Opleverdocumentatie

Inleiding

Vanuit de Hogeschool van Arnhem en Nijmegen (HAN) zijn er multidiciplinaire projectteams samengesteld om het probleem van Burgers' Zoo aan te pakken. Een eerder team heeft acht weken gewerkt aan een mogelijke oplossing. Daarbij zijn er onderzoeken gedaan, concepten ontworpen, prototypes uitgewerkt en getest. De resultaten hiervan zijn in een adviesrapport verwerkt en overhandigd aan een tweede projectteam. Dat projectteam heeft de focus gelegd op de technische haalbaarheid van de ontworpen concepten en dit ook op grotere schaal getest. Dit tweede team heeft de bevindingen hiervan verwerkt in dit document om te overhandigen aan een volgende projectgroep.

Burgers' Zoo heeft vanuit eigen onderzoek aangegeven dat bezoekers zich negatief uiten over de mobiliteit in het park en daardoor hun bezoek als minder prettig ervaren. Het bezoek wordt gezien als een dagtaak wat ook zwaar is door de lange afstanden lopen op heuvelig terrein. Hierdoor zijn bezoekers van het park minder snel geneigd om het park opnieuw te bezoeken. Burgers' Zoo heeft dit probleem voorgelegd aan de HAN, waarna meerdere projectteams hier mee aan de slag zijn gegaan.

In dit rapport wordt allereerst het concept beschreven, gevolgd door een beschrijving en uitleg van de technieken die gebruikt zijn. Daarbij is een handleiding opgenomen om een demo te starten. Daarna wordt er beschreven hoe er verder gewerkt kan worden met de software en hardware die opgeleverd is. Tot slot zijn er aanbevelingen voor een volgende projectteam. Achter aan het verslag is nog een lijst te vinden met installatieinstructies voor de genoemde software producten.

Inhoudsopgave

- Concept
- Systeemoverview
    - Gateway
    - Frontend apps
    - Backend
    - Database
- Opzethandleiding
    - Starten Backend + Web Apps
        - Randvoorwaarden
        - Het starten van de web applicaties en de database
            - Installeren van de dependencies
            - Development applicaties
            - productie builds (minified)
            - De applicaties bezoeken
    - Installatie Gateway

            - MongoDB
            - Java
            - RXTX
            - Aansluiting Arduino
            - Start gateway
        - Poot
            - 1. Onderdelen
            - 2. Behuizing
            - 3. Aansluiting
            - 4. Code uploaden
            - 5. SD kaart
            - 6. Aanzetten
                - Maduino
                - Auduino
            - 7. Pasjes
- Ontwikkelhandleiding
    - Ontwikkeling front-end applicaties
        - Redux
        - Folderindeling (package structuur)
        - Tips
    - Loopback
        - Installatie
        - Gebruik:
            - Maak een eerste app:
            - Maak een API
            - Data opslaan
            - Nieuwe modellen aanmaken
            - Relaties
            - Modellen aanpassen
            - Eigen endpoints toevoegen
            - API explorer
                - Filters
            - Swagger Generator
    - Ontwikkeling Gateway
        - Eisen
        - Werking Gradle
        - Folder structiuur
- Seriële communicatie
    - Uitvoeren op de Raspberry Pi 3
    - Afspraken MySensors protecol

        - Ontwikkeling Poot
            - Ontwikkelomgeving
            - Libraries
                - Audino
                - Maduino
- Bijlagen
    - Repo commando's
    - Software lijst

Concept

In het park van Burgers' Zoo wordt een speurtocht uitgezet om zo het mobiliteitsprobleem te verminderen door afleiding voor de kinderen.

Veel voorkomende bezoekers in Burgers' Zoo zijn kinderen met hun (groot)ouders. Kinderen zitten vol met energie en zijn enthousiast dus rennen door het park heen. Tegelijkertijd lopen de ouders hier achteraan en moeten ze de kinderen in de gaten houden. Door het heuvelachtige landschap waarop Burgers' Zoo gebouwd is kan dit heel vermoeiend zijn. Een speurtocht geeft de kinderen afleiding en zorgt er voor dat de kinderen bij een punt in de speurtocht stil blijven staan.
In het park staan bij elk dierenverblijf borden met informatie over de dieren of planten die daar te vinden zijn. Dit zijn vaak lange stukken tekst die de kinderen zelf niet kunnen lezen. Dit zorgt er voor dat de ouders dit moeten voorlezen. Dit brengt dus extra last met zich mee voor de ouders. Dit wordt opgelost met de speurtocht doordat er weetjes afgespeeld worden in kindertaal.

De speurtocht is bedoeld voor kinderen om een ranger te spelen. De kinderen krijgen bij binnenkomst een NFC pas, hun rangerpas, die ze kunnen gebruiken om mee te doen aan de speurtocht. Door het park heen staan speurpunten, in de vorm van een dierenpoot, verspreid. Wanneer een ranger zijn rangerpas scant bij een poot krijgt hij een dierenweetje te horen samen met een dierengeluid. Het doel is om alle dierenpoten te vinden en te scannen. Als de ranger de dierentuin verlaat krijgt hij een ranger certificaat. Op dit certificaat staat een unieke ranger code die hij kan gebruiken om thuis in te loggen op de ranger app.

In de ranger app kan een ranger kijken wat hij allemaal gedaan heeft tijdens zijn bezoek aan Burgers' Zoo. Hier is te vinden welke poten hij wanneer gescand heeft. Hier is te zien welk weetje en dierengeluid er bij hoort, zodat dit nogmaals geluisterd kan worden. De app geeft ook aan welke punten nog niet gescand zijn. Dit kan een aanleiding zijn om het park nog een keer te bezoeken, om zo alle punten van de speurtocht te vinden. Hier kunnen beloningen aan gekoppeld worden in de vorm van korting of een gift. Medewerkers van Burgers' Zoo kunnen de poten en de speurtocht beheren door hier andere dierengeluidjes of weetjes aan toe te voegen.

Systeemoverview

De poot bestaat uit twee Arduino's. Er is één Arduino die volledig gaat over het afspelen van audio (genaamd Auduino). In een later stadium zou deze Auduino ook verantwoordelijk worden voor het opslaan van nieuwe ontvangen audiobestanden. De Auduino wordt aangestuurd door de Master Arduino.

De Master Arduino (genaamd Maduino) is verantwoordelijk voor alle primaire functionaliteiten en het aansturen van de Auduino. Zo zal de master Arduino een NFC-scanner hebben om passen te detecteren. Ook zal de Maduino de temperatuur en luchtvochtigheid meten. De Maduino staat via de NRF24 chip in verbinding met de Gateway en zal zo de Gateway op de hoogte houden over welke passen gesand zijn.

De resultaten van een speurtocht van bezoekers kunnen ingezien worden in de Ranger app. Dit is een webapp die na het bezoek aan Burgers' Zoo gebruikt kan worden om terug te kijken op een boezoek aan het park. De verkregen informatie wordt hier nogmaals getoond.

Beheerders van Burgers' Zoo kunnen via de Admin app de speurtocht bijhouden en aanpassen. Zo is het mogelijk op poten opnieuw te configureren of nieuwe geluiden of weetjes toe te voegen aan de speurtocht

De Ranger app en de Admin app draaien samen met de Backend en de database in Docker containers. Dit alles draait op een server.

Gateway

De poten zullen communiceren met de twee backends van de twee groepen via een gateway. Deze gateway bestaat uit een Arduino en een Raspberry Pi. De Arduino zal draadloos communiceren via NRF24 chips met de poten en alle informatie doorsturen naar de Raspberry Pi. De Pi zal via HTTP/JSON communiceren met de backend's. De Pi kan op zijn beurt weer de Arduino binnen de gateway aansturen om zo informatie bij de poten te krijgen.

Frontend apps

De gebruikers zullen werken met een van de twee client-applicaties: de Ranger App voor de rangers en de Admin App voor de administratoren. Deze twee applicaties draaien in de browser en zullen via HTTP/JSON communiceren met de Backends.

De front-end apps zijn modulair opgezet, deze apps draaien op hun eigen plekje en roepen het REST backend middels HTTP aan. Als zei data willen manipuleren zal dit dus ook via de back-end moeten verlopen. Alle front-end apps bij elkaar worden gezien als de "front-end laag", zelfs als deze op andere fysieke machines draaien. Het los koppelen van de applicaties bevordert de werkbaarheid en stabiliteit van de architectuur. Elke app kan afzonderlijk gedeployed / getest worden zonder de rest van de architectuur te beïnvloeden.

Backend

Het back-end betreft een REST api welke wordt aangesproken met de verschillende front-ends. De REST api zelf spreekt de datalaag aan om zijn data op te slaan en op te halen. Deze laag kan wederom uitgebreid worden met meerdere instanties van de back-end en/of met een loadbalancer.

Database

De database laag zal enkel en alleen de database bevatten, op het moment van prototyping is dit één Mongo database. Dit kan echter uitgebreid worden met meerdere instances (voor redundancy, uitbreidbaarheid) en eventueel voorzien worden van een load balancer.

Opzethandleiding

Om het systeem in zijn geheel op te zetten zijn drie onderdelen nodig: Web-Backend + Webapps, Gateway en Poot. In onderstaande drie koppen wordt uitgelegd hoe deze drie onderdelen in te stellen. Wanneer alle drie de onderdelen opgezet zijn kan het systeem gebruikt worden.

Starten Backend + Web Apps

Dit hoofdstuk zal beschrijven hoe alle webapplicaties, de backend en de database opgestart moeten worden. Ook zal dit hoofdstuk beschrijven hoe de database gevuld kan worden met het seedscript zodat er wat testdata in de apps staat.

Randvoorwaarden

Om alle applicaties te draaien moeten er een aantal dingen geregeld worden op de pc/laptop. De tabel hieronder geeft aan welke stukken software benodigd zijn en zal, waar mogelijk, een link worden geven naar de officiele website.

- Mongo (https://www.mongodb.com/)
- Docker (https://www.docker.com/)
- Docker-compose (https://docs.docker.com/compose/)
- Node (https://nodejs.org/)
- Npm (https://www.npmjs.com/)

Als alle bovenstaande software geinstalleerd is dan kunnen alle apps gestart worden, om de database te vullen is er echter nog een extra stukje software nodig. De software heet `mongorestore` en komt , ten tijde van schrijven, mee geinstalleerd met het mongo pakket (Op Windows met mongotools (https://github.com/mongodb/mongo-tools)). Bekijk deze website (https://docs.mongodb.com/manual/reference/program/mongorestore/) voor meer informatie.

Het starten van de web applicaties en de database

Het volgende hoofdstuk zal uitleggen hoe de applicaties gestart kunnen worden in zowel development modus als productie modus. Voor het testen is alleen de development modus meer als genoeg.

> ***NOTE! de commando's zijn bedacht voor Linux en Mac OS X, hieronder wordt beschreven hoe het werkt voor alle drie de systemen al is het zeer aan te raden om een Linux Virtual machine op te zetten. (klik hier (https://www.storagecraft.com/blog/the-dead-simple-guide-to-installing-a-linux-virtual-machine-on-windows/) voor uitleg)***

Installeren van de dependencies

Om de benodigde NPM software te installeren moet het volgende commando gedraaid worden:

```
npm install
```

Dit zal tegelijkertijd de docker containers die nodig zijn voor het draaien van de software aanmaken.
Om te controleren of het installeren goed is gegaan kan je in de opdrachtprompt zien hoe lang het geduurd heeft.

Dit ziet er als volgt uit:

```
up to date in 2.16s
```

Development applicaties

Om de applicaties in development modus te starten (in Docker) moet het volgende commando worden uitgevoerd:

```
npm start
```

Dit zal, onder water, het volgende draaien:

```
npm run copy-endpoint-dev && docker-compose down && docker-compose up
```

Het commando "copy-endpoint-dev" gaat echter **fout** op een Windows systeem omdat Windows geen fatsoenlijke copy tool op de command line heeft. Als het toch op windows moet draaien moet je de kopieerstap even zelf uitvoeren. De stappen zijn dan als volgt:

1. Kopieer het bestand `./config/dev.json` naar de volgende twee locaties:
    - *./apps/admin/src/constants/external-variables/endpoints.json*
    - *./apps/ranger/src/constants/external-variables/endpoints.json*
2. Draai het `docker-compose down` commando
3. Draai het `docker-compose up` commando

productie builds (minified)

Om de applicaties in productie modus te starten (in Docker) moet het volgende commando worden uitgevoerd:

```
npm run build
```

Dit zal, onder water, het volgende draaien:

```
npm run copy-endpoint-prod && docker-compose -f ./dockerfiles/prod/docker-compose.yml down && docker-compose -f ./dockerfiles/prod/docker-compose.yml up && bash seedscript.sh -h servers.rickvanlieshout.com:8019
```

Vervang hier de server (`servers.rickvanlieshout.com:8019`) met de juiste productie server en verander ook de variabelen in `./config/prod.json` naar de benodige productie variabelen.

Het commando "copy-endpoint-dev" gaat echter **fout** op een Windows systeem omdat Windows geen fatsoenlijke copy tool op de command line heeft. Als het toch op windows moet draaien moet je de kopieerstap even zelf uitvoeren. De stappen zijn dan als volgt:

1. Kopieer het bestand `./config/prod.json` naar de volgende twee locaties:
    - *./apps/admin/src/constants/external-variables/endpoints.json*
    - *./apps/ranger/src/constants/external-variables/endpoints.json*
2. Draai het `docker-compose down` commando
3. Draai het `docker-compose up` commando

4. Draai het `bash seedscript.sh SERVERURL` bestand. (op Windows heb je hier de bash shell voor Windows (https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/) voor nodig.)

De applicaties bezoeken

Om de applicaties te bezoeken, en ze te gebruiken, ga je naar de volgende web adressen:

Applicatie	Adres in develop modus	Adres in productie modus
Back-end api	http://localhost:8001	http://localhost:8011
Admin / educatie applicatie	http://localhost:8002	http://localhost:8012
Ranger applicatie	http://localhost:8003	http://localhost:8013
De database	http://localhost:8009 mongo://localhost:8009	http://localhost:8009 mongo://localhost:8009

Installatie Gateway

Om de gateway werkend te krijgen zijn er een aantal vereisten:

- Raspberry Pi 3.
- Arduino Nano, Mega of Uno met een aangesloten NRF24L01+.
- Er is een werkende versie van `RASPBIAN STRETCH WITH DESKTOP` geïnstalleerd. Mogenlijk werkt het ook met de minimale versie, echter is dit (nog) niet getest. Download (https://www.raspberrypi.org/downloads/raspbian/)
- Er is toegang via SSH of direct op de Raspberry Pi 3 terminal toegang.

MongoDB

Omdat er het een en ander wordt opgeslagen op de Raspberry Pi 3 moet er een database geïnstalleerd worden, in dit geval MongoDB.

Dit kan op de Raspberry Pi 3 gedaan worden met de volgende commando's in de terminal:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install mongodb-server -y
```

Het upgrade process kan een tijdje duren.

Als de MongoDB server succesvol geïnstalleerd is, kan deze service gestart worden door:

```
$ sudo service mongodb start
```

Deze service zal geen console uitput geven aangezien dit in de achtergrond draait.

Java

De gateway draait in een JVM en het is dus nodig om de juiste Java installatie te installeren.

```
sudo su
# Het kan zijn dat je op dit moment een wachtwoord moet invullen, dit is om het root account te gebruiken
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee /etc/apt/sources.list.d/webupd8team-java.list
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886 -y
```

Mocht bij het uitvoeren van het bovenstaande een error naar voren komen over het ontbreken van `dirmngr` dan kan dat gefixed worden door dit te installeren:

```
sudo apt-get install dirmngr -y
```

Tijdens het installeren van Java 8 is het gebruikelijk dat er gebruikersvoorwaarden geaccepteerd moeten worden, deze moeten geaccepteerd worden.

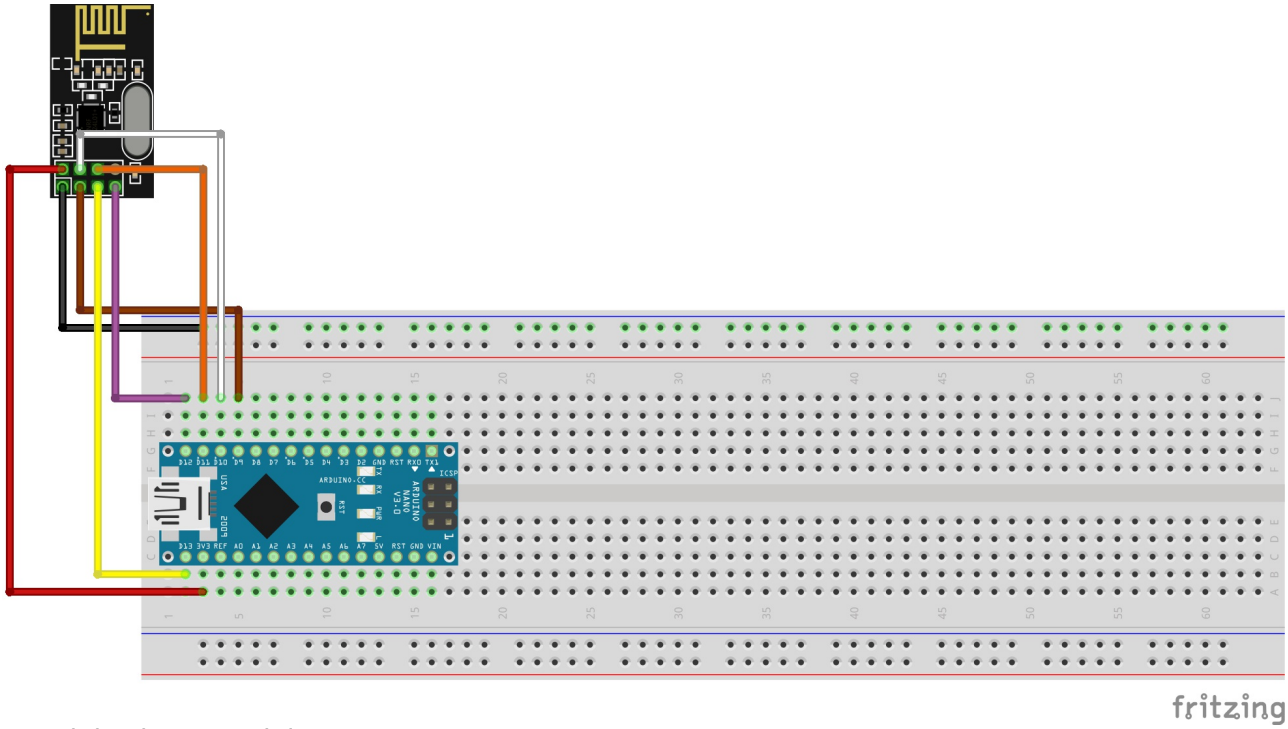
```
sudo apt-get update
sudo apt-get install oracle-java8-installer -y
```

RXTX

De Java gateway applicatie praat via het Serial protocol met de arduino. Hiervoor is de RXTX library nodig. Installeer deze library met het volgende commando:

```
sudo apt-get install librxtx-java -y
```

Aansluiting Arduino



Aansluitschema Arduino Nano met een NRF24L01+

De Arduino moet vervolgens verbonden worden via een USB kabel met de Raspberry Pi 3.

Start gateway

Om de gateway te starten moet eerst de MongoDB aan staan. Dat kan met het eerste commando.
Als deze draait kan daarna de gateway zelf gestart worden.

```
sudo service mongodb start #starts mongo service

java -Djava.library.path=/usr/lib/jni -jar gateway.jar #gateway.jar is te vervangen met de jar naam van de gateway
```

Bij het tweede gedeelte, om Java te starten, is het belangrijk dat de volgende regel voor de -jar komt: `-Djava.library.path=/usr/lib/jni`. Anders wordt de RXTX library niet goed geladen.

Poot

In dit hoofdstuk wordt beschreven hoe een fysieke poot kan worden gebouwd.

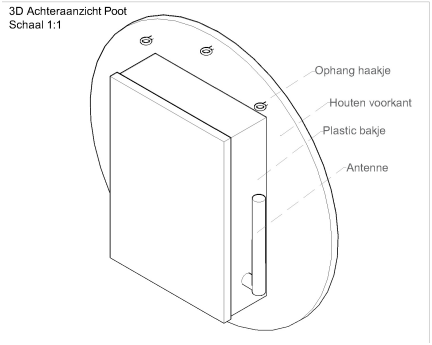
1. Onderdelen

Voor het bouwen van de Poot zijn de volgende interne componenten nodig. Daarnaast zijn er ook nog materialen voor de behuizing nodig. Deze worden in het volgende hoofdstuk behandeld.

- 2x Arduino Nano
- Rode led
- Oranje led
- Groene led
- Blauwe led
- Long Range NRF24L01+
- MicroSD kaart SPI lezer
- MicroSD kaart (max 32 gb) (zie onderstaande hoofdstuk)
- RFID-RC522 (Pas lezer)
- Socket Adapter Plate voor NRF24L01+
- Speaker met 3.5mm audio jack input
- 3.5mm audio jack output for arduino
- Meer dan genoeg Jumper wires
- Breadboard
- Pasjes (zie laatste hoofdstuk)

2. Behuizing

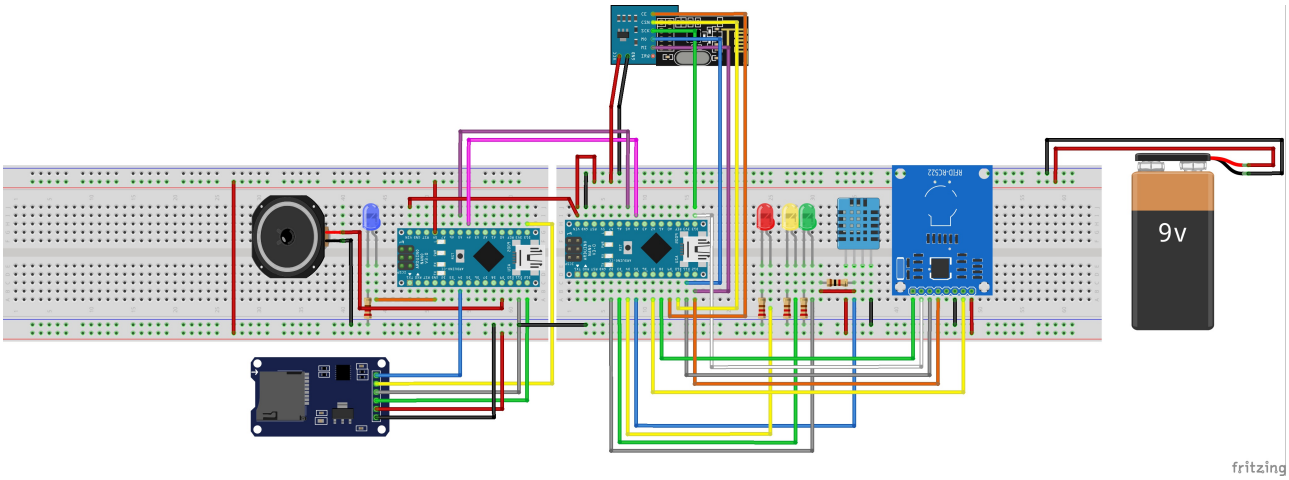
De behuizing van de poot bestaat uit een houten voorplaat met op de achterkant een plastic bakje met daarin alle electronica. Hieronder een sfeerimpressie van de behuizing.



Een gedetailieërd ontwerp voor de behuizing is [hier (https://github.com/HANICA-MinorMulti/nj2017-iot-dwa-BurgersZoo1/blob/master/documentatie/technisch-ontwerp/poot-fysiek/poot-fysiek-ontwerp.md)](https://github.com/HANICA-MinorMulti/nj2017-iot-dwa-BurgersZoo1/blob/master/documentatie/technisch-ontwerp/poot-fysiek/poot-fysiek-ontwerp.md) te vinden.

3. Aansluiting

Sluit de twee arduino's aan volgens het onderstaande aansluitschema of gebruik de aansluittabellen daaronder om de verbindingen tussen componenten en Arduino's te maken.



Auduino

Component	Pin op Component	Pin op Arduino
Blauwe LED	+	D2
Blauwe LED	-	GND
Speaker	+	D9
Speaker	-	GND
SD-reader	SCK	D13
SD-reader	CS	D4
SD-reader	MOSI	D11
SD-reader	MISO	D12
SD-reader	VCC	5V
SD-reader	GND	GND
Maduino	A4	A4
Maduino	A5	A5
Maduino	GND	GND

Maduino

Component	Pin op Component	Pin op Maduino
Groene LED	+	D2
Groene LED	-	GND
Oranje LED	+	D3
Oranje LED	-	GND
Rode LED	+	D4
Rode LED	-	GRND
NRF	VCC	3.3V

NRF	GND	GND
NRF	CE	D10
NRF	CSN / CS	D9
NRF	CSK	D13
NRF	MOSI	D11
NRF	MISO	D12
NRF	IRQ	NIET AANGESLOTEN
RFID	VCC	3.3V
RFID	RST	D7
RFID	GND	GND
RFID	MISO	D12
RFID	MOSI	D11
RFID	SCK	D13
RFID	NSS	D8
RFID	IRQ	NIET AANGESLOTEN

4. Code uploaden

De volgende stap is om de code te uploaden naar de Arduino's.

1. Download hier (https://github.com/HANICA-MinorMulti/nj2017-iot-dwa-BurgersZoo1/tree/master/deliverables/compiled-code) de gecompileerde hex bestanden.
2. Installeer XLoader (http://www.hobbytronics.co.uk/arduino-xloader) of Arduino Uno Uploader Tool (http://rlangoy.github.io/Arduino-Uno-Uploader-Tool/)
3. Gebruik deze tool om de gedownloade hex files naar de juiste Arduino te uploaden.

Een alternatief is het handmatig compileren en uploaden van de code. Lees hiervoor de uitleg in het hoofdstuk ontwikkelhandleiding.

5. SD kaart

De poot maakt gebruik van een MicroSD kaart waarop de weetjes en en dierengeluiden op staan. Deze MicroSD kaart kan maximaal 32GB zijn en moet geformateerd zijn in FAT32.

Op de MicroSD kaart moeten de audio bestanden volgens de volgende naamgeving:

- Een diergeluid in de root: dier.wav
- Weetjes genummerd, beginnend vanaf 0.
  Dus 0.wav, 1.wav, 2.wav etc.

De inhoud van de MicroSD kaart zal er als volgt uit zien:

```
/dier.wav
/0.wav
/1.wav
/2.wav
/3.wav
/[0...+].wav
```

Wanneer een wav bestandje op de MicroSD gezet moet worden, zal deze geconverteerd worden naar de goede settings.
** todo: uitleggen hoe de wav bestanden geconverteerd moeten worden, dit komt in week 7 & 8. **

6. Aanzetten

Nadat alle bovenstaande stappen doorlopen zijn kunnen de twee Arduino's op stroom worden gezet. Vervolgens gaan de twee Arduino's initialiseren. Om te kunnen zien of onderdelen goed worden geinitialiseerd moet worden gekeken naar de statuslampjes. Per Arduino is hieronder te vinden wat de statuslampjes betekeken.

Maduino

Als alles goed gaat dan gaan er 3 lampjes branden zodra de Maduino wordt opgestart. Wanneer het initialiseren compleet is gaan de drie lampjes uit. Verder zijn er een aantal abnormale lampcodes. Deze zijn hieronder uitgewerkt.

Gedrag	Betekenis
Alle lampjes blijven aan staan.	Bezig met opstart sequence. Zoalang de Arduino nog niet verbonden is met een gateway worden deze lampjes laten zien. Als dit lang zo blijft staan dan is er waarschijnlijk iets mis met de NRF24L01+ óf kan de gateway niet worden gevonden.
Geel lampje continu aan	Er kan niet worden verbonden met de auduino. Er is iets mis met het verzenden naar de Auduino met I2C.
Geel lampje kort knipper	Er wordt verzonden naar de Auduino. Wanneer je een hele korte gele knipper ziet dan wordt er een signiaaltje verzonden naar de Auduino.
Rood lampje 0.5 seconde knipper	De gescande pas kon niet worden geauthenticeerd.
Rood lampje 1 seconde knipper	De gescande heeft niet de inhoud `Burgers Zoo`.
Groen lampje 1 seconde aan	Er is een valide pas gescant.

Auduino

De Auduino laat geen led branden als er hij niks aan het doen is. De volgende statussen kunnen worden afgelezen van de blauwe status led:

Gedrag	Betekenis
Blauwe led knippert snel	De SD kaart lezer kon niet worden geinitialiseerd.

Blauwe led brand continu Er wordt een geluidje afgespeeld

7. Pasjes

De poot accepteert alleen nfc pasjes met een frequentie van 13.56 MHz. Elk pasje moet een uniek identificatienummer hebben. Verder moet de op de pas de tekst `Burgers' Zoo` staan. Pasjes die niet aan deze eisen voldoen worden niet geaccepteerd door de poot.

Ontwikkelhandleiding

Om verder aan het systeem te kunnen ontwikkelen is dit hoofdstuk in het leven geroepen. Hieronder wordt uitgelegd hoe de ontwikkelomgevingen in te stellen en wat er nodig is om de code te kunnen begrijpen.

Ontwikkeling front-end applicaties

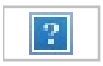
De front-end applicaties zijn opgebouwd met een React seed van Corey House (https://github.com/coryhouse), namelijk react-slingshot (https://github.com/coryhouse/react-slingshot). Corey heeft op zijn Github repository een uitgebreide uitleg (https://github.com/coryhouse/react-slingshot/blob/master/README.md#get-started) staan over het werken met zijn slingshot seed. De gekozen bundler voor dit project is Webpack (https://webpack.js.org/).

Redux

Het maken van een applicatie in React kan snel uit de hand lopen als je applicatie groter wordt. Om structuur in de applicatie
aan te brengen is gekozen om Redux te gebruiken.

Met Redux beheer je state op applicatieniveau. Redux bestaat
uit een aantal onderdelen maar het belangrijkste dat je moet onthouden zijn de volgende drie:

- **store**: opslagplaats van alle data als één groot object (POJO).
- **reducers**: pure functies die de app-state bewerken op basis van binnengekregen data uit zgn. action creators.
- **action creators**: functies die iets doen, bijvoorbeeld een API benaderen of iets uitrekenen. De uitkomst geven ze door aan de reducers. Dat doorgeven wordt '***dispatchen***' genoemd.



Folderindeling (package structuur)

De folderindeling hanteert de naamgevingen van Redux, zodat iedereen die weet wat Redux is meteen snapt waar files moeten komen te staan.

```
.
├── /build/                     # De folder voor gecompileerde output
├── /node_modules/              # 3rd-party libraries en utilities
├── /src/                       # The source code of the application
│   ├── /actions/               # Action creators zoals beschreven in Redux
│   ├── /components/            # React components die enkel UI logica bevatten
│   ├── /constants/             # Constantes die over meerdere files gebruikt
worden
│   ├── /containers/            # React components die toegang hebben tot de
Redux app state
│   ├── /reducers/              # Redux reducers
│   ├── /routes/                # Page/screen components met routing informatie
│   ├── /store/                 # Bevat configuratie die nodig is om de Redux
store op te bouwen
│   ├── /styles/                # Bevat styling files die over meerdere
componenten gebruikt wordt
│   ├── /index.ejs              # Template voor de index.html file, wordt door
webpack gebruikt
│   ├── /index.jsx              # Startup script, koppelt React aan de DOM
│   └── ...                     # Overige core files die door webpack gebruikt
worden.
├── /tools/                     # Build automation scripts en utilities die
webpack gebruikt.
├── package.json                # De lijst met 3rd party libraries en utilities
```

Tips

Op Linux en Mac OS X zit verder nog een limiet op het aantal bestanden / mappen waar een gebruiker tegelijk naar mag "luisteren" voor veranderingen. Om dat op te lossen moet je het volgende commando uitvoeren:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo
sysctl -p
```

Voor technische info klik hier (https://github.com/emcrisostomo/fswatch), voor sysctl uitleg klik (https://wiki.archlinux.org/index.php/sysctl) hier.

Loopback

Loopback functioneert als het hart van de backend. Met een paar commando's is er gemakkelijk een krachtige API opgezet.

Installatie

Via de commandline valt Loopback gemakkelijk te installeren. Met dit commando wordt Loopback global geïnstalleerd. Dit zorgt er voor dat je loopback ook kan gebruiken buiten je huidige project.

```
npm install -g loopback-cli
```

Loopback is een commandline tool, dus vereist niets anders dan je commandline tool om er mee aan de slag te kunnen.

Gebruik:

Loopback genereert een API op basis van vragen en antwoorden in je commandline tool. Loopback werkt via models om data op te slaan, en functioneert hier dus als middleware tussen je applicatie en je data opslag.
De versie de gebruikt wordt is [versie 3.x (http://loopback.io/doc/en/lb3/index.html)]

Maak een eerste app:

Hieronder staat een voorbeeld van de Loopback website.

```
$ lb          //dit is het basis commando om een nieuw project te starten
? What's the name of your application?    hello-world
? Enter name of the directory to contain the project:    hello-world

? Which version of LoopBack would you like to use? 3.x (current)
? What kind of application do you have in mind? hello-world (A project
containing a controller,
including a single vanilla Message and a single remote method)
...
I'm all done. Running npm install for you to install the required dependencies.
If this fails, try running the command yourself.
...
```

Wat je nu hebt gemaakt is een hello-world app. De optie voor een API server is: `api-server`. Je kunt navigeren door de pijltjestoetsen te gebruiken en op enter te drukken. Je ziet hier ook de vraag-antwoord aard van Loopback terug. Loopback vraagt simpel wat je wil en geeft een aantal opties. Zo kun je een gehele applicatie op te stellen, zonder zelf een hoop code te hoeven intikken.

Maak een API

Ga naar de map waarin je de API wil aanmaken.
Gebruik de Loopback [application generator (http://loopback.io/doc/en/lb3/Application-generator.html)] om het project aan te maken.

```
> lb
? What's the name of your application? (demo)    demo     //tussen haakjes
staat standaard de map waarin je nu bent. Als je op enter drukt wordt de waarde
tussen de haakjes gebruikt. Zo hoef je niet dingen over te typen.
? Which version of LoopBack would you like to use? (Use arrow keys)
  2.x (long term support
  3.x (current)   <--- pak deze
? What kind of application do you have in mind? (Use arrow keys)
> api-server (A LoopBack API server with local User auth)         <---
selecteer deze
  empty-server (An empty LoopBack API, without any configured models or
datasources)
  hello-world (A project containing a controller, including a single vanilla
Message and a single remote method)
  notes (A project containing a basic working example, including a memory
database)
```

Nu komt de echte Loopback magie. Er worden een aantal bestanden aangemaakt waarin de configuratie van je project staat. Zo wordt er ook een package.json bestand aangemaakt met alle dependencies die je zeker nodig hebt, ze worden ook gelijk geïnstalleerd.
Als Loopback klaar is zie je hetvolgende:

```
Next steps:
  Create a model in your app
  $ lb model

  Run the app
  $ node .
```

Loopback geeft hier dus al aan wat de volgende stappen zijn die je moet ondernemen.
Het is handiger als je, voordat je deze stappen volgt, eerst zorgt voor een dataopslag.

Data opslaan

Zoals hierboven al gezegd, loopback werkt via models. Dit betekent dat alle data op een logische manier opgeslagen wordt in modellen. Deze modellen kunnen relaties hebben om zo data uit te breiden.

Allereerst is er een vorm van dataopslag nodig, bijvoorbeeld een Mongo (https://www.mongodb.com/) database.
Daarvoor kun je de Loopback data source generator (http://loopback.io/doc/en/lb3/Data-source-generator.html) gebruiken.

```
> lb datasource
? Enter the datasource name:  mongo
? Select the connector for mongo: (Use arrow keys)
MongoDB (supported by Strongloop)  <--- scroll net zo lang totdat je deze tegen
komt.
? Connection string url to override other settings (eg:
mongodb://username:password@hostname:port/database):    //Wat loopback hier
aangeeft is dat je ook een volledige mongo connection string kan gebruiken in
plaats van alle waarden zelf invoeren later. In dit voorbeeld laten we deze
leeg om te laten zien hoe je de rest invult.
? host: localhost
? port: 27017   //dit is de default mongo port
? user:    //laten we leeg, want we hebben geen user authenticatie nodig
? password:   //idem dito
? database: demo
? Install loopback-connector-mongodb@^1.4 (Y/n)   //default antwoord is hier
Yes, dit wil je zeker doen omdat
```

Nu is er een nieuwe Mongo database aangemaakt. Er is een connector geïnstalleerd via npm die er voor zorgt dat je applicatie kan praten met de mongo database. De database is geconfigureerd via het pad en poort dat je opgegeven hebt, en de databse heeft een naam gekeregen. Optioneel is user authenticatie, dat is hier weggelaten.

Nieuwe modellen aanmaken

Om een nieuw model te maken kun je de loopback model generator (http://loopback.io/doc/en/lb3/Model-generator.html) gebruiken.

```
lb model
? Enter the model name: Persoon
? Select the datasource to attach Persoon to:
  db (memory)
  mongo (mongodb)   <-- dit is de datasource die we net aangemaakt hebben.
Selecteer deze.
? Select the model's base class (Use arrow keys)
  Model
  PersistedModel   <-- we kiezen hiervoor. Een persistedModel wordt gebruikt
wanneer je CRUD acties wil uitvoeren met data
? Expose Person via the REST API? (Y/n) Y  <-- wil je er via de REST API bij
kunnen of is het alleen voor intern gebruik?
? Custom plural form (used to build REST URL):  personen  <-- geef de naam van
de REST URL op. Dit is in het meervoud. In het nederlands raad ik aan om dit
altijd in te voeren, anders krijg je persoon + s als antwoord.
? Common model or server only? (User arrow keys)  common // is het voor client
en server of alleen voor de server?
Let's add some Persoon properties now.

Enter an empty property name when done.   //dit zijn de velden die je in je
model wil hebben.
? Property name: naam
? Property type: (Use arrow keys)  string
? Required? (y/N)  Y
? Default value[leave blank for none]:


// Dit gaat door totdat je een property leeg laat. Laten we de volgende waarden
ook toevoegen:
leeftijd: number
geslacht: string

// maak op dezelfde manier ook een hobby model aan met als property naam:
string
```

Relaties

Om een relatie aan te maken kun je de relation generator (http://loopback.io/doc/en/lb3/Relation-generator.html) van Loopback gebruiken.

```
lb relation
? Select the model to create the relationship from:  Persoon
? Relation type: has one
? Choose a model to create the realationship with: hobby
? Enter the property name for the realtion: hobby
? Optionally enter a custom foreign key:
? Allow the relatoin to be nested in REST APIs: No
? Disable the relation from being included: No
```

Er is nu een relatie gelegd tussen een persoon en een hobby. Dit zal later terug te vinden zijn in de API explorer
Om er achter te komen wat de relatie types exact doen, raadpleeg dan de [Loopback documentatie over relaties (http://loopback.io/doc/en/lb3/Creating-model-relations.html)](http://loopback.io/doc/en/lb3/Creating-model-relations.html).

Modellen aanpassen

Loopback houdt zijn modellen bij in JSON bestanden. Wanneer je een model aanmaakt bepaal je in feite ook waar het model neer wordt gezet: in de server of in de common map. Daar, in het mapje models, vind je voor elk model een .json en een .js bestand. Deze bestanden samen maken het model dat Loopback gebruikt. Je kunt hier handmatig ook dingen wijzigen zonder de commandline tool te gebruiken.
Hieronder vind je een voorbeld van het persoon model dat we net aangemaakt hebben.

```
{
  "name": "Persoon",
  "plural": "personen",
  "base": "PersistedModel",
  "idInjection": true,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "naam": {
      "type": "string",
      "required": true
    },
    "leeftijd": {
      "type": "number",
      "required": true
    },
    "geslacht": {
      "type": "string"
    }
  },
  "validations": [],
  "relations": {
    "hobby": {
      "type": "hasOne",
      "model": "hobby",
      "foreignKey": ""
    }
  },
  "acls": [],
  "methods": {}
}
```

Eigen endpoints toevoegen

Loopback biedt de mogelijkheid om eigen endpoints of logica toe te voegen. Dit kan op model niveau, door dit toe te voegen aan het javascript bestand van de model (models/persoon.js). Verder is er nog een map boot in de map server. Daar kunnen allerlei scripts geplaatst worden die gedraaid moeten worden wanneer de API opgezet wordt.

API explorer

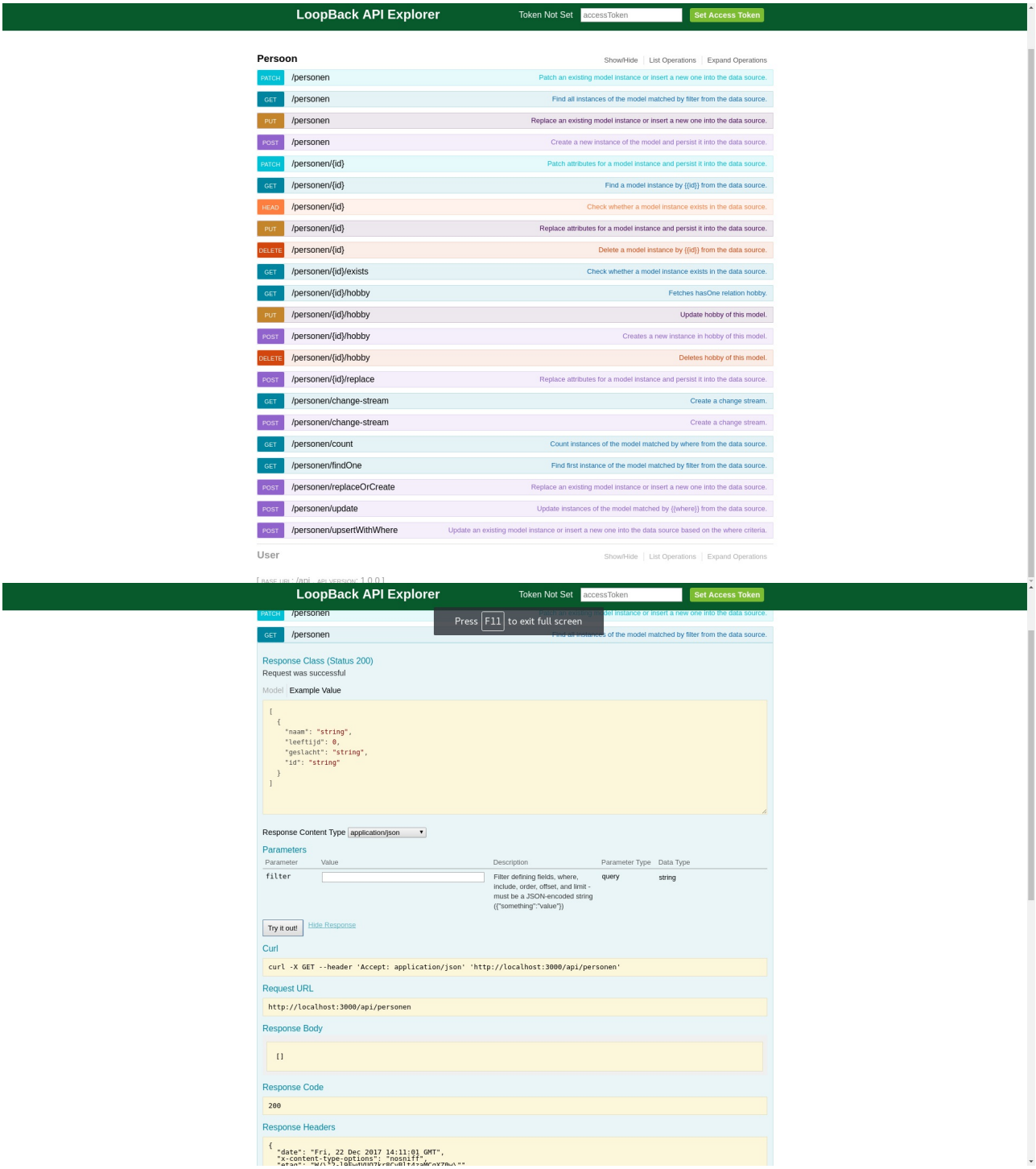
Loopback heeft een ingebouwde API explorer om te kijken hoe je API er uit ziet, maar ook om hier requests te testen en uit te voeren. De requests die je hierin uitvoert zijn échte requests, dus zonder mock data.
Om dit voor elkaar te krijgen moet je het project runnen en vervolgens naar het juiste adres gaan.
In de projectmap:

```
node .

Browse your REST API at http://0.0.0.0:3000/explorer  <-- dit geeft loopback
terug als antwoord. In feite een beschrijving van waar je alles kan beginnen.
Web server listening at: http://0.0.0.0:3000/
```

Om de API explorer te gebruiken moet je dus ook gaan naar http://localhost:3000/explorer .
Je zult hier zien dat er veel meer request methoden aangemaakt zijn per model dan je zou verwachten. Deze werken echter allemaal.

Voorbeeld:



Filters

Een van de meest gebruikte filters in Loopback is de [include filter (https://loopback.io/doc/en/lb3/Include-filter.html)](https://loopback.io/doc/en/lb3/Include-filter.html), deze wordt gebruikt om verschillende domeinmodellen te koppelen.

Een include filter zorgt ervoor dat je documenten terugkrijgt met de eventuele subdocumenten daarin opgenomen.

Neem bijv het model "Bestelling", dit heeft de volgende velden:

- bestellingsnummer
- artikelen

- klantnaam

▶ Klik hier om het JSON model te zien

De klantnaam geeft daar niet veel informatie over de klant, de klant heeft namelijk de volgende velden:

- klantnaam
- leeftijd
- email

Om deze informatie **toch** te krijgen moeten we een include filter gebruiken.

Het resultaat wordt dan:

- bestellingsnummer
- artikelen
- klant
    - klantnaam
    - leeftijd
    - email

▶ Klik hier om het JSON model te zien

Naast de include filter zijn er nog veel meer filters, deze zijn echter in veel mindere maten gebruikt in dit project. Om te zien hoe de rest van de filters werken verwijzen we door naar [de documentatie (https://loopback.io/doc/en/lb3/Querying-data.html)]

Swagger Generator

Loopback kan zelfs zijn eigen API documenteren. Hiervoor is de [Loopback API definition generator (https://loopback.io/doc/en/lb3/API-definition-generator.html)] (OpenAPI spec) bedoeld.
Dit valt op de volgende manier te gebruiken:

```
lb export-api-def
```

Gebruik de `-o` tag om een pad aan te geven waar je de definitie op wil slaan, anders wordt het alleen in de commandline getoond.

Ontwikkeling Gateway

Eisen

- Java 8 SDK [installatie (http://bfy.tw/FhgO)]
- Gradle 4.4 [installatie (http://bfy.tw/FhgK)]
- RXTX, zie kopje "Seriële communicatie"

Werking Gradle

Gradle is een build tool, te vergelijken met Mavan, die alle dependencies beheerd van de applicatie. Hiermee worden third-party dependencies gedownload en kan het project zelf gecompileerd worden.

De basis taken die uitgevoerd kunnen worden:

- `gradle clean`: verwijderd de build directory
- `gradle build`: bouwt het volledige project
- `gradle test`: voert alle tests uit in het project
- `gradle jar`: bouwt een jar die uitgevoerd kan wroden om de applicatie te starten.
- `gradle clean build test jar`: om alle bovenstaande taken in één keer uit te voeren

Folder structiuur

De applicatie bestaat uit een vrij standaard Java folder structure.

```
|── /build/
|── /gradle/
|── /out/
|── /src/
|   |── /main/
|   |   |── /java/
|   |   |   |── /nl/
|   |   |   |   |── /han/
|   |   |   |   |   |── /han/
|   |   |   |   |   |   |── /Application.java # Start punt van de applicatie
|   |   |   |   |   |   |── /mysensors/ # MySensors parsers en communicatie
seriële poort
|   |   |   |   |   |   |── /backend/ # connectie naar backend
|   |   |   |   |   |   |── /gateway/ # controllers van Spark
|   |   |   |── /gateway.properties # Settings zoals poort nummer
|   |   |── /resources/
|   |── /test/* # Bevat alle tests van de applicatie
|── /readme.md # Eigenschappen applicatie beschreven
|── /build.gradle # Build file, bevat dependencies etc.
```

Seriële communicatie

Voor het gebruik van de gateway moet er een library geïnstalleerd zijn op de pc. Het gaat om de RXTX library. Deze zorgt voor de seriële communicatie.

De library is te vinden via de volgende links:

- Linux: ~~http://rxtx.qbang.org (http://rxtx.qbang.org/wiki/index.php/Download)~~ Het is alleen nodig om het volgende commando uit te voeren: `sudo apt-get install librxtx-java`
- Windows 64 bit: http://fizzed.com (http://fizzed.com/oss/rxtx-for-java)

Zonder de instalatie van deze library zal er bij het bouwen van het project fouten optreden.

Uitvoeren op de Raspberry Pi 3

Bij het uitvoeren van de jar op de Raspberry Pi 3 moet het volgende gebruikt worden:
`java -Djava.library.path=/usr/lib/jni -jar <GATEWAY.jar>`
Het gedeelte van `-Djava.library.path=/usr/lib/jni` moet voor de `-jar` staan

Afspraken MySensors protecol

- V_VAR1 wordt gebruikt om van de node naar de gateway zijn poot id te presenteren.
- V_VAR2 wordt gebruikt om een pas id van een node naar de gateway te sturen.
- V_VAR3 wordt gebruikt om vanaf de gateway naar een node een poot id te sturen.
- V_VAR4 is gereserveerd voor het versturen van audio files.
- V_VAR5 is gereserveerd voor het configureren van de Arduino, zoals het resetten van EEPROM of resetten van de Arduino zelf.
- V_TEMP wordt gebruikt om de temperatuur data door te sturen.
- V_HUM wordt gebruik om de humidity te versturen

Ontwikkeling Poot

De code van de Poot is gebouwd met behulp van het Arduino platform en een aantal libraries. Een gedetailleerd ontwerp van de code is te vinden in het technisch ontwerp.

Ontwikkelomgeving

Om te kunnen ontwikkelen aan de poot is PlatformIO (http://platformio.org/) vereist. Installeer PlatformIO volgens de officiele handleiding (http://platformio.org/platformio-ide). Vervolgens kan het project geopend worden in de ide. Dan kan de code gecompileerd en geupload worden. Er is gekozen voor Atom over de standaard arduino IDE omdat deze IDE automatisch libraries download én er een mogelijkheid is voor live-code feedback. Zie dit (https://github.com/HANICA-MinorMulti/nj2017-iot-dwa-BurgersZoo1/tree/master/documentatie/onderzoeken/arduino-ide) onderzoekje voor details over deze keuze.

Libraries

Voor het ontwikkelen van de poot worden de volgende libraries gebruikt. Deze libraries worden automatisch gedownload door PlatformIO wanneer er voor de eerste keer gebouwd wordt.

Audino

- SPI, versie:1.2.1
- Wire, versie: 1.0.0
- SD, versie 1.2.1
- https://github.com/TMRh20/TMRpcm.git#v1.0, versie 1.0
- https://github.com/jbeynon/sdfatlib.git, versie commit: 2ee66d98e28758783400617f477da37d4379d47d

Maduino

- SPI, versie:1.2.1
- Wire, versie: 1.0.0
- MFRC522, versie 1.3.6
- MySensors, 2.1.1

Bijlagen

Repo commando's

In deze bijlage wordt uitgelegd wat de "repo commando's" inhouden.
Een "repo commando" is één van de scripts die in de package.json staan.

Dat levert de volgende items op:

Commando	resultaat	Notities
start	Start de apps in development modus	
build	Start de apps in productie modus (en bouwt productie files)	
build-docker	Bouwt zowel de dev als de productie docker images.	
build-docker-dev	Bouwt de dev docker image.	
build-docker-prod	Bouwt de productie docker image	
compile-deliverables	Bouwt alle documentatie	
compile-images	Verzamelt alle images in de deliverables/images map zodat ze gebruikt kunnen worden in de documentatie	! werkt niet op Windows
compile-pva	Bouwt het Plan van Aanpak	Wordt gebouwt in de deliverables map
compile-fo	Bouwt het Functioneel ontwerp	Wordt gebouwt in de deliverables map
compile-to	Bouwt het technisch ontwerp	Wordt gebouwt in de deliverables map
compile-testplan	Bouwt het testplan	Wordt gebouwt in de deliverables map
compile-opleverdocumentatie	Bouwt de opleverdocumentatie	Wordt gebouwt in de deliverables map
generate-pdfs	Zet alle gebouwde bestanden om naar een .pdf	Wordt gebouwt in de deliverables/pdfs map
copy-endpoint-prod	Dit kopieërd alle development instellingen naar de apps	! werkt niet op Windows
copy-endpoint-dev	Dit kopieërd alle development instellingen naar de apps	! werkt niet op Windows
postinstall	Dit script draait NA een npm install en zal de "build-docker" taak uitvoeren	

Software lijst

In deze lijst vindt je voor de meeste software links naar installatiehandleidingen.

Product	Windows	Mac OS
Mongo	link (https://docs.mongodb.com/manual/installation/)	link (https://docs.mongodb.com
Docker	link (https://docs.docker.com/toolbox/toolbox_install_windows/)	link (https://docs.docker.co mac/install/)
docker-compose	link (https://docs.docker.com/compose/install/)	link (https://docs.docker.co
node	link (http://blog.teamtreehouse.com/install-node-js-npm-windows)	link (http://nodesource.com nodejs-tutorial-mac-os-x/)
npm	link (https://www.npmjs.com/package/npm)	link (https://www.npmjs.cor

bash	link (https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/)	-