**Software Design Document**

**Project Infinity Dice**

**Avans Hogeschool**

Reference: Software Design Document R01.docx

Reference date: 2020-05-28

Author(s): terbei, triomm

Status: [~~Concept~~] / [Accepted]

Versiebeheer

|  |  |  |
| --- | --- | --- |
| Rel. | Datum | Wijzigingen |
| R01 | 2020-05-20 | Definitieve versie |
|  |  |  |

Distributie

|  |  |  |  |
| --- | --- | --- | --- |
| Naam | R01 | R02 | R03 |
| Remo de Jong | x |  |  |
| Richard Jansen | x |  |  |
| Terence Beijloos | x |  |  |
| Tristan van Ommeren | x |  |  |
| Jasper van den Heuvel | x |  |  |
| Joan Schrasser | x |  |  |

Inhoud

[1. Introductie 5](#_Toc40897267)

[1.1. Achtergrond 5](#_Toc40897268)

[1.2. Afkortingen 5](#_Toc40897269)

[2. De Android-applicatie 6](#_Toc40897270)

[2.1. Een schets 6](#_Toc40897271)

[2.2. Activities, Fragments en Services 6](#_Toc40897272)

[3. Bluetooth verbinding 7](#_Toc40897273)

[3.1. Hoe kan een Androidapplicatie verbonden worden met de SoC? 7](#_Toc40897274)

[3.2. Hoe wordt de verbinding met hetzelfde apparaat hersteld als deze verbroken is? 7](#_Toc40897275)

[3.3. Hoe kan ervoor gezorgd worden dat er maar één en hetzelfde apparaat per aan tijd verbonden kan worden? 8](#_Toc40897276)

[3.4. Hoe wordt een in run-time aanpasbaar bericht gestuurd naar de SoC? 8](#_Toc40897277)

[3.5. Hoe ziet de data die naar de SoC wordt gestuurd eruit? 8](#_Toc40897278)

[3.6. Hoe wordt een in run-time aanpasbaar bericht gestuurd naar de Androidapplicatie? 9](#_Toc40897279)

[4. Accelerometer 10](#_Toc40897280)

[5. Gyroscoop 11](#_Toc40897281)

[6. LED 12](#_Toc40897282)

[7. Magic effect 13](#_Toc40897283)

[7.1. Natuurkundig probleem 13](#_Toc40897284)

[7.2. Machine learning 13](#_Toc40897285)

[7.2.1. Verzamelen van data (simulatie) 13](#_Toc40897286)

[7.2.2. Mogelijke modellen 13](#_Toc40897287)

[7.2.3. Deep neuraal netwerk 14](#_Toc40897288)

[7.2.4. Integreren in de SoC 14](#_Toc40897289)

[8. Draadloos programmeren 15](#_Toc40897290)

[Appendix A. Afbeeldingen en maken van service 16](#_Toc40897291)

Lijst van figuren

[Figuur 1 app layout 6](#_Toc40897292)

[Figuur 2 BLE event handler 9](#_Toc40897293)

[Figuur 3 Versturen Bluetooth bericht 9](#_Toc40897294)

[Figuur 4 If-statement die het triggeren tijdens opstarten voorkomt 10](#_Toc40897295)

[Figuur 5 Activity diagram voor het updaten van de leds 12](#_Toc40897296)

[Figuur 6 OTA draadloos software updaten 16](#_Toc40897297)

[Figuur 7 Linear regression 17](#_Toc40897298)

[Figuur 8 kmeans 17](#_Toc40897299)

[Figuur 9 kNN 18](#_Toc40897300)

[Figuur 10 Service enum 18](#_Toc40897301)

[Figuur 11 CuSTS1 database 18](#_Toc40897302)

[Figuur 12 toevoegen service 19](#_Toc40897303)

Referenties

|  |  |  |
| --- | --- | --- |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | Linear Regression using Python  Animesh Agarwal  2018 Oct 5th  <https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2> |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | Running K-means  The VLFeat Authors  2007  <https://www.vlfeat.org/overview/kmeans.html> |
|  |  |  |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | kNN Algorithm with Data-Driven k Value  Debo Cheng, Shichao Zhang, Zhenyun Deng, Yonghua Zhu  10.1007/978-3-319-14717-8\_39, 2014-12  <https://www.researchgate.net/publication/293487460_kNN> |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | But what is a Neural Network? | Deep learning, chapter 1  3Blue1Brown  2017 okt 5  <https://www.youtube.com/watch?v=aircAruvnKk> |
|  |  |  |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | OTA Update Process (Bluetooth Low Energy)  Amazon alexa  2010  <https://developer.amazon.com/en-US/docs/alexa/alexa-gadgets-toolkit/bluetooth-le-ota-update-process.html> |
|  |  |  |
| [] | Title:  Author (Company):  ID, Version, Date:  File: | Intel HEX  Zie References  2020 mei 5  <https://en.wikipedia.org/wiki/Intel_HEX> |

# Introductie

## Achtergrond

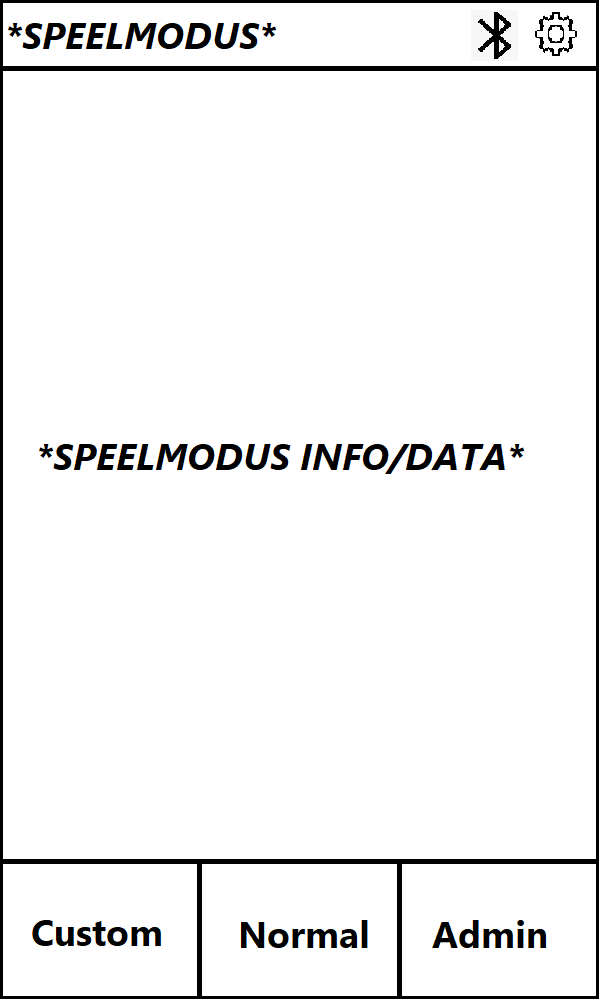
Dit document is geschreven in opdracht van Avans Hogeschool voor het voorlichtingsproject onder leiding van Jasper van den Heuvel.

## Afkortingen

|  |  |
| --- | --- |
| SoC | System on Chip |
| BLE | Bluetooth Low Energy |
| kNN | K nearest neighbours |
| kmeans | Machine learning algorithme |
| Mx.x | Verwijzing naar een eis in het PvE |

# De Android-applicatie

## Een schets

In Figuur 1 app layout zie je een schets van de Android-applicatie, zoals deze eruit moet komen te zien. De onderste vakken dienen als knoppen en kunnen gebruikt worden om van speelmodus te wisselen. In het grootste vak staat informatie, zoals de sliders bij “Custom mode” om de percentages aan te passen. Bovenaan staat de titel van de speelmodus, waarin de gebruiker zich bevindt. Tot slot zijn er rechtsboven twee aparte knoppen voor Bluetooth, zodat je via een apart scherm kunt verbinden met de SoC, en instellingen, zodat algemene instellingen aangepast kunnen worden.

Figuur 1 app layout

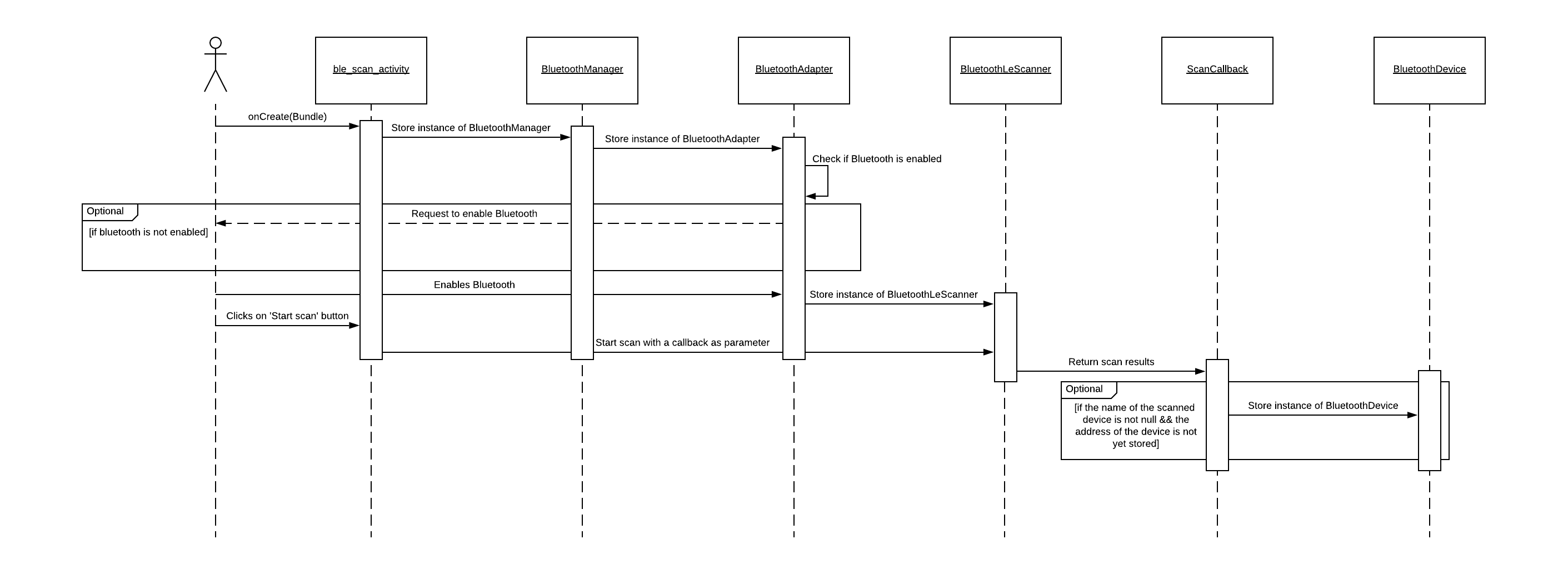
## Activities, Fragments en Services

Er worden verschillende componenten gebruikt bij het programmeren van de applicatie. Voor de verschillende speelmodi is Fragment gekozen, omdat deze geschikter zijn voor het opslaan of onthouden van data. Voor de Bluetooth knop rechtsboven in het scherm is gekozen voor een Activity en een Service. Door op de knop te klikken, wordt er een nieuwe Activity gestart. Hierin kan een verbinding worden gemaakt met de SoC. De Service wordt vervolgens gebruikt om deze connectie op te slaan en op de achtergrond te bewaren, omdat bij het wisselen van Activities, de data van de vorige Activity verloren gaat.

# Bluetooth verbinding

## Hoe wordt het voor de gebruiker mogelijk om in de applicatie te scannen naar BLE-apparaten?

Als de gebruiker de applicatie voor het eerst opent, is er nog geen bluetoothverbinding met een apparaat. De gebruiker wordt gevraagd toegang te verlenen tot zijn/haar locatie, omdat voor het detecteren en identificeren van bluetoothapparaten locatietoegang is vereist. Deze vereiste is toegevoegd in Android 6.0 om gebruikers te voorzien van betere gegevensbescherming. (Android, z.d.-a). Als de gebruiker vervolgens op het icoon van Bluetooth klikt, kan het scannen worden gestart door op de ‘Start scan’-knop te klikken. De werking van het scannen wordt aangetoond middels een sequence-diagram:



* BluetoothManager
  + Deze klasse wordt gebruikt in de onCreate()-methode en zorgt voor het verkrijgen van een instantie van BluetoothAdapter van het mobiele apparaat.
* BluetoothAdapter
  + Deze klasse wordt gebruikt voor het verkrijgen van een BluetoothLeScanner instantie en voor het controleren van de beschikbaarheid van Blueooth op het apparaat. Dit wordt ook gedaan in de onCreate()-methode.
* BluetoothLeScanner
  + Deze klasse bevat functies, zoals startScan(ScanCallback) en stopScan(ScanCallback). Deze functies worden gebruikt om te scannen naar BLE-apparaten en krijgen als parameter een ScanCallback mee.
* BluetoothDevice
  + Deze klasse wordt gebruikt om een instantie van de gescande SoC op te slaan in een variabele. Deze kan vervolgens worden gebruikt in andere klassen.
* ScanCallback
  + Deze klasse gebruikt de functie onScanResult(int, ScanResult). ScanCallback zorgt ervoor dat de resultaten van een BLE-scan terug worden gegeven aan de gebruiker. Met behulp van de onScanResult-functie kan bijvoorbeeld de naam van de SoC in de app worden weergegeven.

Verder moet de gebruiker aan de app toestemming geven tot locatie, zodat BLE-apparaten gevonden kunnen worden. De code hiervan staat echter niet in de Service.

## Hoe wordt de verbinding met hetzelfde apparaat hersteld als deze verbroken is?

Zoals hierboven beschreven kan in de Service een instantie van BluetoothDevice opgeslagen worden. Door de SoC op te slaan in BluetoothDevice, kan bijvoorbeeld het herstarten van de app direct worden verbonden met de SoC. De verbinding wordt dan alleen verbroken als de gebruiker hiervoor kiest door te verbinden met een ander apparaat of door de verbinding handmatig te verbreken. Als er opnieuw verbinding wordt gemaakt met de SoC dan wordt de eerder verstuurde key opnieuw gestuurd, hierdoor weet de SoC dat de app dezelfde is als die voorheen heeft verbonden.

## Hoe kan ervoor gezorgd worden dat er maar één en hetzelfde apparaat per aan tijd verbonden kan worden?

Deze vraag moet beantwoord worden voor Operationele specificatie software m4.4 en M4.5. In da1458x\_config\_basic.h staat “#define CFG\_MAX\_CONNECTIONS (1) “ dit geeft aan dat er maximaal 1 verbinding met de gelijktijdig SoC mogelijk is. Wanneer de Android app met de SoC verbonden wordt, wordt er een key gestuurd naar de SoC. Deze key wordt onthouden en gecheckt wanneer er een nieuwe verbinding wordt gemaakt, is de key niet gelijk aan de eerste key die verzonden is naar de SoC dan wordt de verbinding met “app\_easy\_gap\_disconnect(conidx)” verbroken.

## Hoe wordt een in run-time aanpasbaar bericht gestuurd naar de SoC?

De functie writeCharacteristic van BluetoothGatt wordt gebruikt om een String aan data te sturen naar een bepaalde characteristic op de SoC. Iedere keer dat de gebruiker via de custom speelmodus data verstuurd, wordt deze String aangepast en opnieuw verstuurd naar de SoC. Dit wordt bijgehouden met behulp van een BluetoothGattCallback. De BluetoothGattCallback roept de onCharacteristicWrite functie aan wanneer de gebruiker data verstuurt. Deze functie is echter niet noodzakelijk voor de SoC, maar geldt meer als een handigheid voor de app. Met behulp van onCharacteristicWrite kan er op basis van de data die is verstuurd een actie worden verricht.

## Hoe ziet de data die naar de SoC wordt gestuurd eruit?

Vanuit de app wordt er data gestuurd naar de SoC, zoals in hoofdstuk 3.4 is beschreven. De string die wordt verstuurd vanuit “Custom mode” bestaat uit

18 cijfers (6 x 3). De zes staat voor zes zijden en de drie voor het percentage. In het volgende voorbeeld is er 10% kans dat er één oog wordt gegooid, 20% kans voor twee ogen, 15% voor drie ogen, 30% voor vier ogen, 5% voor vijf ogen en 20% voor zes ogen: 010020015030005020. De som van deze waarden moet altijd 100% zijn. In “Normal mode” wordt er een string gestuurd met een lengte die kleiner is dan de string in “Custom mode”, zodat de SoC weet dat de gebruiker zich in “Normal mode” bevindt. Dit kan bijvoorbeeld “123” zijn.

## Hoe wordt een in run-time aanpasbaar bericht gestuurd naar de Androidapplicatie?

Voor het versturen van een bericht is een service en een characteristic nodig, hoe deze gemaakt wordenstaat in de appendix.

In user\_peripheral.c is een functie genaamd user\_catch\_rest\_hndl, deze functie vangt bluetooth aanvragen op. In deze functie staat een switch case waarin de ontwikkelaar aan kan geven wat er moet gebeuren bij bepaalde events. De switch case is te zien in Figuur 2 BLE event handler. Case DICE\_CHANGE\_CFG wordt uitgevoerd wanneer notificatie wordt aangezet door de verbonden telefoon en DICE\_CHANGE\_VAL wordt aangeroepen als de notificatie is aangezet en er bericht verstuurd wordt vanaf de telefoon naar de SoC.

Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving

Figuur 2 BLE event handler

DICE\_CHANGE\_CFG en DICE\_CHANGE\_VAL zijn beschreven in het “user\_custs1\_def.c”. Om berichten te versturen is het nodig voor de server om notificatie aan te zetten. Elke keer dat de status van de notificatie wordt aangepast wordt de code onder DICE\_CHANGE\_CFG uitgevoerd, via   
msg\_param->value[0] is te zien of de server notificaties aan of uit zet, 0 is uit anders is aan. Op dezelfde manier kan er data ontvangen worden in de case DICE\_CHANGE\_VAL.

Als de notificaties aan staat kan data verstuurd worden naar de server, dit is te zien in Figuur 3 Versturen Bluetooth bericht. In dit voorbeeld wordt 420 verstuurd. Dit wordt eerst gekopieerd naar een string en vervolgens wordt het bericht klaar gemaakt. Belangrijk is om de req->handle het deel van de characteristic te zetten dat er geschreven moet worden.

Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving

Figuur 3 Versturen Bluetooth bericht

# Accelerometer

Zoals beschreven staat in het hardware design document is het doel van de accelerometer het bepalen of de dobbelsteen geworpen is. De accelerometer kan dit bepalen door een versnelling in 6 richtingen te meten. De gekozen accelerometer kan een interrupt genereren als er een versnelling wordt gemeten in 6 richtingen. Dit kan gedaan worden door bit “6D” (naam niet hexadecimaal) te schrijven van register INT\_GEN\_CFG\_XL (06h). In de registers INT\_GEN\_THS\_X\_XL, INT\_GEN\_THS\_Z\_XL en INT\_GEN\_THS\_Z\_XL kan de threshold worden gezet voor de interrupt. Omdat de dobbelsteen niet gerealiseerd is, is niet bekend wat de juiste threshold is.

De interrupt op de SoC wordt gezet door eerst de interrupt aan te zetten vervolgens wordt de interrupt niveau gezet en als laatste wordt de callback functie gezet, dit gebeurt met de functies “GPIO\_EnableIRQ”, “GPIO\_SetIRQInputLevel” en “GPIO\_RegisterCallback” in de functie “dice\_sensor\_api\_set\_isr“ in de header “dice\_sensor\_api”.

Belnagrijk is het toevoegen van de volgende if-statement in de isr voor de rede die er als comment bij staat.

Afbeelding met binnen, oranje, vasthouden, man

Automatisch gegenereerde beschrijving

Figuur 4 If-statement die het triggeren tijdens opstarten voorkomt

# Gyroscoop

Na onderzoek is gekozen voor een deep neuraal netwerk (zie Magic effect). Een neuraal netwerk maakt een formule aan de hand van input en wat de output moet zijn. Voor natuurkundige formules moet de waarde van de gyroscoop naar een standaard eenheid worden geschreven maar voor een neuraal netwerk is dat wellicht niet nodig, dit moet getest worden door met de dobbelsteen een behoorlijk aantal keer te rollen. Helaas is dit niet getest omdat er geen dobbelsteen is om mee te rollen. Een andere reden om de data van de gyroscoop niet om te schrijven naar heading, pitch en roll is de berekeningstijd. Het omschrijven gebeurt met atan2, wortels en delen van doubles, dit zijn tijdsintensieve operaties die wellicht vaak uitgevoerd moeten worden, aan de hand van de nauwkeurigheid van het neuraal netwerk, hoe minder nauwkeurig hoe vaker berekend moet worden.

# LED

Hoe de leds aangestuurd worden staat beschreven in het hardware ontwerp. Er zijn twee verschillende manieren om een functie over x seconde aan te roepen zonder de code te blokkeren. Het blokkeren van de code is in strijd met M2.10, M4.4 en M4.5. Manier één is gebruik maken van “app\_easy\_timer”, voordeel meerdere timers kunnen hiermee gemaakt worden, nadeel de minimale tijd is 10ms. Een andere manier is gebruik maken van “systick”, voordeel de theoretische minimale tijd is 1us maar dit zal in praktijk meer zijn omdat het systeem anders te weinig tijd heeft. Een nadeel is dat er maar één timer gemaakt kan worden met “systick”.

60hz is gekozen als vernieuwingsfrequentie voor elke zijde van de dobbelsteen, omdat dit een veel gebruikte vernieuwingssnelheid is en met deze snelheid niet te zien is dat de zijdes knipperen.

De formule voor de LED\_UPDATE\_TIMER = ((1/frequentie)/hoeveelheid zijdes) \* 10^6. Ingevuld geeft dit ((1/60)/6) \* 10^6 = 2777us. 10ms > 2777us dus voldoet “app\_easy\_timer” hier niet aan. “systick” is gekozen omdat het wel aan 2777us kan voldoen en het wordt nog niet gebruikt. Het is mogelijk om meerdere “app\_easy\_timer” te gebruiken om wel 60hz te halen maar hierdoor wordt de code minder overzichtelijk.

In Figuur 5 Activity diagram voor het updaten van de leds is beschreven hoe de dobbelsteen de zijdes update wanneer de cijfers op de zijkante niet veranderen. Als de cijfers op de zijdes wel moeten veranderen dan wordt er met “led\_change\_number\_on\_side(side,number)” eerst de zijde voorzien van het nieuwe nummer dan worden andere zijkanten ook aangepast zodat altijd twee overstaande zijde een som van 7 hebben.

Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving

Figuur 5 Activity diagram voor het updaten van de leds

# Magic effect

Met magic effect wordt bedoeld dat de dobbelsteen gedurende de rol de zijdes kan veranderen waardoor een gewenst nummer boven aan komt te liggen. De data die gebruikt kan worden om gedurende de rol te bepalen welke zijde uiteindelijk boven komt te liggen is de draaiversnelling in drie richtingen en de oriëntatie. De dobbelsteen begint met het berekenen als er een interrupt komt van de accelerometer.

## Natuurkundig probleem

In eerste instantie lijkt het op een natuurkundig probleem op te lossen met een formule, in theorie is dat ook zo, maar in praktijk is er nauwelijks informatie te vinden over een dergelijke formule. Wanneer er wel wat over een rollende kubus werd gezegd ging men ervan uit dat de kubus niet stuitert, niet glijdt en dus zich eigenlijk in een totaal surrealistische wereld bevindt, dit biedt geen oplossing. Er moet dus worden gekeken naar een andere oplossing.

## Machine learning

Machine learning is aan de hand van data formule(s) aanpassen zodat het patronen herkend. Omdat het maken van een natuurkundige formule voor het bepalen op welke zijde de dobbelsteen terecht komt erg ingewikkeld is, is het beter om dit de computer te laten doen. Er zijn verschillende manieren modellen

### Verzamelen van data (simulatie)

Omdat er geen dobbelsteen is moet er op een ander manier gekeken worden hoe goed bepaalde modellen werken. Een simulatie in Unity is gekozen om data te verzamelen omdat er veel informatie over te vinden is en de oriëntatie van objecten verkrijgbaar is. Een ander programma waarnaar gekeken is, is Blender. Blender is niet gekozen omdat dit programma meer gericht is om het maken van mooie animaties dan het simuleren.

Om te kunnen bepalen op welke zijde de dobbelsteen terecht komt moet ook de juiste data gemeten worden. Alleen de rotatiesnelheid is niet voldoende om te bepalen op welke zijde de dobbelsteen terecht komt, ook de begin oriëntatie is belangrijk. Zowel de oriëntatie als de rotatiesnelheid heeft drie waardes nodig om beschreven te worden en dit geeft zes waardes die gebruikt kunnen worden om de bepalen te doen.

### Mogelijke modellen

Machine learning zou goed gebruikt kunnen worden omdat in theorie een formule voor het probleem is te maken. Machine learning bestaat niet alleen uit (deep) neuraal netwerks maar ook uit andere varianten zoals

* Lineaire functie maken (lineaire regressie)
* Groeperen van data (kmeans)
* Kijken naar de dichtstbijzijnde buur(en) (knn)

Lineaire regressie werkt niet omdat die door de data een lijn probeer te trekken, hieruit komt dus altijd het gemiddelde en het gemiddelde van een rol is 3.5. Voor meer informatie over lineaire regressie zie Figuur 7 Linear regression en referentie 1.1.

Kmeans en knn beide werken met groepen maar op een verschillende manier, voor informatie over kmeans zie Figuur 8 kmeans en referentie 1.3, voor meer informatie over knn zie Figuur 9 kNN en referentie 1.2. Het is moeilijk om te beredeneren of het groeperen van data in dit geval een gewenst resultaat geeft daarom zijn kmeans en knn beide geprobeerd en geven beide een nauwkeurigheid van ongeveer 1/3. 1/3 is twee keer beter dan gokken maar zeker geen acceptabele waarde.

Er moet dus gekeken worden naar een deep neuraal netwerk.

### Deep neuraal netwerk

De werking van een neuraal netwerk is beschreven in referentie 1.4.

Deep neuraal netwerks bestaan uit inputs, in dit geval zes (x, y, z voor begin oriëntatie en x, y, z van de rotatiesnelheid), outputs in dit geval ook zes (zes verschillende zijdes van een dobbelsteen) en verborgen lagen. Veel bronnen zeggen over de architectuur van een network dat de beste manier om een zo goed mogelijk model te vinden is, is simpel weg verschillende netwerken proberen. In netwerken is de activation functie, de hoeveelheid verborgen lagen en de hoeveelheid nodes per laag aan te passen.

Als activation functie is gekozen voor makkelijk te berekenen sigmoïd functie. De uitkomst van de formule is 0 als x < -2.5, de uitkomst is 1 als x > 2.5 en anders is de uitkomst gelijk aan 0.2x + 0.5. Dit is een soort sigmoïd functie maar dan zonder de rondingen en dat maakt het sneller te berekenen. De snelheid waarmee het berekend wordt is belangrijk omdat het voor elke verbinding tussen nodes berekend moet worden, dit kan oplopen tot duizenden keren.

Na proberen is gebleken dat vooral de hoeveelheid nodes in de eerste verborgen laag grote impact heeft op de nauwkeurigheid, hoe meer hoe beter. Met 240 nodes en één hidden layer is een nauwkeurigheid van 80% gehaald, dit moet wel genuanceerd worden. De data die aan het netwerk gegeven wordt is niet alleen het moment dat de dobbelsteen de grond raakt maar ook ongeveer 60 momenten totdat de dobbelsteen tot stilstand komt en hoe verder in de rol hoe makkelijker te bepalen is op welke zijde die uiteindelijk terecht komt.

### Integreren in de SoC

Tot verbazing is er geen code beschikbaar om een neuraal netwerk in C te implementeren, daarom zal er C code geschreven worden die alleen het voorspel gedeelte van een neuraal netwerk doet aan de hand van vooraf gegeven weights, biases en netwerk architectuur. De C code wordt voorzien door een script die een Keras neuraal netwerk als input neemt en daarvan defines en arrays genereerd waarmee de C code aan de slag kan.

De C code verwacht een drietal defines waarmee informatie gegeven wordt over het netwerk

#define NETWORK\_SIZE x is de hoeveelheid lagen in het netwerk (inclusief input en output lagen)

#define MAX\_NOTES\_IN\_LAYER x is het maximale aantal nodes in één laag, dit wordt gebruikt om arrays mee te initialiseren.

#define BIGGEST\_CONNECTION x is de grootste verbinding tussen twee lagen, ook om een array mee te initialiseren.

Verder wordt er ook een drietal arrays verwacht

static const float BIAS[] = x is een één dimensionaal array met alle biases.

static const uint16\_t LAYER\_SIZE[NETWORK\_SIZE] = x bevat de architectuur van het netwerk waarvan de eerste plek de hoeveelheid input nodes is, tussenplekken zijn de verborgen lagen en de laaste plek de hoeveelheid output nodes is.

static const float WEIGHTS[] = x bezit alle weights van het netwerk.

#### Verzamelen data in de SoC

Omdat de echte dobbelsteen zeker anders zal zijn dan de dobbelsteen in simulatie moet er ook data verzameld worden van de echte dobbelsteen. De dobbelsteen bevat 250KB aan flash om gegevens in op te slaan. Hiervan is 32KB al in gebruik door de code. De rest van het flash kan gebruikt worden om rollen in op te slaan. Er is gekozen voor het werken met pages omdat dit sneller is dan het werken met bytes. Elke page is 256 bytes groot, dit betekent dat er ruimte is voor (250K – 32K)/256 = 851 pages. De data wat verzameld moet worden per meting is in totaal 12 bytes groot (x, y en z zijn twee bytes per stuk en dat keer twee voor de oriëntatie en de rolsnelheid). 256/12 = 21 met 4 bytes over. De op één na laatste byte van de page wordt gebruikt om de hoeveelheid metingen in de page aan te geven en de laatste byte wordt gebruikt om de uitkomst op te slaan, welk getal bovenop te lezen is -1, -1 omdat het neuraal netwerk werkt van 0 tot en met 5.

# Draadloos programmeren

Voordat draadloos programmeren besproken wordt moet eerst het bootproces duidelijk zijn van de SoC. De SoC kijkt naar het flash en zoekt op adres 0x00000000 naar 0x70500000000012E4. Staat dit er dan wordt de rest van het programma in ram geladen tot dat de SoC EOF tegen komt in het flash. Voor meer informatie kijk in de referentie 1.6 op pagina 4 .

Een eis van de klant is dat het mogelijk is om draadloos software updates te doen (S2.2), dit kan met OTA. De sequence diagram die dit proces beschrijft is Figuur 6 OTA draadloos software updaten te vinden in de apendix. De informatie die ontvangen wordt via Bluetooth moet naar het flash worden geschreven, dit gebeurt in “OTA STREAM”.

Het eerste idee was om eerst de host (Amazon Echo in de figuur) alles naar de SoC te laten versturen. De SoC schrijft vervolgens alles naar flash. Als alles eenmaal verstuurd is wordt er een CRC gedaan en klopt het gebeurt er niets anders wordt het flash verwijderd en begint het proces opnieuw. Dit gaat goed zolang het proces een keer succesvol afgerond is voordat de SoC uit gaat. Want als de SoC uitgaat voordat het proces succesvol is afgerond dan kan er nergens van geboot worden en is de dobbelsteen onbruikbaar.

De oplossing is precies hetzelfde proces als hierboven beschreven is maar dan wordt de ontvangen data niet op de bootlocatie gezet maar op een andere locatie in het flash. Als de CRC succesvol is dan wordt de data in de andere locatie gekopieerd naar de bootlocatie. In Verzamelen data in de SoC staat beschreven hoe flash gebruikt wordt voor het trainen van de dobbelsteen maar als eindproduct gebeurt dit niet meer en daarom is deze oplossing mogelijk. Er bestaat nog steeds het gevaar dat tijdens het kopiëren van geheugen de SoC wordt uitgezet maar aan dat risico is niet te ontkomen. Wat wel mogelijk is, is een check vooraf die controleert of de batterij boven 50% is zodat de dobbelsteen altijd genoeg stroom heeft om de update uit te voeren.

# Bibliografie

* Android. (z.d.-a). *Android 6.0 Changes |* . Geraadpleegd op 28 mei 2020, van <https://developer.android.com/about/versions/marshmallow/android-6.0-changes>

1. Figuren

Afbeelding met schermafbeelding

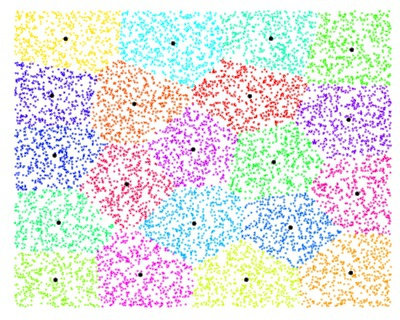
Automatisch gegenereerde beschrijving

Figuur 6 OTA draadloos software updaten

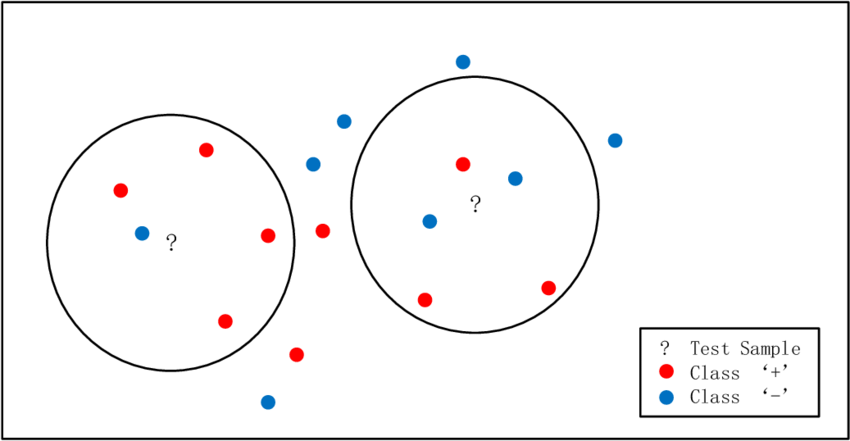
Afbeelding met kaart, foto, verschillend, man

Automatisch gegenereerde beschrijving

Figuur 7 Linear regression



Figuur 8 kmeans

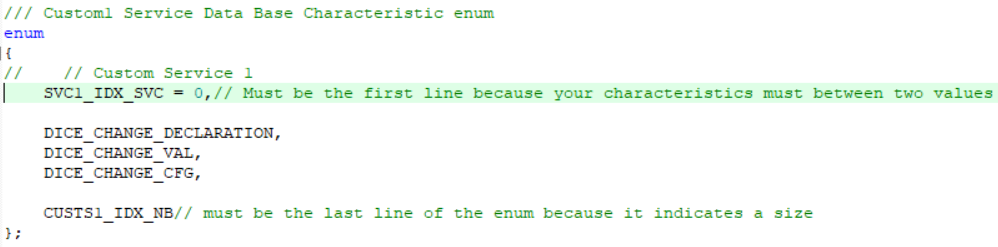


Figuur 9 kNN

1. Maken van een service

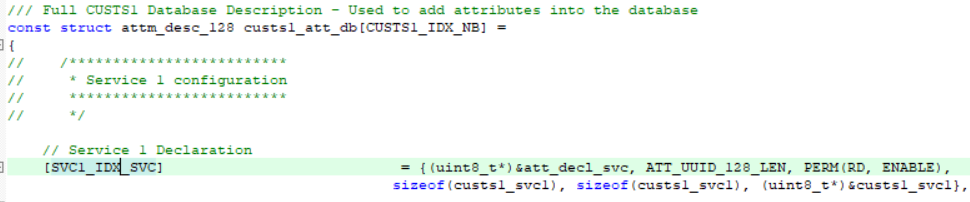
Een service is een groep waarin characteristics gestopt kunnen worden. Het is mogelijk om alle characteristics in één service te stoppen maar services zijn bedoeld om characteristics te groeperen met hetzelfde onderwerp.

Een service wordt gemaakt door in user\_custs1\_def.h een in de enum een naam toe te voegen, in dit geval is de naam SVC1\_IDX\_SVC. De service moet een een waarde hebben die gelijk is aan de laagste characteristic – 1, het moet dus hoger in de enum staan zoals hieronder te zien is. In dit voorbeeld heeft de laagste characteristic van deze service een waarde 1 dus moet de service de waarde 0 hebben.



Figuur 10 Service enum

Als de service is toegevoegd in de enum dan wordt deze nu in user\_custs1\_def.c in de custs1\_att\_db struct toegevoegd zoals hieronder weergegeven is, de struct moet custs1\_att\_db heten omdat dit gebruikt wordt in drivers via een extern! att\_dec1\_svc is een id dat aangeeft dat dit een service declaratie is, deze declaratie is 128 bits lang en dat verklaart het volgende ATT\_UUID\_128\_LEN. Alle services hebben alleen het permission read (RD). Daarna volgen twee grotes, de eerste geeft de maximale grote aan en de tweede geeft de huidige aan. Dit is beide de grote van custs1\_svc1 wat het id is van de service. Het laatste geeft de huidige waarde aan.



Figuur 11 CuSTS1 database

Als laatste voeg je de service toe aan custs1\_services, let op deze moet ook altijd deze naam hebben vanwege dezelfde reden als custs1\_att\_db! CUSTS1\_IDX\_NB moet altijd in het laaste element van de array staan en het moet als laatste voorkomen in de enum in user\_custs1\_def.h, dit komt omdat CUSTS1\_IDX\_NB een grote aangeeft.



Figuur 12 toevoegen service