

Verslag E-ink room reservation display

31/05/2019

Baptiste Pattyn, Michel Dequick, Stijn Declerck, Ine Vanderhaeghe

E-ink room reservation display

1 Inhoud

Inhoudsopgave

1 Inhoud	2
2 Probleemstelling	3
3 Aanpak	5
4 Implementatie	6
4.1 Database en Wifi Access Point	6
4.1.1 Database	6
4.1.2 Wifi Access Point op Raspberry Pi	7
4.1.3 Webserver en GET request	9
4.2 Communicatie tussen de mbed en de database	10
4.2.1 De verbinding	10
4.2.2 De gegevens	10
4.3 JSON parsing	11
4.4 E-paper display	11
5 Resultaten	12
6 In het vervolg	14
7 Conclusie	15

2 Probleemstelling

Geschreven door Ine

De bedoeling van dit project is om een display te maken voor klaslokalen, dat dynamisch kan veranderen per uur.

Op het scherm moeten verschillende zaken komen: het nummer van het klaslokaal, de datum, op welke uren het lokaal bezet is op een bepaalde dag, welk vak op dit moment gegeven wordt en de docent van dit vak.

Er zijn verschillende zaken waar rekening mee gehouden moet worden om dit te realiseren:

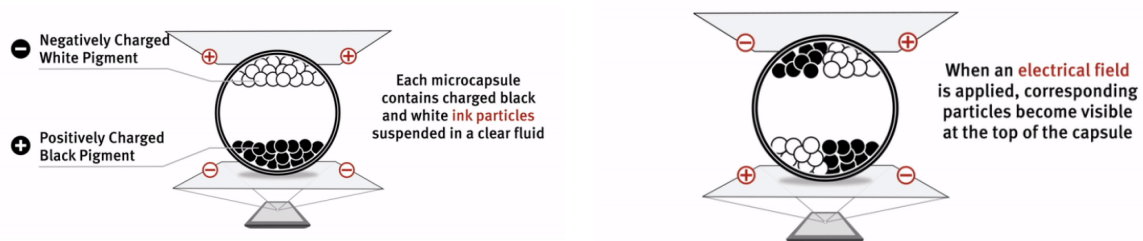
Er moet een E-ink display geïmplementeerd worden. Hierbij wordt bekeken als er burn-in of ghosting kan optreden. Ook wordt rekening gehouden met de grootte van de memory in de display.

Er moet een databank opgezet worden waar alle data van de lokalen in opgeslagen is. Deze databank is bereikbaar via wifi.

Er moet een connectie gemaakt worden via een wifi module van de databank naar de display.

Een E-ink display krijgt hier de voorkeur, omdat dit het meest energiezuinig is. Dit is zo omdat de display enkel voeding nodig heeft om het scherm te veranderen. Het scherm moet maar om het uur aangepast worden, dus alle tijd daartussen heeft een E-ink display geen voeding nodig. Een ander voordeel van een E-ink display is dat het geen licht uitzendt, maar het reflecteert licht zoals een blad papier. Hierdoor is het gemakkelijker leesbaar, ook als er in de omgeving veel licht is.

De eenvoudigste E-ink display is de "two pigment ink system". Dit bestaat uit kleine gebieden die dipolen zijn (deze worden gebruikt als de pixels van de afbeelding). De positieve kant van de gebieden bestaat uit wit pigment, de negatieve kant is zwart pigment. Deze gebieden bevinden zich in een bubbel van olie zodat ze gemakkelijk kunnen omdraaien, en zitten tussen transparante elektrode lagen. Wanneer nu op die elektrodelagen een spanning wordt gezet, kan bepaald worden welke delen van het scherm zwart zijn, en welke wit. Op die manier kan een zwart-wit afbeelding op de display geprogrammeerd worden. (zie afbeelding 1).



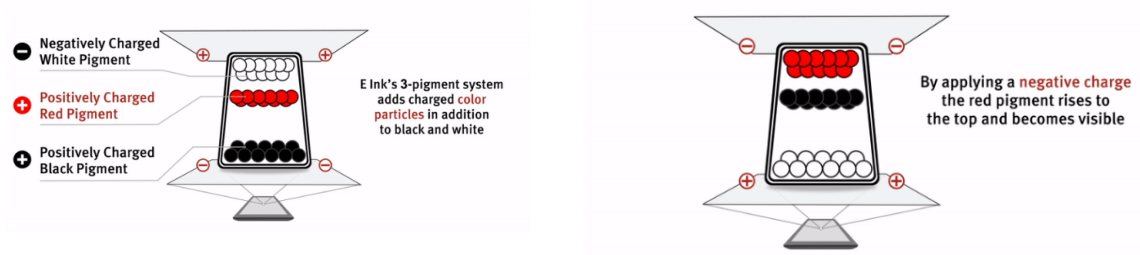
Figuur 1: two pigment ink system

Er bestaat ook een "Three pigment ink system". Dit werkt op ongeveer dezelfde manier als two pigment ink system (zie afbeelding 2). Hier is het witte pigment negatief geladen en het rode pigment is positief geladen. Om het zwarte pigment aan de oppervlakte te brengen, moet er een gesplitste lading aangebracht worden.

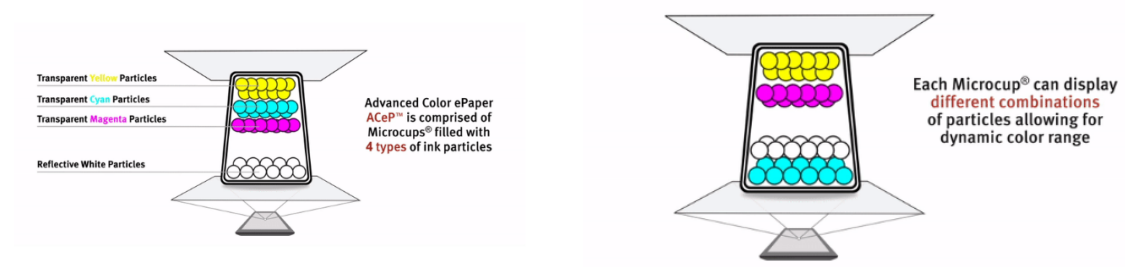
De E-ink display die in dit project gebruikt wordt, gebruikt het three pigment ink system. Op deze manier kan op een overzichtelijke manier op een tijdslijn aangeduid worden op welke uren het lokaal bezet is.

Daarnaast bestaat er ook een "Advanced color ePaper". Hierbij kunnen nog meer kleuren op het scherm afgebeeld worden. (zie afbeelding 3). Dit wordt in dit project niet gebruikt. Dit systeem gebruikt 4 kleuren: cyaan, magenta, geel en wit.

bron figuren: [4].



Figuur 2: three pigment ink system



Figuur 3: Advanced color ePaper

bron werking ACeP: [1].

Dit is het materiaal dat gebruikt werd:

Mbed FRDM-K64F

met application shield en click shield

ESP-WROOM-02 click met ESP8266EX wifi bord

Datasheet ESP-WROOM-02: [2]

Datasheet ESP8266EX: [3]

Eink click, small and big display

3 Aanpak

Geschreven door Ine

Eerst hebben we alle deelopdrachten op een rijtje gezet en een planning gemaakt. Daarbij maakten we ook een taakverdeling op.

Dit zijn de grootste taken:

Een library maken voor de display – Michel

Eerst werd er bekeken hoe we een afbeelding op de display konden krijgen. Op het internet vonden we een voorbeeld van een E-ink display die gemaakt was met een Arduino. Wij maken gebruik van een mbed, dus we moesten de code aanpassen. Als eerste hebben we geprobeerd om een afbeelding op de display te zetten én om een afbeelding die er op staat uit te wissen. Daarna maakten we een ontwerp van wat we allemaal op het scherm wilden hebben en waar juist. We hebben dan ook geprobeerd om dit op de display te krijgen.

Een database opstellen met een wifi access point – Baptiste

We hebben een database opgesteld op een Raspberry Pi. Daar werd ook een wifi access point op geïmplementeerd zodat de display met de databank kan communiceren. Eens de databank opgevuld was kon getest worden als de data in een tabel per lokaal gezet kon worden.

Wifi connectiviteit en JSON parser maken – Stijn

Het is belangrijk om via wifi te kunnen communiceren tussen de E-ink display en de databank. Hiervoor moesten we onderzoeken hoe een HTTP GET request werkt en hoe we het zelf konden implementeren in onze toepassing.

Er moet ook een manier gevonden worden om de data te versturen. Hiervoor werd een JSON parser gemaakt.

Er waren ook enkele kleinere taken:

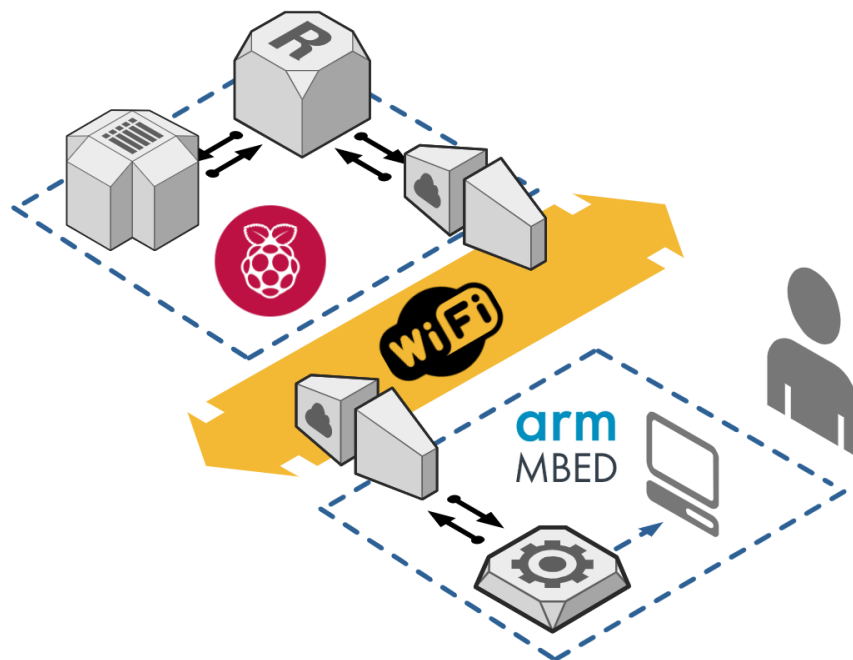
De databank invullen – Ine

De databank werd opgevuld met fictieve data over de lokaalbezetting zodat we een proof of concept konden maken om voor te stellen op de presentatie van ons project.

Het verslag en de powerpointpresentatie opstarten – Ine

Om het verslag te maken heeft iedereen het deel waar hij zelf het meest aan gewerkt heeft aangevuld.

In figuur 4 is te zien hoe de verschillende delen aan elkaar gelinkt zijn.



Figuur 4: aanpak

4 Implementatie

4.1 Database en Wifi Access Point

Geschreven door Baptiste

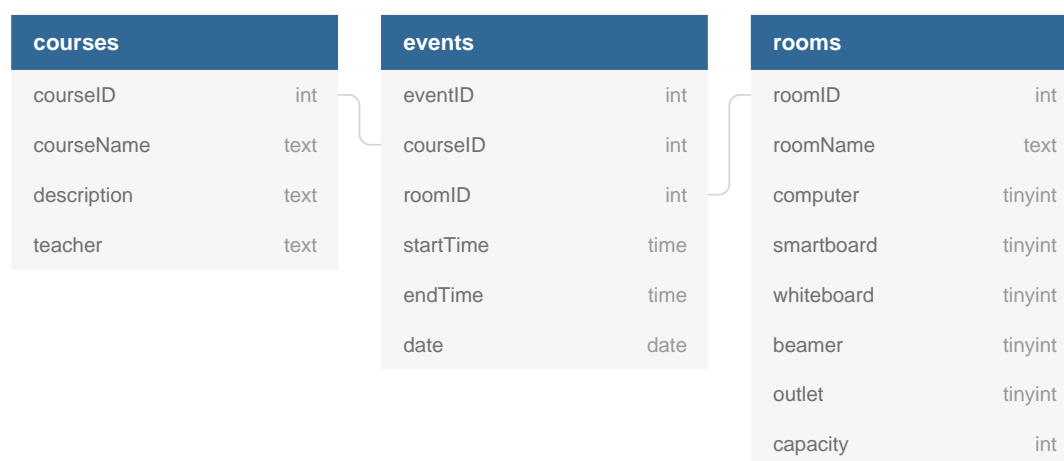
Voor het opslaan van de gegevens is er een database opgezet op een Raspberry Pi model B+ door gebruik te maken van phpMyAdmin. Hier draait een SQL server op waarin alle gegevens voor de komende events van verschillende lokalen kunnen opgeslagen worden. Verder zijn alle gegevens van de courses er ook opgeslagen en de algemene info van de rooms. Om ervoor te zorgen dat onze MBED toegang kan krijgen tot deze database hebben we een Wifi Access Point (WAP) op de Raspberry Pi gezet, waarmee de MBED zich kan verbinden via een ESP8266 Wifi chip. Het opvragen van de gegevens door de mbed gebeurt via een GET request op de HTTP poort van de Raspberry Pi (poort 80).

4.1.1 Database

Deze database genaamd roomdb bestaat uit 3 verschillende tables: courses, events en rooms. De structuur van de database en de relaties tussen de verschillende tables staat afgebeeld in figuur 6. Op de Raspberry Pi staan verschillende PHP files die ervoor zorgen dat de webserver kan verbinden met de database. Een belangrijke file hierbij is dbConnect.php (zie figuur 5), in deze file staan de gegevens die nodig zijn om te kunnen verbinden met de database. In deze file declareren we verschillende variabelen: `$host`, `$user`, `$password` en `$database`. Deze worden dan gebruikt in een `mysqli_connect()` commando om een verbinding op te zetten met de database. Hierna wordt er nog een controle gedaan om te kijken of de verbinding tot stand is gekomen, indien dit niet het geval is wordt er een error gegenereerd. Deze code is enkel bedoeld voor tijdens het opzetten van de webserver en de database en dient verwijderd te worden indien de code definitief in gebruik wordt genomen. Als `$host` wordt het local host adres 127.0.0.1 gebruikt omdat de database zich op de Raspberry Pi zelf bevindt. De `$user` en het `$password` zijn specifiek voor elke database en hangen af van welke user ingesteld werden op de SQL server.

```
<?php
$host = "127.0.0.1";
$user = "root";
$password = "password" ;
$database = "roomdb";
$conn = mysqli_connect($host, $user, $password, $database);
// delete when ready to launch
if(!$conn){
die("Connection failed: " . $conn ->connect_error);
}
?>
```

Figuur 5: dbConnect.php



holistics

Figuur 6: Database structuur

4.1.2 Wifi Access Point op Raspberry Pi

Voor het opzetten van de WAP op de Raspberry Pi werd een online step-by-step guide gevolgd [5]. De belangrijkste services die hiervoor geïnstalleerd werden op de Raspberry Pi zijn **hostapd** en **dnsmasq**.

DNSmasq

Deze service is in principe gewoon een DHCP (Dynamic Host Configuration Protocol) server. Deze zal ervoor zorgen dat een dynamisch IP adres kan toegekend worden op het netwerk zonder dat alle apparaten manueel moeten verbonden worden. In de config file van dnsmasq (die zich onder het volgende path bevindt `/etc/dnsmasq.conf`) wordt een range toegekend van IP adressen die de Wifi kan gebruiken om toe te kennen aan apparaten. Verder wordt ook bepaald hoelang deze IP adressen kunnen gebruikt worden door de apparaten die zich verbinden met het Wifi netwerk. De config file ziet er dan als volgt uit 7 .

```
interface=wlan0
dhcp-range=192.168.0.11,192.168.0.30,255.255.255.0,24h
```

Figuur 7: dnsmasq.conf

Hostapd

Deze service zorgt ervoor dat er een WAP wordt opgezet. In de config file van hostapd (die zich onder het volgende path bevindt `/etc/hostapd/hostapd.conf`) wordt het SSID en het wachtwoord ingesteld van de WAP. Verder wordt hier ook een bridge ingesteld tussen de ethernet verbinding en de wlan0 van de Raspberry Pi. Dit zorgt ervoor dat er zowel via ethernet als via de WAP toegang kan verkregen worden tot de webserver. Dankzij deze file gebruikt de Wifi steeds channel 7. Ook wordt de Wifi Protected Access (WPA) versie ingesteld en de vorm van management die gebruikt zal worden voor het WPA wachtwoord wordt bepaald. Voor deze opdracht wordt WPA-PSK gebruikt wat staat voor Pre-Shared Key. Een wachtwoord bestaat uit minimaal 8 en maximaal 63 tekens. De config file ziet er dan als volgt uit 8.

```
interface=wlan0
driver=nl80211
bridge=br0
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
wpa_pairwise=CCMP
ssid=teamzoepertoff
wpa_passphrase=stijnkanniks
```

Figuur 8: hostapd.conf

4.1.3 Webserver en GET request

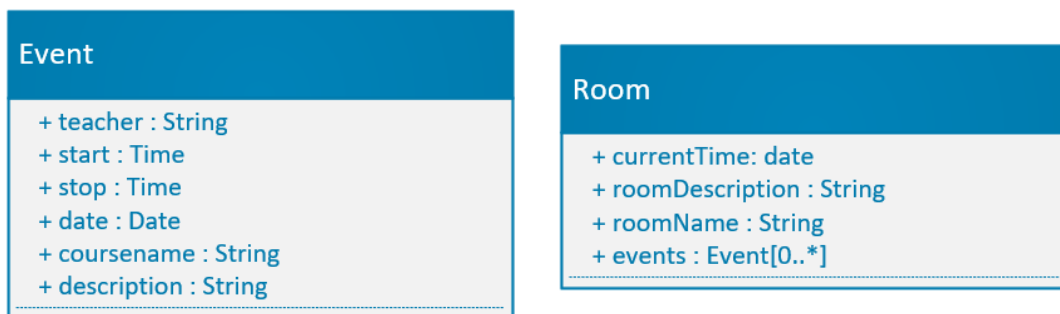
De webserver die op de Raspberry Pi draait is een apache 2.4 server die de mogelijkheid heeft om samen te werken met de SQL database. Voor deze specifieke applicatie worden slechts vier van de zes aanwezige files gebruikt:

- `dbConnect.php`
- `event.php`
- `room.php`
- `roominfo.php`

De overige twee files (`index.php` en `showTable.php`) kunnen nodig zijn indien er via een webbrowser toegang verkregen moet worden tot de lokaalbezetting. Wanneer er een GET request verstuurd wordt vanuit de MBED, wordt de `index.php` file overgeslagen. Een GET request ziet er als volgt uit:

`http://ip-address/E-ink-room/roominfo.php?roomName=02.85.`

Er wordt dus rechtstreeks naar `roominfo.php` gegaan en als variabele wordt het nummer meegegeven van het lokaal waarvan de info verkregen moet worden. De inhoud van `dbConnect.php` werd hierboven al besproken, dus zal hier niet meer verder op ingaan worden. `event.php` en `room.php`, zijn hele kleine files die beide één enkele klasse bevatten. De UML diagramma's voor deze klassen staan in figuur 9.



Figuur 9: UML diagram

De laatste en tevens ook grootste file is `roominfo.php`. Hierin worden de klassen die in de vorige files gedefinieerd werden gebruikt. Bij het GET-request wordt een variabele meegegeven die `roomName` genaamd is. Deze wordt opgeslagen in een variabele met de naam `$roomName`. Om veiligheidsredenen is het belangrijk om eerst de door de gebruiker meegegeven variabele te sanitizen. Daardoor kan er geen SQL code geïnjecteerd worden die er eventueel voor kan zorgen dat onze database aangepast wordt zonder dat dit de bedoeling is. De methode die hiervoor gebruikt werd is `mysqli_real_escape_string(string $escapestr)`.

De variabele `$roomName` kan nu gebruikt worden om een query uit te voeren naar de database om zo het `roomID` en de `roomDescription` te kunnen verkrijgen van het opgevraagde lokaal. Het opvragen van deze gegevens gebeurt volgens de code in figuur 10. Eerst wordt een statement aangemaakt via de `prepare()` methode. Hierin wordt de SQL code gezet met de parameters die moeten geselecteerd worden en de voorwaarde die aangeeft welke entries van de database mogen opgehaald worden. De waarde van de variabele waarop geselecteerd wordt, wordt later toegekend met de `bind_param()` methode. De eerste parameter geeft aan dat we selecteren op een variabele van het type String en de tweede parameter is de variabele zelf. Na de `execute()` methode gebruiken we de `bind_result()` methode om de variabelen van de opgehaalde database entry toe te kennen aan de correcte variabelen zodat ze later gebruikt kunnen worden. Het `roomID` is nodig omdat op die manier gezocht kan worden in de event table naar alle geplande events die het opgegeven `roomID` bevatten.

```
$roomName = mysqli_real_escape_string($conn, $_GET['roomName']);  
$stmt = $conn->prepare("SELECT roomID, roomDescription FROM rooms WHERE roomName=?");  
$stmt->bind_param("s", $roomName);  
$stmt->execute();  
$stmt->bind_result($roomID, $roomDescription);
```

Figuur 10: Info over lokaal ophalen uit database

Na het ophalen van het roomID wordt een SQL query uitgevoerd om alle entries uit de events table te halen die doorgaan in het opgevraagde lokaal. De resultaten van deze queries worden gerangschikt in arrays, in dezelfde volgorde als in de table. Zo kan men op een eenvoudige manier de verschillende variabelen opvragen door de indexen te gebruiken van de arrays. Alle elementen van een bepaald event bevinden zich namelijk op de plaats in de array met dezelfde index. Als laatste stap dienen via de courseID's de courseNames opgevraagd te worden in de courses table om zo alle nodige info beschikbaar te hebben om terug te sturen naar de MBED.

Eerst dienen alle gegevens opgestuurd te worden voor ze in een JSON-file kunnen. Hiervoor wordt een object aangemaakt van het type Room. Zoals aangetoond in figuur 9 heeft dit object 4 objecten. Via de `$myJson = json_encode($room)` kan het object dan omgezet worden naar een JSON variabele die dan teruggestuurd kan worden met het commando `echo "$myJson";`. Deze JSON-file met alle gegevens wordt dan op de MBED verwerkt.

4.2 Communicatie tussen de mbed en de database

Geschreven door Stijn

Teneinde het lessenrooster te printen heeft de mbed gegevens nodig van de database. Communicatie tussen de mbed en de database is dus nodig. De mbed heeft de rol van client en vraagt dus die gegevens aan de database, die logischerwijs de rol van server heeft.

De communicatie gebeurt via wifi. Op de mbed zit een wi-fi module ESP-WROOM-02 Click [2] die in staat is de data te ontvangen en versturen. In het programma dat loopt op de mbed wordt gebruik gemaakt van 2 functies om gegevens te verkrijgen.

4.2.1 De verbinding

Hoe wordt deze verbinding tot stand gebracht?

De eerste functie `init()` maakt verbinding met het wifi access point van de database en gaat als volgt te werk: eerst wordt gecontroleerd of de mbed een wi-fi module heeft (dit is in ons geval de ESP-WROOM-02). Daarna probeert de mbed aan te melden bij de raspberry pi met een opgegeven `WIFI_SSID` en het daarbij horende opgegeven `WIFI_PASSWORD`. Deze kunnen eenvoudig worden aangepast in een `.json` file.

4.2.2 De gegevens

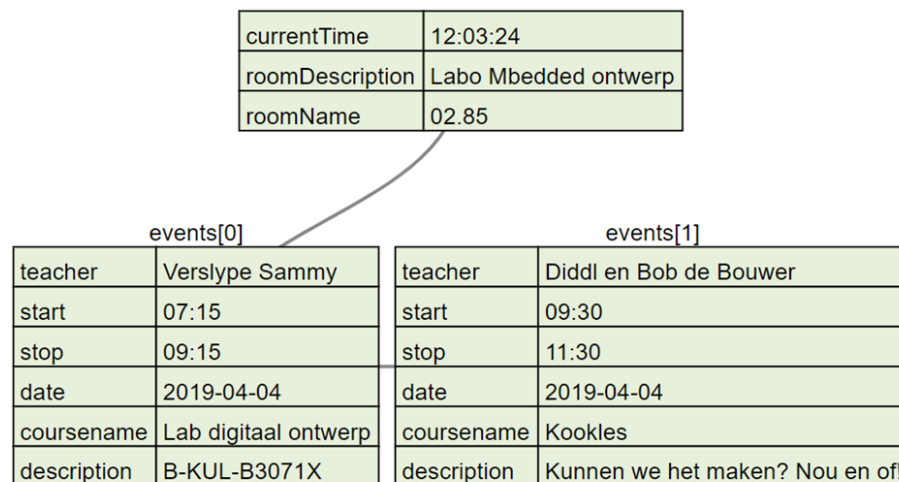
Hoe worden de gegevens uit de database verkregen?

De tweede functie `get(IP_ADDRESS, ROOM)` geeft een `JSON_STRING` terug met daarin alle gegevens voor één lokaal. Het `IP address` en `room number` wordt meegegeven met deze functie. Eerst wordt het GET request samengesteld met de gekregen waarden. Daarna wordt een `TCP Socket` geopend en gekoppeld aan de Wi-Fi interface, waarna een connectie wordt gelegd met de database. Als die connectie er is, wordt het GET request verstuurd naar de database, die daar dan normaal gezien op antwoordt. De mbed leest de data in en slaat ze op in een buffer. Dit is echter nog de ruwe data. De buffer wordt uitgelezen en enkel de nodige gegevens worden opgeslagen in een tweede buffer. Die laatste is een `JSON_STRING` en wordt dus teruggegeven door de functie.

4.3 JSON parsing

Geschreven door Michel

Voor het parsen van de ontvangen JSON strings wordt er gebruik gemaakt Samuel Mokrani's MedJSONValue library. De library biedt de mogelijkheid aan om een string naar een MbedJSONValue object te parsen. Uit het bekomen object kan dan gemakkelijk aan de hand van de JSON keys de warden gehaald worden en direct getypecast worden naar het gedefinieerde type. In tegenstelling met vele andere librarys kan deze wel overweg met JSON arrays van objecten, meeste librarys ondersteunen alleen het parsen van single-type arrays. Een handige feature is dat er ook makkelijk nagegaan kan worden hoeveel elementen een array bevat. ??



Figuur 11: overzicht van de json parser

4.4 E-paper display

Geschreven door Michel

De driver is gebaseerd op een arduino library die door de producent van de display beschikbaar gesteld is geweest. Deze wordt geport naar MBED. Aan de hand van een proof of concept art is er besloten hoe het scherm ingedeeldt wordt. Het tekenen van de objecten gebeurt in een lus die telkens een klein deel van het scherm gaat renderen. Dit is zo besloten zodat de beeld buffer variable kan gemaakt worden om RAM gegeugen uit te sparen. Na het renderen van een sub-frame wordt die naar de S-RAM van het beeldscherm geschreven langs de SPI bus. Eenmaal alle sub-frames gerendert en weggeschreven zijn wordt het signal gegeven aan de display om zijn S-RAM te displayen.

5 Resultaten

Geschreven door Ine

We zijn er in geslaagd om uit de databank info van lokaal 02.85 filteren en in een tabel zetten. Zie figuur 12.

Bezetting Lokaal 02.85

Date	Start Time	End Time	Course Name	Course Description	Teacher(s)
2019-03-27	08:15:00	17:45:00	Projectlab bachelor elektronica-ICT	B-KUL-B3390N	Espeel Ludovic Lannoo Jonas
2019-04-03	08:15:00	17:45:00	Projectlab bachelor elektronica-ICT	B-KUL-B3390N	Espeel Ludovic Lannoo Jonas
2019-04-05	12:30:00	18:45:00	Projectlab bachelor elektronica-ICT	B-KUL-B3390N	Espeel Ludovic Lannoo Jonas
2019-04-05	08:15:00	10:15:00	C programmatie	B-KUL-B3391N	Van Waes Jonas
2019-04-04	13:30:00	15:30:00	Projectlab bachelor elektronica-ICT	B-KUL-B3390N	Espeel Ludovic Lannoo Jonas

Figuur 12: lokaalbezetting lokaal 02.85

Wanneer we merkten dat dit werkte, hebben we daar nog data aan toegevoegd. Ook wordt de data niet meer in een bestand opgeslagen, omdat dit moeilijk is om door te sturen. Nu kunnen we de info van lokaal 02.85 filteren op 4 april 2019. Zie figuur 13. Alles staat nu als tekst na elkaar. Dit is gemakkelijker om door te sturen. Eens het doorgestuurd is, kunnen we er opnieuw de juiste data uit filteren en er een tabel van maken om op een overzichtelijke manier op de display weer te geven.

```
{
  "roomDescription": "Labo Mbedded ontwerp",
  "roomName": "02.85",
  "events": [
    {
      "teacher": "Verslype Sammy",
      "start": "07:15",
      "stop": "09:15",
      "date": "2019-04-04",
      "coursename": "Lab digitaal ontwerp",
      "description": "B-KUL-B3071X"
    },
    {
      "teacher": "Diddl,Bob de Bouwer",
      "start": "09:30",
      "stop": "11:30",
      "date": "2019-04-04",
      "coursename": "Kookles",
      "description": "Kunnen we het maken? Nou en of!"
    },
    {
      "teacher": "Dora the explorer",
      "start": "11:30",
      "stop": "13:30",
      "date": "2019-04-04",
      "coursename": "Aardrijkskunde",
      "description": "We did it! We did it! We did it!"
    },
    {
      "teacher": "Espeel Ludovic,Lannoo Jonas",
      "start": "13:30",
      "stop": "15:30",
      "date": "2019-04-04",
      "coursename": "Projectlab bachelor elektronica-ICT",
      "description": "B-KUL-B3390N"
    },
    {
      "teacher": "meneer Boma",
      "start": "15:45",
      "stop": "17:45",
      "date": "2019-04-04",
      "coursename": "Stijn kan niks",
      "description": "Mijn gedacht..."
    }
  ]
}
```

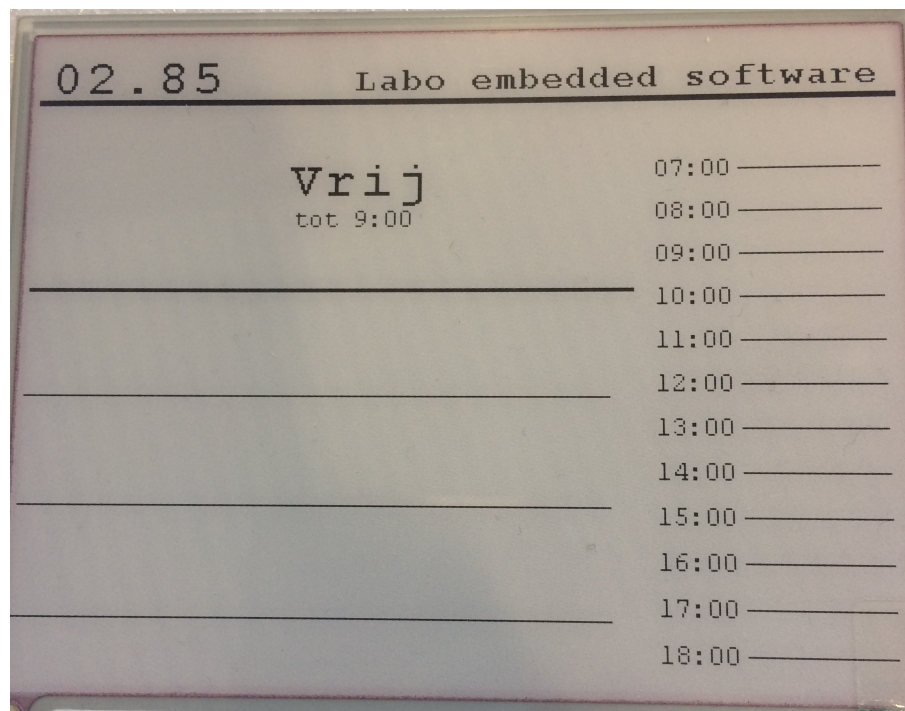
Figuur 13: lokaalbezetting lokaal 02.85 op 04/04/2019

We kunnen ook een figuur op de E-ink display afbeelden. Figuur 14 is een voorbeeld van voor we de data uit de databank konden doorsturen. Hier staan dus nog geen lessen in, maar het toont de lay-out van de display.

De database en het accesspoint waren relatief snel opgezet. Ook iets afbeelden op de display ging relatief vlot. Het moeilijkste was om de data te kunnen versturen naar de display. Het was ook niet eenvoudig om een framebuffer te maken, omdat we dachten dat het niet mogelijk was om een buffer te maken van het hele scherm. We hebben het scherm dus moeten verdelen zodat we voor elk stuk apart een buffer konden maken en zo het scherm stuk per stuk op te daten.

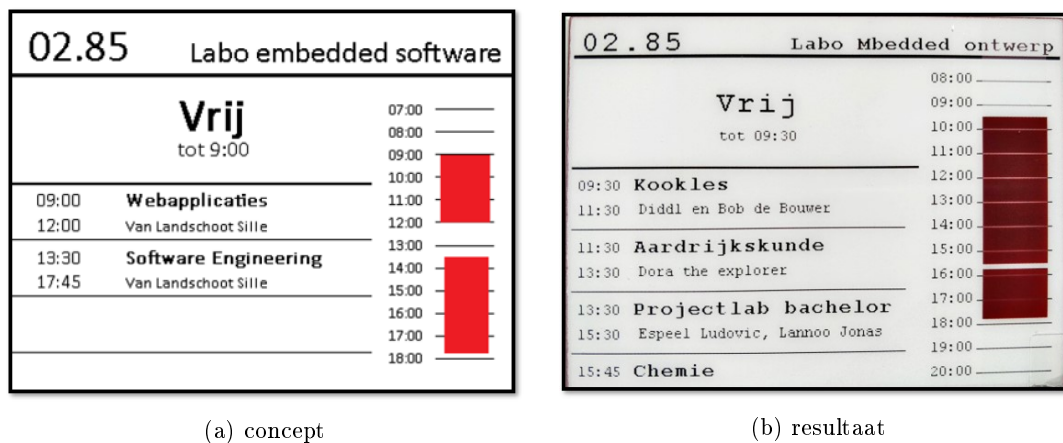
Uiteindelijk is in figuur 15 te zien wat het resultaat van dit project is. Links (15a) is te zien wat het doel van het project was. Rechts (15b) is het resultaat te zien wat wij hebben kunnen realiseren.

Hier is te zien dat het gelukt is om de verschillende zaken op het scherm te zetten die we op voorhand



Figuur 14: E-ink display

bedacht hadden. Op de tijdslijn is met rood aangeduid wanneer het lokaal niet vrij is. In dit voorbeeld is het lokaal dus niet bezet tot 9:30. Daarna zijn wel nog lessen gepland die ook zichtbaar zijn.



Figuur 15: Het resultaat van dit project

6 In het vervolg

Geschreven door Ine

Als we dit project verder zouden uitwerken, zouden we een andere mbed en wifi module gebruiken. Nu hebben we veel tijd verloren met het zoeken naar hoe we iets moeten implementeren. We zouden kunnen werken met modules die we kennen, waardoor alles sneller zou kunnen gaan.

We hebben veel tijd verloren door te zoeken naar hoe we een framebuffer van heel het scherm kunnen maken. We dachten dat dat niet mogelijk was, dus hebben we een oplossing gezocht om een buffer van een deel van het scherm te maken. Dit was niet gemakkelijk en heeft veel tijd ingenomen. Achteraf hebben we gehoord dat het wel mogelijk is om heel het scherm in één keer up te daten. We zouden dus kunnen uitzoeken hoe dat werkt en het scherm op die manier implementeren, omdat het updaten van het scherm zo sneller zou gaan.

Verder zouden we ook tijdens het zoeken naar oplossingen proberen meer geduld te hebben. Als er problemen waren met dingen die we niet begrepen, zouden we moeten efficiënter zoeken naar oplossingen. We zouden kunnen één manier van werken proberen volledig uit te werken en te begrijpen, in plaats van telkens iets kleins niet lukt, een volledig nieuwe manier te zoeken.

Tijdens het project hebben we ondervonden dat de database elke les een nieuw IP adres kreeg. Dit heeft af en toe voor problemen gezorgd. Het zou dus gemakkelijker zijn als we de database een statisch IP adres zouden geven.

Als we nog wat meer tijd hadden gehad, zouden we kunnen zoeken naar een manier om de JSON buffer automatisch af te laten knippen wanneer alle bruikbare data er in zit. Tot nu toe krijgt de buffer een vaste lengte, en de plaats die niet gebruikt wordt, wordt opgevuld met onleesbare tekens. Dit zou dan niet meer gebeuren.

7 Conclusie

Geschreven door Ine

We hebben een manier gevonden om op een energie zuinige manier een leslokaal dynamisch aan te duiden. We kunnen op een E-ink display aanduiden op welke uren van de dag het lokaal vrij is. Wanneer het lokaal niet vrij is, is duidelijk wie in het lokaal zit, welk vak er op dat moment gegeven wordt en hoe lang het nog zal duren voor het lokaal weer vrij is.

Referenties

- [1] Good e Reader. The first product with e ink advanced color e-paper is now available, 2019. [Online; accessed 13-May-2019].
- [2] Espressif Inc. *Datasheet ESP-WROOM-02*, 2.0 edition, 2016. [Online; accessed 13-May-2019].
- [3] Espressif Inc. *Datasheet ESP8266EX*, 6.0 edition, 2018. [Online; accessed 13-May-2019].
- [4] E Ink Holdings Inc. Electronic ink, 2017. [Online; accessed 13-May-2019].
- [5] SuerferTim. How to use your raspberry pi as a wireless access point, 2017. [Online; accessed 13-May-2019].