Systeem documentatie

NAO PAD team 7
Stefan de kraker
Jazzley Louiseville
Renze Goossens
Alperen Elcik

Mei 2020

Contents

1 Introductie

In deze Systeem documentatie bespreken we ons product. We hebben op basis van de NAO een entertainment systeem gemaakt, waarmee ouderen vermaakt kunnen worden. Dit op basis van de NAO. Daarnaast hebben we een externe tool gebouwd, zodat de ouderen makkelijk kunnen interacteren met de robot. We doorlopen in dit document alle features van ons product, en bespreken we hoe het externe product tot stand is gekomen.

2 Functionele beschrijving

De NAO heeft een aantal functies die hij kan uitvoeren. Al deze functies zijn verwerkt in een website die op de Raspberry Pi draait. Die website draait op een webserver gemaakt in Flask. De Flask webserver zorgt ervoor dat alle Routes naar de juiste pagina's leiden. Alle in python geprogrammeerde functies zijn gekoppeld aan deze website. Om te kunnen interacteren met de NAO hebben wij een ouderwetse druk telefoon aangepast, alle knoppen worden nu uitgelezen door de Raspberry pi. Aan de Raspberry pi zit een scherm gekoppeld, zodat de website te zien en te bedienen valt. Door middel van de telefoon en door de touchscreen valt te navigeren door de webapplicatie.

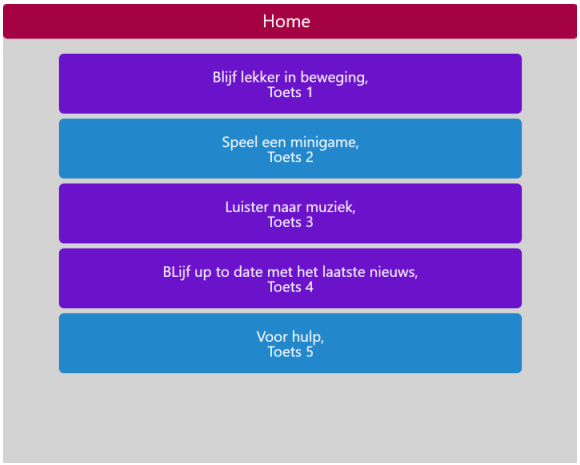

Figure 1: thuis pagina

2.1 beweeg pagina

Op deze pagina zijn 3 afbeeldingen te zien. Als je op deze afbeeldingen drukt, dan gaat de NAO een aantal bewegingen demonstreren. Dit hebben we kunnen realiseren met de ALAnimationPlayer service. Door een switch case te maken in python en elke knop aan een nummer te verbinden hebben we een soort van menu gemaakt. Dus 3 verschillende dansjes. Die dansjes bestaan uit verschillende animaties die na elkaar spelen. Tussendoor wordt met de ALTextToSpeech service instructies gegeven om de gebruiker aan te moedigen. Hieronder in de afbeelding is te zien hoe de pagina er uit ziet.

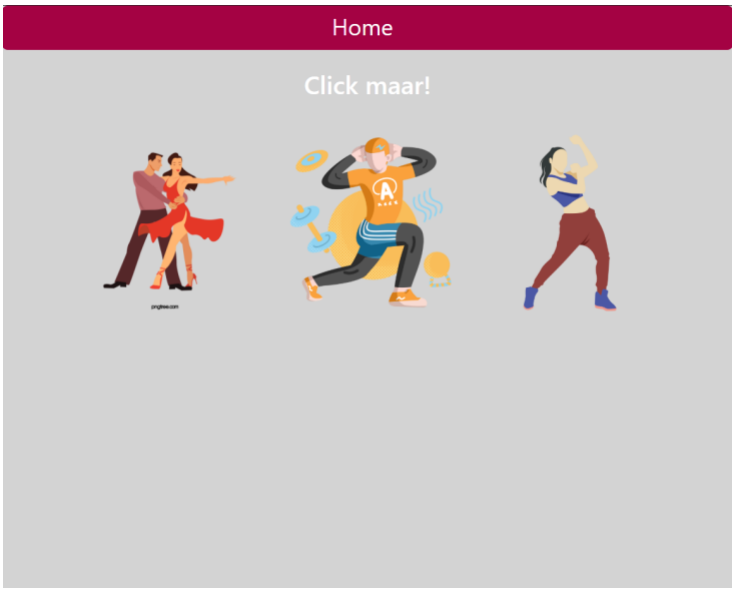

Figure 2: Dit is de beweeg pagina

2.2 Minigames pagina

Op deze pagina kan je kiezen uit twee verschillende spelletjes. Zo kan je sudoku spelen en rekensommetjes oplossen. Op deze pagina wordt je doorverwezen naar één van de twee spelletjes. Je kan tevens ook terug naar de thuispagina navigeren. In het onderstaande figuur is de pagina te zien.

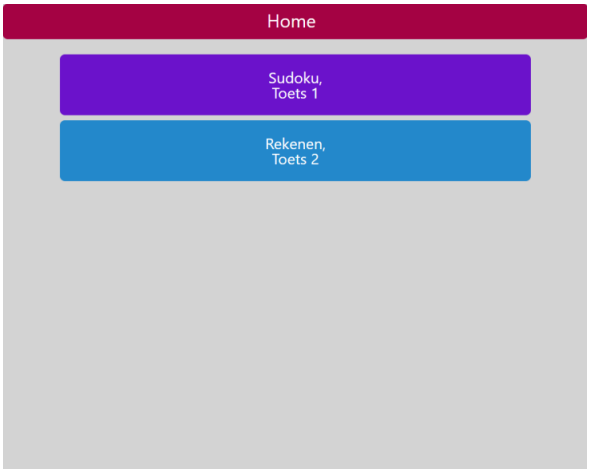

Figure 3: Dit is de minigames pagina

2.2.1 sudoku

Voor het maken van de sudoku pagina hebben wij een sudoku API gebruikt. Op deze manier hebben we niet het hele spel opnieuw moeten uitvinden. Deze API komt van een open scoure Github account, de API is geschreven in Java Script, en kon zo in worden ingezet in de cliënt zijde van de website. Zoals te zien is in de onderstaande afbeelding is dat er twee knoppen zijn. Die knoppen kunnen worden gebruikt om je te helpen bij het spelen van de sudoku.

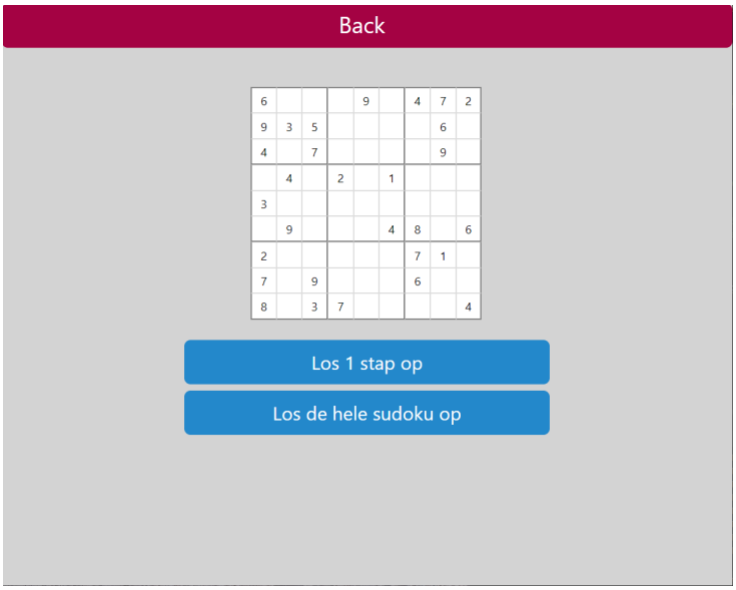

Figure 4: Dit is de sudoku pagina

2.2.2 rekensommetjes

Bij de pagina met rekensommetjes heb je keuze uit verschillende moeilijkheidsgraden. Zodra en gekozen is voor een bepaalde graad, dan wordt je doorverwezen naar een andere pagina met sommetjes. De NAO leest de som voor, en de som is te zien op de pagina. Het invullen van de som verloop volledig via de telefoon. de toetsen 0-9 kunnen gebruikt worden om het getal in te vullen. Controleren kan door op "*" te drukken, ook wel enter.

Figure 5: keuzemenu van moeilijkheidsgraad

Hieronder is een voorbeeld van niveau 3

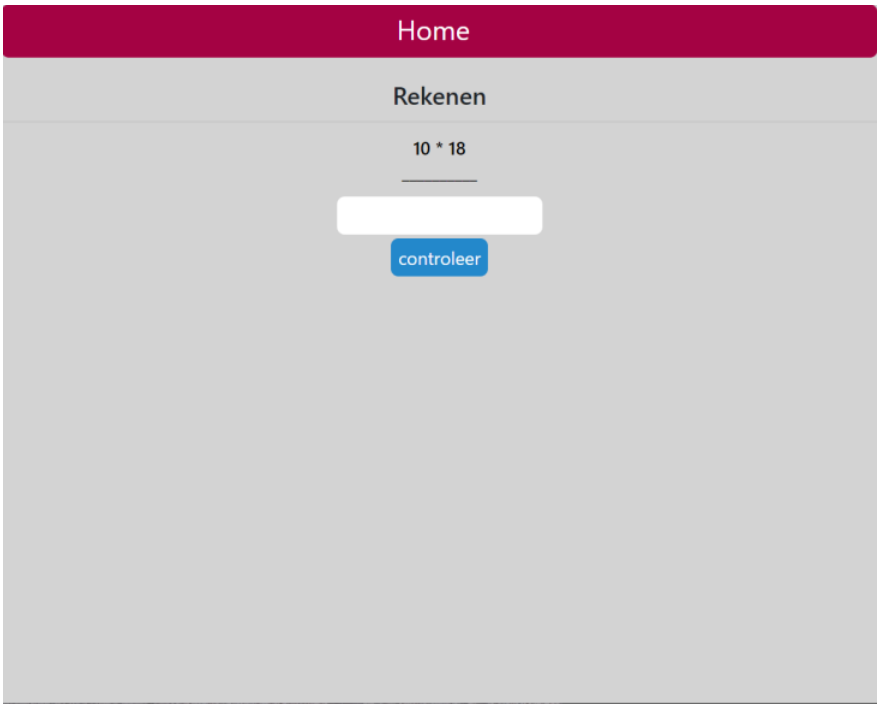

Figure 6: Niveau 3 van de rekensommen.

2.3 muziek pagina

Bij de muziek pagina heb je keuze uit vijf verschillende liedjes. Door gebruik te maken van de `"AlAudioPlayer"` service kan de NAO muziek afspelen. Door een soort 'switch case' te maken in python kan je via de telefoon kiezen uit de nummers. De nummers staan in een .mp3 file op de NAO. Alle nummers zijn nostalgische nummers om de oudere gebruikers te vermaken. Aan het begin van elk nummer zal de NAO gaan staan en zwaaien. Daarvoor is de `"ALRobotPosture"` service nodig. Met deze service kan je de NAO een positie aan laten nemen. Als de NAO muziek gaat spelen wil je natuurlijk wel dat hij staat.

2.4 nieuws pagina

Voor het ophalen van het nieuws maken we gebruik van de NewsAPI. Deze API zorgt ervoor dat we alle "headlines" gemakkelijk kunnen ophalen in een JSON formaat. Om deze informatie op te vragen wordt er gebruik gemaakt van een (Cross-Origin Resource Sharing) CORS proxy. Deze proxy zorgt ervoor dat de API HTTP aanvragen kan doen bij de verschillende website waar de API zijn informatie ophaalt. Vervolgens wordt de informatie van deze JSON weer gebruikt om de pagina te vullen met informatie.
Om de hoeveel informatie te bepreken op de pagina zullen er in totaal 3 artikelen

op de pagina staan. De API haalt veel ver dan 3 artikelen op, daarom zal er per keer 3 keer een willekeurig artikel worden gekozen.

2.5 support pagina

De support pagina is eigenlijk een simpele pagina met tekst er op, voor als de gebruiker niet precies weet hoe het apparaat werkt.

3 Ontwerp externe hardware

3.1 Telefoon

De keuze is gevallen op het gebruik van een oude telefoon. Dit is gedaan omdat dit een nostalgisch gevoel geeft aan de de doelgroep. Om dit goed te kunnen maken is er gekozen een 3D ontwerp te maken in Autodesk Inventor 2020. alle files zijn ook te vinden in de Gitlab in de map Design files. Ook zijn er `.STL` en `.DXF` bestanden aanwezig voor het fabriceren van de onderdelen.

4 documentatie code

Als andere software developers onze code willen gebruiken kunnen zij hier terecht. Hieronder staat hoe we gewerkt hebben en waar op gelet moet worden bij het overnemen van ons werk. Het is opgedeeld in verschillende stukken zodat alles helder blijft. Als er nog onduidelijkheden zijn scroll dan naar hoofdstuk 5

4.1 UML

Hieronder is een afbeelding van de UML die gebruikt is in dit project. In de UML staan alle functies die we hebben geïmplementeerd in de NAO. Voor een groter overzicht van de UML kan je in de Gitlab vinden.

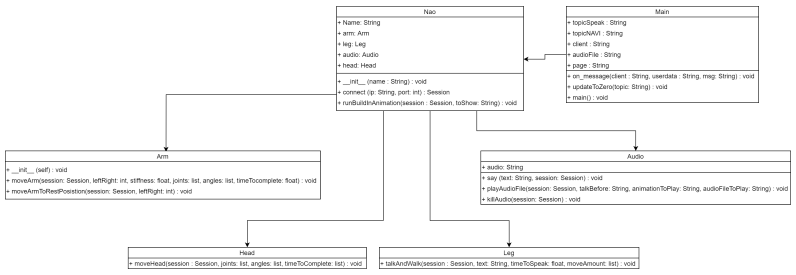

Figure 7: Ontwerp van de UML

4.2 MQTT

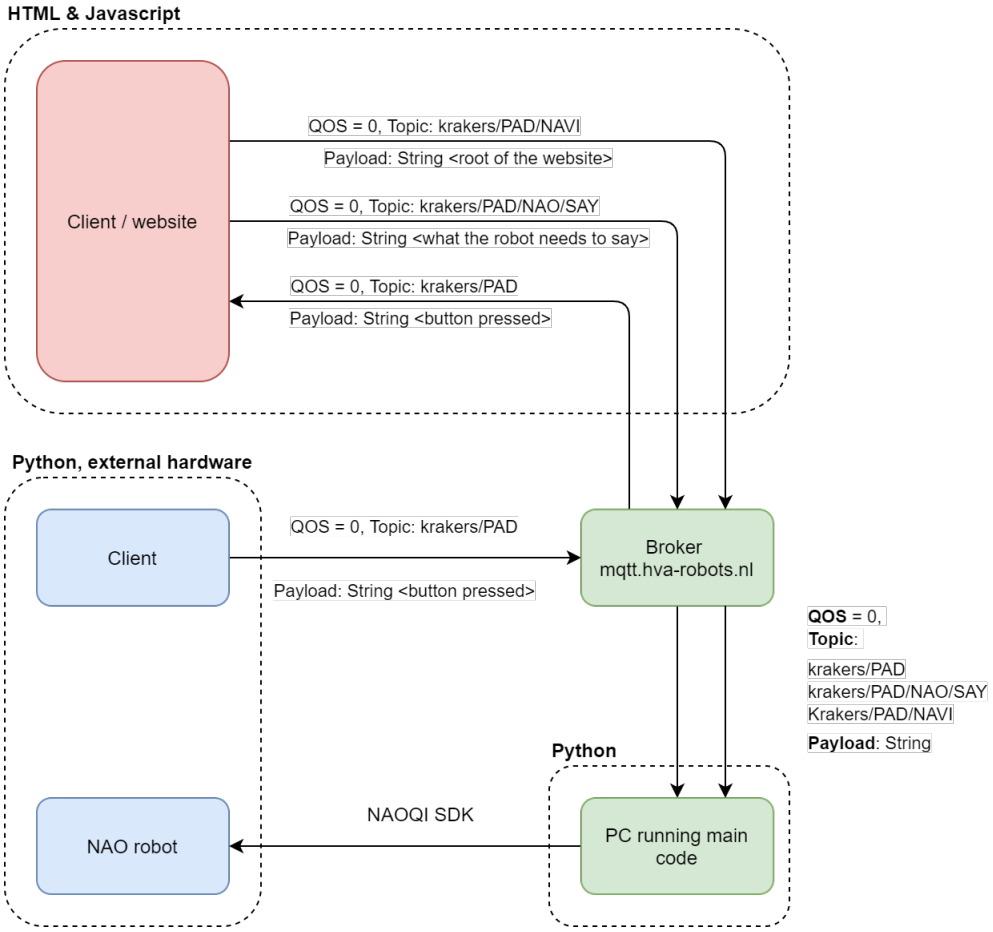

Figure 8: overzicht van de MQTT communicatie tussen de verschillende elementen

Om de toets slagen van de telefoon op een ander apparaat te krijgen wordt er gebruik gemaakt van MQTT. Het voordeel hiervan is dat het een cliënt server concept is. Zo kunnen er dus gemakkelijk meerdere applicaties en apparaten mee verbinden. De server is van het HvA en we hoeven dus daar niet over na te denken. We zorgen ervoor dat we ons aanmelden bij de juiste topic (gesprek). Vervolgens maken we gebruik van de call back functies binnen de Eclipse Paho MQTT bibliotheek. Als er dan een nieuw bericht binnen komt kunnen we kijken of dit van het juiste topic komt, en vervolgens dan code uitvoeren op basis van het bericht. Hier onder is een deel van de code die op de cliënt zijde van de website draait.

De functie `client.onMessageArrived = function (message)` zorgt er voor op het moment dat er een bericht binnen komt dat de website daarmee iets zal doen. In dit geval zal er op basis van het bericht naar een andere pagina worden genavigeerd.

```
// Connect
client.connect({userName: ID, password: passWord, onSuccess:
    onConnect});

// Clear the old status, just to be sure
function onConnect() {
    client.subscribe(<topic to subscribe to>);
    sendMessage(<message to send>, <topic to send to>);
}
// Send a message
function sendMessage(toSend, whereToSend) {
    const message = new Paho.MQTT.Message(toSend);
    message.destinationName = whereToSend;
    client.send(message);
    }
// If there is a message, redirect.
client.onMessageArrived = function (message) {
    console.log("Message Arrived: " + message.payloadString);
    switch (message.payloadString) {
        case '1':
            window.location.href = '/beweeg';
            break;
        case '2':
            window.location.href = '/minigames';
            break;
        case '3':
            window.location.href = '/muziek';
            break;
```

Listing 1: Java script voorbeeld MQTT versturen en ontvangen

4.3 De website

De website wordt gehost op het platform `Oege`. De website maakt gebruik van verschillende connecties, libraries en frameworks. Waaronder: `Mqtt`(Voor de connectie met de robot) `Bootstrap`(voor de HTML en CSS styling) `Flask`(Voor het opstarten van de website/server) Er wordt veel gedaan op de frontend van de website, door gebruik te maken van javascript denk hieraan het sudoku spelletje dat geïmplementeerd is op de website. De connecties tussen de verschillende webpagina's worden geregeld op de backend door gebruik te maken van flask(een python library). De website maakt gebruik van 2 API's waaronder een sudoku api en een news api. Zodat al eerder geschreven code niet herschreven hoeft te worden, maar zeer makkelijk geïmplementeerd kan worden.

De website is geschreven met behulp van de markup-taal `HTML5, PYTHON, Javascript` en voor de opmaak/styling hebben wij `CSS` gebruikt. De website dient altijd

draaiend te zijn zodat de gebruiker er altijd bij kan. Er is geen gebruik gemaakt van een log in systeem, want persoonlijke data wordt namelijk niet onthouden. Waardoor er ook weinig security risico's zijn voor de gebruiker. De website kan volledig bestuurd worden door gebruik te maken van de bakelieten telefoon geleverd met het product. De website wordt weergegeven op een Raspberry pi scherm van 7' met een resolutie van 1200x800.

5 Configuratie

5.1 Java Script

Om er voor te zorgen dat het project zal werken is het belangrijk dat er twee configuratie bestanden aanwezig zijn. Deze staan niet in de Gitlab, dit heeft te maken dat hier wachtwoorden en andere gevoelige informatie in komt te staan die beter niet met iedereen gedeeld moet worden.
Voor de cliënt zijde van de website is er het volgende configuratie bestand nodig:

```
1  const config = function () {
2      const configMqtt = {
3          HOST_MQTT_BROKER: "ip naar broker",
4          MQTT_ID: "gebruikersnaam",
5          MQTT_PASSWORD: "wachtwoord",
6          MQTT_TOPIC_PHONE: "*/PAD",
7          MQTT_TOPIC_PY: "*/PAD/NAVI",
8          MQTT_TOPIC_SAY: "*/PAD/NAO/SAY"
9      };
10     return configMqtt;
11 }
```

Listing 2: configuratie Java script

Dit bestand moet op de volgende plek aanwezig zijn:
`C:\folder\folderOfGitlab\Website\webapp\static\JS`
en moet als volgt worden genoemd : `config.js`

5.2 Python

Ook voor python is er ook een configuratie bestand nodig. Hier staat de zelf MQTT zaken in als voor Java Script plus zaken die belangrijk zijn voor de robot.

```
# coding=utf-8
MQTT = {
    "CLIENT": "PC",
    "BROKER": "ip naar broker",
    "TOPIC_SPEAK": "*/PAD/NAO/SAY",
    "TOPIC_NAVI": "*/PAD/NAVI",
    "USERNAME": "gebruikersnaam",
    "PASSWORD": "wachtwoord"
}

NAO = {
    "NAME": "naam van de robot",
    "IP_ADRESS": "ip adres naar de robot",  # 127.0.0.1
    "PORT": <portnummer>  # 9559
}

DICTIONARY = [
    # Home | index 0
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Bweging | index 1
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Games | index 2
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Sudoku | index 3
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Rekenen | index 4
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Rekenen N1 | index 5
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Rekenen N3 | index 6
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Muziek | index 7
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Nieuws | index 8
    ["Ziet hier zinnen in voor de robot om uit te spreken"],
    # Hulp | index 9
    ["Ziet hier zinnen in voor de robot om uit te spreken"]]
```

Listing 3: configuratie Python

Dit bestand moet op de volgende plek aanwezig zijn:
`C:\folder\folderOfGitlab\Robot\`
en moet als volgt worden genoemd : `configRobot.py`

6 De NAO

6.1 externe info

Alle code die geschreven is staat in:
https://gitlab.fdmci.hva.nl/krakers/pad-project-team-3
verder zijn er nog handige links waar wij onze info vandaan hebben gehaald.

- https://flask.palletsprojects.com/en/1.1.x/
- https://www.eclipse.org/paho/clients/js/
- https://www.eclipse.org/paho/clients/python/
- https://newsapi.org/
- http://doc.aldebaran.com/2-8/index$_{d}ev_{g}uide.html$