R3.5 Het Hipper sensorsysteem

Ahmed Nait Aicha, Sven Haitjema, Pascal Wiggers

1. Inleiding

Het Hippersysteem geeft een revalidant en zijn of haar therapeuten inzicht in het beweeggedrag van de revalidant. Het beweeggedrag wordt geregistreerd met behulp van een draagbare PAM sensor, die op basis van accelerometerdata bepaalt hoeveel minuten per dag iemand beweegt. Daarnaast worden BeNext PIR-bewegingssensoren en contactsensoren in het huis van de revalidant geplaatst, waarmee een beeld kan worden verkregen van het dagpatroon van de revalidant.

De verzamelde data kan door de revalidant en therapeuten worden ingezien via een beveiligde webapplicatie.

2. Architectuur

Onderstaand figuur geeft een overzicht van de hardware en software architectuur van het Hippersysteem.

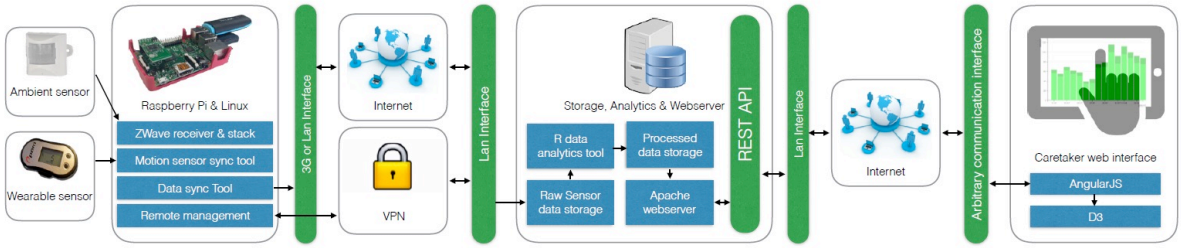

Fig 1. Overzicht van de architectuur van het Hippersysteem.

2.1 Hardware

De hardware bestaat uit een Hipperbox en een aantal sensoren: de draagbare PAM-sensor en de omgevingssensoren. De hipperbox is een Raspberry Pi met een Zwave shield voor de communicatie met de omgevingssensoren, een bluetooth adapter voor communcatie met de PAM sensor en een 4G dongle om verbinding met de server te maken om periodiek (dagelijks) data te versturen van de Hipperbox naar de server.

De door de PAM-sensor verzamelde data wordt op de PAM zelf opgeslagen en naar de Hipperbox verzonden zodra de patiënt in de buurt van deze box is. De PAM kan 30 dagen data opslaan zonder te synchroniseren. De data van de omgevingssensoren wordt via ZWave verstuurd naar de Hipperbox en wordt dus niet opgeslagen op de sensor zelf.

2.2 software

De Hipperbox verzendt de verzamelde sensor data eens per dag via 3G naar een beveiligde server. Om dit mogelijk te maken zijn op de hipperbox een aantal scripts en services geïnstalleerd. Daarnaast zijn een aantal scripts gemaakt om er voor te zorgen dat de hipperbox in de lucht blijft. Een overzicht van deze scripts is te vinden in het onderstaand tabel.

System part	Part subsection	Script name	Script function	Script caller	Script interval	Script location
PAM	Driver Compilation					
		pam.cpp	pam driver source	none	none	/home/pi/pam/pam_driver/
		pam.h	pam driver interface	none	none	/home/pi/pam/pam_driver/
		makepam.sh	pam driver make script	user	none	/home/pi/pam/pam_driver/
	Driver Execution					
		pam	pam driver executable	pamdriver	none	/home/pi/pam/pam_driver/
		pamdriver	pam service executable	init.d	started on boot & controlled by restore_pam_service.sh	/etc/init.d/
	Monitor					
		restore_pam_service.sh	Script that restarts the pam driver	Root Crontab	Every two Hours	/home/pi/pam/pam_driver/
	Data					
		uploadepochlog.sh	Upload pam epoch (15 minute interval moverment data)	Pi User Crontab	Once a day on 23:56	/home/pi/pam/uploadscripts/
		uploadtodaylog.sh	Upload pam today data (day summary)	Pi User Crontab	Once a day on 23:58	/home/pi/pam/uploadscripts/
		PAM_epoch_data_parser.js	Transform PAM epoch data logs to JSON	uploadepochlog.sh	none	/home/pi/pam/uploadscripts/
		PAM_todayvalue_data_parser.js	Transform PAM today value data logs to JSON	uploadtodaylog.sh	none	/home/pi/pam/uploadscripts/
Z-Wave	Driver					
		app.js	NodeJS zwave driver source	zwavedriver	none	/home/pi/zwave/scanner/
		zwavedriver	zwave service executable	init.d	started on boot & controlled by check_zwave_service.sh	/etc/init.d/
		get_cookie.sh	zwave local cookie grab script, necessary for app.js to get acces to the internal zwave driver	Pi User crontab	Every hour	/home/pi/zwave/scanner/
		getsensornames.sh	Wrapper script to extract the names of all sensors attached to the zwave driver	User	None	/home/pi/zwave/scanner/
		get_sensornames.js	NodeJS to extract the names of all sensors attached to the zwave driver			
	Monitor					
		check_zwave_service.sh	Script to check and restart the z-way-server(razberry driver executable) if necessary	Root crontab	Every hour	/home/pi/zwave/

		check_zwave_driver.sh	Script to check and restart the zwavedriver	Root crontab	Every hour	/home/pi/zwave/
	Data					
		uploadzwavelog.sh	Script to upload the zwave data of the current day	PI User crontab	Once a day on 23:57	/home/pi/zwave/scanner/
		uploadsensornames.sh	Script to upload the sensor names registered to the zwave driver	User	None	/home/pi/zwave/scanner/
Server	Status updates/heartbeats	heartbeat.sh	Script to upload the current: temperature,diskspace,uptime, vpn address to the server & provide tunnel backdoor	Root crontab	Every minute	/home/pi/pam/uploadscripts/
	Recovery	restore_box.sh	script that checks for an active internet connection, if it failes, the usb ports are turned off and on, and internet connection is re-initialized, if the script fails five consecutive times, the box is rebooted	Root crontab	Every hour	
	Remote access	heartbeat.sh	Script to read out a command response from the server and execute it. e.g. opening reverse SSH tunnel	Root crontab	Every minute	
	Box ID	get_box_id.sh	Script to get the database id of this box (based on MAC address of ETH0) , used by uploadzwavelog.sh to upload to the correct folder	uploadzwavelog.sh	None	/home/pi/zwave/scanner/

Naast de software geïnstalleerd op de Hipperbox is er ook software op de server geïnstalleerd. Deze software is beschreven in een het hoofdstuk webapplicatie.

3. Data

3.1 Data omgevingssensoren (zwave data)

De data van de omgevingssensoren (zwave-data) wordt op de Hipperbox opgeslagen als CSV-bestanden. Deze bestanden worden per dag opgeslagen en worden 's nachts verstuurd naar de server. De bestandsnaam geeft aan van welke hipperbox de data afkomstig is en welke dag de verzamelde data betreft. De ruwe data heeft het volgende formaat:

<unixtimestamp>, <year>, <month>, <day>,<hour>,<minute>, <second>, <sensor_id>,<type>,<value>

een voorbeeld van de omgevingsdata is hier onder weergegeven:

```
1472625914, 2016,08,31,8,45,14, 9,Movement,1
1472625931, 2016,08,31,8,45,31, 4,Door,1
1472625943, 2016,08,31,8,45,43, 4,Door,0
1472625948, 2016,08,31,8,45,48, 6,Movement,0
1472625963, 2016,08,31,8,46,3, 9,Movement,0
1472625290, 2016,08,31,8,34,50, 6,Light,50
1472625280, 2016,08,31,8,34,40, 8,Light,50
1472625292, 2016,08,31,8,34,52, 9,Light,50
1472625816, 2016,08,31,8,43,36, 11,Light,0
1472625294, 2016,08,31,8,34,54, 4,Temperature,19.3
1472625290, 2016,08,31,8,34,50, 6,Temperature,25.1
```

…

Deze dat wordt overgezet naar de server waar de analyse van de data gebeurt met behulp van een aantal R-scripts. Het resultaat van deze analyse wordt opgeslagen in de processed folder als JSON formaat.

3.2 Data draagbare sensor (PAM-data)

De data van de PAM-sensor wordt op de server opgeslagen als JSON-bestand. De bestandsnamen van deze bestanden geven van welke PAM de data afkomstig is en welk type data het bestand bevat.

<pamnummer>_<datatype>_<datum>.log

Een voorbeeld van de PAM-data is hier onder weergegeven. Naast het seriele nummer van de PAM wordt er per dag de scores per kwartier bijgehouden.

PACKET_START:
jsonstring={"serial_number":"19306","epochdata":[{"date":"2016-09-30","scores":"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000010003010100040100000000000000000000000000000000000000000000000000000000000000000000000000"},{"date":"2016-10-01","scores":"000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000"},
…
{"date":"2016-10-14","scores":"000000000000000000000000000000000000000000000000000000000000000005040203040504040301000104010202010302000002020403030201010001020301010102020100020303000000000000000000000000000000000000000000"}]}

PACKET_END

Een voorbeeld van de today values is hieronder weergegeven. Deze waardes komen overeen met wat op de website wordt weer gegeven.

{"date":"2016-10-18","values":[{"pam":"9.69","light_activity":"63","medium_activity":"11","heavy_activity":"0"}]}

3.3 Data systeemgebruikers

De opgeslagen databestanden, zowel van de PAM als van de omgevingssensoren, bevatten geen verwijzingen naar de persoon van wie deze data afkomstig is. Deze gegevens staan in een MySQL database.

De database legt de koppeling tussen cliënten en de Hipperbox (middels een id) en PAM-sensor (middels een id) die zij voor een bepaalde periode in gebruik hebben. Van een client worden verder de aanspreekvorm, voornaam, achternaam en een username en wachtwoord bijgehouden.

De database bevat daarnaast de gegevens van de betrokken therapeuten en onderzoekers: gebruikersnaam, emailadres en wachtwoord en legt de link tussen de revalidanten en hun behandelend therapeuten.

De wachtwoorden van alle gebruikers worden gehashed voordat deze in de database worden opgeslagen. Het is niet mogelijk vanuit de database het door de gebruiker gekozen wachtwoord te achterhalen. In verband van de veiligheid, biedt het systeem geen mogelijkheid om een nieuw wachtwoord in te stellen. Hiervoor moet contact met de beheerders opgenomen worden.

Per client kan een therapeut notities toevoegen en vragen stellen. Iedere therapeut heeft zijn eigen collectie vragen. De antwoorden worden per client bijgehouden.

Ten behoeve van het onderzoek worden van revalidanten nog een aantal gegeven bijgehouden: geboortedatum, datum van operatie, type operatie, behandellocatie, belastingadvies behandelend arts, datum 100% belasting. Deze gegevens worden buiten de database in een versleuteld bestand opgeslagen. Alleen de hoofdonderzoeker heeft de sleutel van dit bestand.

4. Webapplicatie

De Hipper webapplicatie biedt therapeuten over het internet toegang tot de data die wordt verzameld door het Hipper sensorsysteem. Met behulp van de webapplicatie kunnen therapeuten cliënten aanmelden en cliëntgegevens beheren, visualisatie van de sensordata van hun eigen cliënten inzien en vragen aan cliënten stellen.

De webapplicatie bevat functionaliteit voor het beheren van systemen en therapeutenaccounts (alleen toegankelijk voor onderzoekers) en vormt de basis van de cliëntenapp. Dit document beschrijft de technische architectuur van de webapplicatie. Voor de werking van de applicatie vanuit gebruikersperspectief wordt verwezen naar de handleiding voor therapeuten.

De webapplicatie is een single page application ontwikkeld op basis van angular.js (versie 1.4.8) en d3.js (versie 3). De vormgeving van de applicatie is gebaseerd op de Skeleton CSS Boilerplate. Vormgeving van de app staat in css/app.css, vormgeving van de visualisaties in css/chart.css.

De data is toegankelijk via een beveiligde REST-API geïmplementeerd met het PHP Slim framework. De API is pas toegankelijk nadat een gebruiker heeft ingelogd. Als een gebruiker succesvol inlogt, wordt een cookie op het systeem van de gebruiker geplaatst met daarin een session-key. Om session hijacking te voorkomen wordt bij elke API call gecontroleerd of de session-key nog geldig is. De API communiceert met de database, hierbij wordt rekening gehouden met het gevaar van SQL-injection door uitsluitend gebruik te maken van prepared statements.

De architectuur van de webapplicatie volgt het model-view-controller (MVC) pattern. Data is opgeslagen in modellen. Deze modellen komen overeen met resources (cliënten, pam data, zwave data, notities,…) op de server en halen data op doormiddel van AJAX calls naar de REST-API van de server. Toegang tot data vanuit de rest van de applicatie verloopt altijd via de modellen. De huidige modellen maken geen gebruik van caching of enige andere vorm van locale data-opslag, maar communiceren rechtstreeks met de server. Alle modellen staan in model/models.js. De applicatie benadert data op de server waarvan de applicatie zelf afkomstig is. Als dit verandert moeten de verwijzingen in models.js worden aangepast.

De voor de gebruiker zichtbare schermen zijn views. Dit zijn html-bestanden met een aantal angular attributen.

De koppeling tussen de views wordt gevormd door controllers en directives. Het gedrag van de applicatie wordt bepaald door de top level controller hipperCtrl (in controllers/hipper.js). Deze controller bevat code voor routing naar de juiste subpagina en code om cliëntgegevens op te halen van de hipper server.

Welke gegevens worden opgehaald en welke subpagina en menu items getoond worden, hangt af van een aantal variabelen die door de server in index.html geplaatst worden (is_sohip en is_client).

De functionaliteit van de verschillende views wordt geboden door specifieke controllers. In alle gevallen blijft de functionaliteit van hipperCtrl toegankelijk.

De hoofdfunctionaliteit van de applicatie is het visualiseren van sensordata van specifieke cliënten. Deze functionaliteit wordt geboden door views/main.html en controllers/main.js.

De belangrijkste taak van mainCtrl is het ophalen van pam-data (die in alle visualisaties wordt gebruikt).

Data voor de activiteitengrafieken wordt opgehaald door activitiesCtrl en notities worden beheerd door notesCtrl, beide controllers zijn subcontrollers van mainCtrl (d.w.z. de functionaliteit van mainCtrl blijft toegankelijk in deze controllers).

De meeste schermen bevatten controls voor het selecteren van schermen en het instellen van de datums waarover data moet worden gevisualiseerd. Deze en een ander algemene functionaliteit is in de vorm van custom directives beschikbaar in directives.js.

De data visualisaties zijn eveneens custom directives en geïmplementeerd m.b.v. d3.js. Ieder grafiek heeft een eigen bestand in de map charts. Functionaliteit die door meerdere grafieken wordt gebruikt staat in charts/charts.js. De datavisualisatie grafieken krijgen toegang tot de betreffende data via two way data binding, dit betekent dat de grafieken automatisch worden aangepast als de gelinkte data verandert, bijvoorbeeld omdat een andere tijdsperiode wordt geselecteerd.

De webapplicatie vormt de basis voor de cliëntenapplicatie. In dat geval wordt dat applicatie gestart vanuit een android app. In dat geval is de globale variabele is_client gezet en wordt de answers view geopend i.p.v. de main view. De overeenkomstige AnswersCtrl zorgt dat eventuele vragen die voor de cliënt klaar staan worden getoond.

De relatie tussen de verschillende views, controllers en modellen is schematisch weergegeven in Fig. 3.

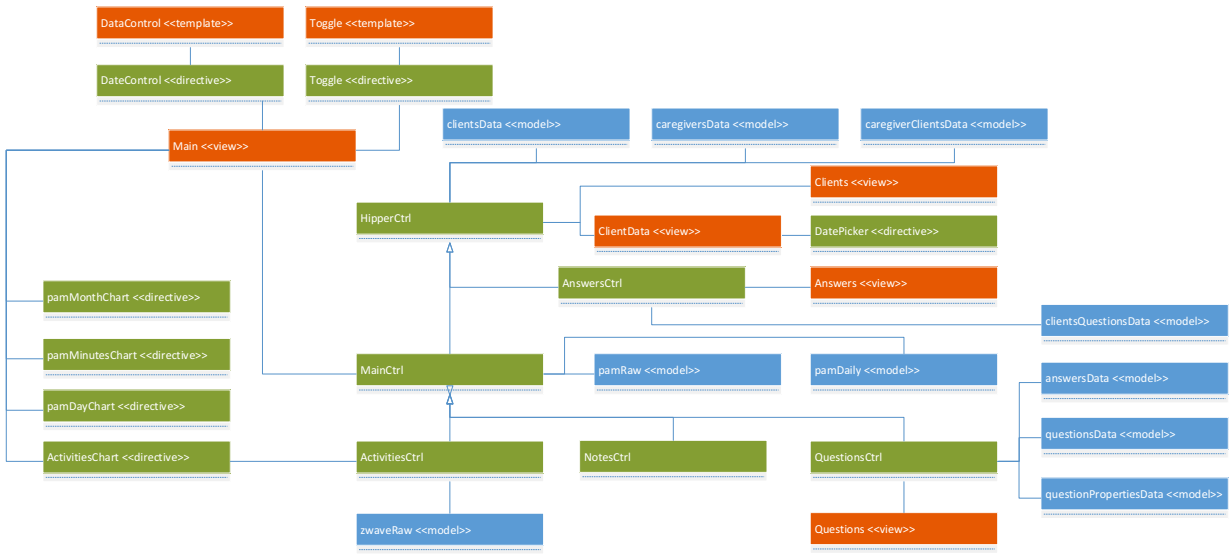

Fig. 3. Class diagram: views zijn weergegeven in rood, modellen in blauw en controllers en directives in groen.