneer de sensor iets ziet binnen de distance EN dat dat binnen de tijd gebeurd. Wanneer 1 van de twee niet waar is, returnt de functie false.

## bool checkmorecolor(int duration, bool green, bool red, bool yellow, bool blue, ledstripport ledstrippie)

Deze functie is een uitgebreide versie van de vorige. Deze functie wordt aangeroepen als er 2 of meer kleuren (sensoren) gecheckt moeten worden. Deze functie werkt ook gewoon met 1 kleur. Deze functie moet true returnen wanneer de sensoren iets zien onder de distance EN dat dat binnen de tijd gebeurd. Dit heb ik getest door alle mogelijke combinaties te runnen. Als alle 4 de bools true zijn moeten alle sensoren iets zien voordat de tijd voorbij is.

## Void normalsimon(ledstripport ledstrippin)

Dit is het programma voor normal simon. Dit is een hoofdfunctie die van alle andere functies afhangt. Dat betekent dat ik deze als laatste heb getest. Alles werkte eigenlijk best goed. Ik heb hier en daar hwlib::wait\_ms() toegevoegd om “acties die je niet kan volgen” te voorkomen. Verder werkte deze functie al heel snel. Er blijft een nieuwe kleur komen totdat er een fout wordt gemaakt of dat de max van de array is bereikt.

## Bool randbool()

Returnt een willekeurige bool. Getest met een cout. Dit werkte prima. Wel werd ik er aan herinnert dat arduino due geen tijd heeft en dus niet compleet random kan zijn met srand(time(0)); Dus moet ik aan het begin van het programma een loopje maken die elke loop wat aftrekt van een integer totdat er op een knop gedrukt wordt. Of een sensor wordt geactiveerd.

## Void levels(ledstripport ledstrippie)

Dit is het “leukere” spel. Dit spel werkt met levels (de level array’s bovenaan). Er kunnen makkelijk nieuwe arrays worden toegevoegd. Ik had wat moeite met het loopen door de array’s. Bij level 3 gaf het programma aan dat er maar 3 integers waren, terwijl er toch echt 5 stonden. Na heel veel testen met cout en logisch verstand, heb ik uiteindelijk gezien dat er een onnodige loop in het programma zat. Hierdoor loopte het programma verticaal ipv horizontaal.

Verder heb ik hier ook getest wanneer je allemaal dood kan gaan. De timing werkte goed, de kleuren gingen correct aan en ik moest correct reageren. Ook werd er een animatie gegeven toen ik af was en toen ik een level haalde. Dat betekent dat de rest van de functie helemaal naar toebehoren werkt.

## Int main(void)

Hier worden de ledstrips aangemaakt en wordt er door verschillende modes geloopt. Met blauw kon je een mode spelen. Groen -> normaal simon, Rood -> 2 speler levels, Geel -> stop programma (Hier was eerst 2 spelers maar hier ben ik helaas niet aan toe gekomen. De functies kunnen wel 2 spelers aan, alleen hebben levels nog niet de mogelijkheid om 4 kleuren op te slaan in de colorarray).

# Library

## static void \_\_attribute\_\_((noinline)) wait\_busy( int32\_t n )

Deze functie is van Wouter van Ooijen. Toch heb ik deze functie getest met de logic analyser in combinatie met de class ledstrip -> functie write(). Omdat het aanroepen tijd kost begint de tijd al met bijna 250 nanoseconden. Met deze wait\_busy functie kan dit een heel klein beetje verhoogt worden zodat de goede tijden kunnen worden gebruikt voor het aansturen van de ledstrip.

## Write(bool x)

Omdat er veel classes zijn met dezelfde functies ga ik dit beschrijven per functie.

Ik focuste met hier heel erg op in de eerste paar dagen. Ik moest verschillende 1nen en 0len sturen. Ik startte eerst met 24 1nen sturen. Een 1 is 900 nanoseconden hoog 350 laag. Ik vulde een simpele hwlib::wait\_ns() met deze waarden. De ledstrip sprong gelijk op wit.

Helaas was dit totaal niet het geval toen ik een 0 wilde schrijven. 350 hoog, 900 laag. Ook hier sprong de ledstrip op wit. Ik deed de tijd lager en lager van de wait(350) maar geen geluk. Zelfs met een hwlib::wait\_ns(0) sprong de ledstrip op wit.

Ik wilde wel kijken hoelang de signalen duurde. Dus pakte ik mijn Logic analyser en kwam er al snel achter dat de wait functie aanroepen best veel tijd koste. Ongeveer 400 nanoseconden. Dus wilde ik de tijd weten zonder wait functie ertussen. Puur een write(1) en write(0) achterelkaar in een for loopje. Hierna kwam ik op het grootste probleem van dit project. Het duurt 550 nanoseconde om de pin van 1 naar 0 te zetten en andersom.

Met hulp van docenten ben ik erachter gekomen dat het zo lang kan duren door een referentie. Helaas moet hwlib::target wel een referentie zijn want we weten niet welke classe de gegeven target is. Due::pin\_out was de oplossing. Deze heeft hoeft geen referentie omdat due pinnen gewoon due pinnen zijn en niets anders. Met behulp van de Logic analyser en de wait\_busy functie is het gelukt om nullen en enen te schrijven naar de ledstrip.

## void blink(led\_pin, int ms, int blinks)

Deze functie werkte meteen. Deze heb ik getest door hem uit te voeren. Ook met ws2812\_port werkte de blink goed, ook in combinatie met ws2812\_invert.

## Void blink(ledstrip\_pin, int ms, int blinks)

Deze functie lijkt heel erg op de vorige functie, alleen werkte deze met ledstrips. Deze is ook getest door hem uit te voeren met verschillende ledstrip\_pinnen zoals ledstrip\_invert en ledstrip\_port.

## Changegroupled(int beginled, int endled, struct ws2812 & rgbcolor)

Deze functie heb ik getest door eerst alle ledstripjes uit te laten. De contructor van de ledstrip\_array kreeg zwart en zwart mee. Daarna deed ik een changegroupled met verschillende kleuren en begin en eindledjes. Het was een kwestie van tellen of de functie goed werkte. Deze functie begon oorspronkelijk bij 0 maar later heb ik ervaren dat beginnen met 1 beter is al gaat het om kleur veranderen van de ledstrip.

Deze functie wilde ik ook laten werken met ledstrip\_port. Toen moest ik het voor elkaar zien te krijgen, ik kwam er achter dat ik dan het aantal ledjes moest weten van de ledstrip\_pins. Maar omdat ledstrip\_port dat niet kan opvragen moest ik functies maken die deze dingen returnen:

## Int Countleds()

Returnt het totaal aantal leds. Een hwlib::cout was voldoende om dit te testen.