Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Smart Mirror

Gruppennummer 09

Informationstechnik Labor Sommersemester 2016

Prof. J. Walter

Gruppenmitglieder:
Hildenbrand, Ludwig Philipp *43145*
Jahn, Felix *43364*

Inhalt

1 Problemstellung

Im Zuge des IoT wird auch immer mehr das Mobiliar immer weiter eingebunden. Ein Smart Mirror gehört hier zu den ersten Einrichtungsgegenständen.

Der Zweck eines Smart Mirror ist es zum Beispiel die Anzeige des Wetter, der Uhrzeit und der nächsten Termine. Hierzu gibt es auch im Internet schon einige Lösungen, welche aber größtenteils nicht zufriedenstellend sind.

2 Stand der Technik

Im Internet gibt es bereits einige Projekte zum Thema Smart Mirror, die aber zum Teil nur einen geringen Funktionsumfang besitzen oder die nötige Peripherie für erweiterte Funktionen nicht beinhalten. Bei kommerziellen Geräten ist dies zwar der Fall aber diese sind noch extrem teuer.

3 Aufgabenstellung

Mit dem Rechnerbaustein „Intel Edison“ soll der Webserver für einen Smart Mirror zur Verfügung gestellt werden.

Die aufgerufene Weboberfläche soll über die USB-Schnittstelle des Intel Edison und einen DisplayLink-Adapter auf einem Monitor dargestellt werden.

Die zu erfüllenden Grundfunktionen beinhalten die Anzeige von Uhrzeit und Datum, sowie des lokalen Wetters. Wünschenswert wäre hierbei eine adaptive Nutzung durch die Angabe eines Aufenthaltsortes bei Zugriff auf den Webserver. So wäre der Webserver universell einsetzbar.

Eine ebenfalls wünschenswerte Funktion wäre die Anzeige aktueller Nachrichten sowie eines iCal-Kalenders, jedoch ist letzterer aus Datenschutzgründen nur schwer umsetzbar.

4 Anforderungsliste

Fakultät Maschinenbau Mechatronik
MMT
Hochschule Karlsruhe
Technik und Wirtschaft

Anforderungsliste

für

Smart Mirror

Auftrag: **09**

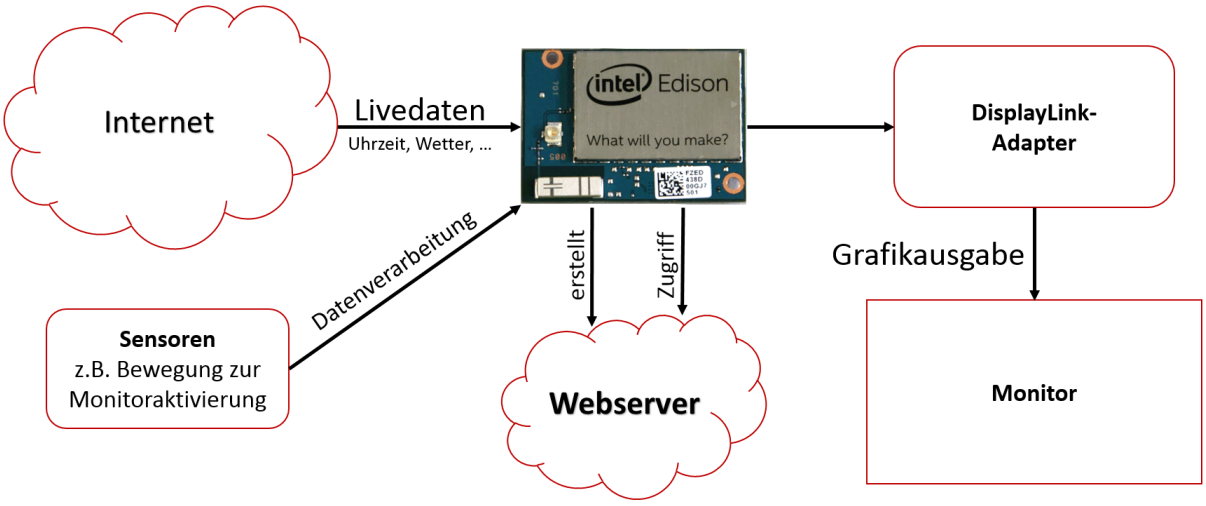
Informationstechnik Labor SS16

Organisations-Daten		Prozess-Daten			Wert - Daten		
Nummer	Name	Art	Phase	Anforderungen	Mindest-Erfüllung	SOLL-Erfüllung	Ideal-Erfüllung
F01		J/N		Erstellen eines Webservers mit Intel Edison			
F02		W		Monitoransteuerung via DisplayLink			
F03		F		Anzeigen von Uhrzeit und Datum auf ext. Monitor	Uhrzeit	Uhrzeit und Datum	Uhrzeit, Datum, Wochentag, KW
F04		F		Anzeigen von Wetterdaten auf ext. Monitor	Temperatur	Temperatur, Wetterlage, Max. und Min. Temperatur	Temperatur, Wetterlage mit Grafik, Max. und Min. Temperatur
F05		W		Adaptive Anzeige von Wetterdaten und Uhrzeit für beliebige Orte		Aufruf über Argument in URL	
F06		W		Anzeigen von aktuellen Nachrichten auf ext. Monitor			
F07		W		Anzeigen von iCal-Kalenderdaten		Anzeige von öffentlichen Kalendern	Anzeigen von privaten Kalendern
F08		F		Kostenrahmen	< 300€	< 250€	< 200€

Anforderungsarten: J/N - Ja/Nein; F - Forderung; W - Wunsch; ***Konstruktionsphase:*** P - Prinzip; K - Konzept; E - Entwurf; A - Ausarbeitung

Ersetzt Ausgabe vom **15.04.2016** Version: **1**		**Ausgabe:** **Version:** **Blatt 1 von 1**	**25.04.2016** **2**

5 Blockschaltbild



Über das eingebaute WiFi-Modul kann der Intel Edison sich aktuelle Daten wie das lokale Wetter aus dem Internet herunterladen und in einen Webserver integrieren. Dieser wird vom Edison erstellt und direkt aufgerufen. Über einen DisplayLink-Adapter kann dem Intel Edison eine Grafikausgabe hinzugefügt werden, an dem ein Monitor betrieben wird.

Zusätzlich wäre die Nutzung verschiedener Sensoren denkbar, welche direkt am Breadboard des Intel Edison angeschlossen werden könnten.

6 Zeitplan

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft		**Zeitplan** für **Smart Mirror**								**MTB732** Sommersemester 2016 22.04.2016
		2016								**Jahr**
Nr.	**Aktivitäten**	März			April				Mai	**Monat**
		11	12	13	14	15	16	17	18	**Kalenderwoche**
1	Projektdefinition erstellen	X								
2	Erste Software auf Ubuntu erstellen		X	X						
3	Webserver erstellen				X					
4	Migration Software auf Edison					X	X	X		
5	Grafische Oberfläche auf Edison einrichten und Ansteuerung DiplayLink						X	X		Bemerkung: Umsetzung mit VNC-Viewer
6	Erweiterung des Smart Mirror Software							X	X	
7	Abschlusspräsentation								X	
		Präsentation: 06.05.2016								

7 Vorbereitung des Intel Edison

Aufgrund der speziellen Anforderungen an den Intel Edison, ist die Software XDK von Intel nicht zur Installation geeignet. Um unter anderem die Displayausgabe über den DisplayLink-Adapter zu realisieren, muss ein alternatives Linux mit zusätzlichen Kernel-Treibern aufgespielt werden. Dadurch geht leider die Unterstützung durch die Software XDK komplett verloren.

7.1 Installation Linux

Als Linux-Version kommt „Ubilinux“ von www.emutexlabs.com zum Einsatz. Dieses beinhaltet auf Debian-Basis bereits zahlreiche Kernel-Patches, welche zur Inbetriebnahme des Smart Mirror benötigt werden. Das Flashen des Intel Edison erfolgt von einem Ubuntu-Computer, da Windows nur eingeschränkt zum Flashen fähig ist.

Nach der Installation von Ubilinux, kann man die Konfiguration über die serielle Schnittstelle vornehmen. Die Anmeldung erfolgt über die standardmäßig eingestellten Daten

```
ubilinux login: root
Password: edison
```

7.2 WLAN-Konfiguration

Die WLAN-Einstellungen müssen ebenfalls über die Konsole vorgenommen werden. Dazu wird zunächst die Konfigurationsdatei mit dem Konsoleneditor „nano“ aufgerufen:

```
root@ubilinux:~# nano /etc/network/interfaces
```

Hier muss die Zeile

```
auto usb0
```

mit einem ‚#‘ auskommentiert und die Zeile

```
#auto wlan0
```

einkommentiert werden, damit der Intel Edison beim Booten automatisch eine WLAN-Verbindung erstellt.

Das WLAN-Netz kann in den Zeilen

```
      wpa-ssid WLAN-Name
      wpa-psk WLAN-Passwort
```

konfiguriert werden, bevor der Texteditor verlassen wird.

Nach einem Neustart des Edison kann nun der WLAN-Status abgefragt und die IP-Adresse ausgelesen werden:

```
root@ubilinux:~# reboot
root@ubilinux:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:14354 (14.0 KiB)  TX bytes:14354 (14.0 KiB)

wlan0     Link encap:Ethernet  HWaddr 78:4b:87:a2:8f:d5
          inet addr:192.168.0.35  Bcast:192.168.255.255  Mask:255.255.0.0
          inet6 addr: fd00::7a4b:87ff:fea2:8fd5/64 Scope:Global
          inet6 addr: fe80::7a4b:87ff:fea2:8fd5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:64507 errors:0 dropped:0 overruns:0 frame:0
          TX packets:39653 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:84620766 (80.7 MiB)  TX bytes:2802084 (2.6 MiB)
```

7.3 Paketverwaltung

Um benötigte Tools für die Display-Ausgabe und den Webserver installieren zu können, ist es notwendig, zunächst Speicher freizugeben, indem die folgenden vorinstallierten Packages deinstalliert werden:

python	python2.7-mini	lynx-cur	g++-4.7
python-dev	perl	g++	gcc
cmake	python-minimal	perl-base	gcc-4.7
cmake-data	python2.7	perl-modules	
curl	python2.7-dev	lynx	

Das Deinstallieren wie auch die Installation neuer Pakete erfolgt mit dem bash-basierten package-manager ‚apt-get'. Dazu wird der purge-Befehl angewendet und sämtliche zu deinstallierenden Pakete auf einmal ausgewählt:

```
root@ubilinux:~# apt-get purge cmake cmake-data curl python python-dev
python-minimal python2.7 python2.7-dev python2.7-mini perl perl-base perl-
modules lynx lynx-cur g++ g++-4.7 gcc gcc-4.7
```

Nun können die neuen Pakete installiert werden, welche zur Verwendung für den Smart Mirror benötigt werden. Dazu wird zunächst die Datenbank aktualisiert:

```
root@ubilinux:~# apt-get update
root@ubilinux:~# apt-get upgrade
```

Bevor mit der Installation des xfce4-Pakets für die grafische Oberfläche begonnen werden kann, muss zunächst die automatische Installation empfohlener Pakete abgeschaltet werden, um keinen unnötigen Speicherplatz zu verbrauchen. Dazu wird eine neue Datei erstellt und mit dem nano-Editor geöffnet:

```
root@ubilinux:~# nano /etc/apt/apt.conf
```

Folgende Zeilen müssen zur Abschaltung in die Konfigurationsdatei kopiert werden:

```
APT::Install-Recommends "0";
APT::Install-Suggests "0";
```

Nun können die Pakete für eine minimale grafische Oberfläche installiert werden:

```
root@ubilinux:~# apt-get install xserver-xorg xserver-xorg-core xfonts-base xinit x11-xserver-utils
root@ubilinux:~# apt-get install xfwm4 xfce4-panel xfce4-settings xfce4-session xfce4-terminal xfce4-utils xfdesktop4 xfce4-taskmanager tango-icon-theme
```

Bei Fehlermeldungen während der Installation müssen möglicherweise die Befehle möglicherweise mit dem Zusatz

```
--fix-missing
```

wiederholt werden, ebenfalls ist möglicherweise eine Paketreparatur nötig:

```
root@ubilinux:~# apt-get --fix-broken
```

7.4 Installation des Webservers

Die Anzeige des SmartMirrors soll über den Browser ablaufen, welcher auf einen lokalen Webserver zugreift. Die Auswahl fiel hier auf „nginx“, dieser Webserver braucht etwa 10MB Speicherplatz und ist damit für den geringen Speicher des Edison bestens geeignet.

```
root@ubilinux:~# apt-get install nginx
```

Nach der Installation kann der Server über die folgende Eingabe gestartet werden.

```
root@ubilinux:~# service nginx restart
```

Zur Konfiguration des Webservers öffnet man dessen Konfigurationsdatei mit dem Texteditor „nano“

```
root@ubilinux:~# nano /etc/nginx/nginx.conf
```

Hier ändert man zuerst die Anzahl der worker_processes.

```
worker_processes 1;
```

Weiter unter müssen zwei Zeilen mit einem ‚#‘ auskommentiert werden und ein paar Zeilen eingefügt werden:

```
        #include /etc/nginx/conf.d/*.conf;
        #include /etc/nginx/sites-enabled/*;
server{
        listen 80;
        server_name _;
        location / {
                root /home/www/;
                index index.html;
                }
        }
}
```

Hier wird auch festgelegt aus welchem Verzeichnis heraus der der Webserver seine Daten lädt und welches die Startseite ist. In diesem Fall ist es der Ordner „www“ auf der home-Partition und als

Startseite wird die Datei index.html geladen. Nun kann man mit Strg+X speichern und Beenden. Abschließend muss noch der Order www auf der home-Partition erstellt werden.

```
root@ubilinux:~# mkdir /home/www/
```

7.5 Installation des Web-Browsers

Die Anzeige erfolgt über einen Web-Browser, dieser muss CSS3 unterstützen und möglichst wenig Speicher belegt. Aus diesem Grund haben wurde der Browser „Midori“ gewählt.

```
root@ubilinux:~# apt-get install midori
```

Zusätzlich wird noch das Paket „unclutter“ benötigt um später, wenn der Browser im Vollbildmodus läuft, den Mauszeiger ausblenden zu können.

```
root@ubilinux:~# apt-get install unclutter
```

Nach der Installation kann man über die grafische Oberfläche den Browser öffnen und auf die Testseite des Webservers zugreifen.

7.6 Einrichten einer VNC-Verbindung

Aufgrund fehlender Hardware wird zum Test der grafischen Oberfläche das Tool VNC-Server eingesetzt, welches einen Remotezugriff über LAN auf den Intel Edison ermöglicht.

Dazu wird auf einem PC die Software ‚VNC Viewer‘ installiert, auf dem Intel Edison wird der VNC-Server wie folgt eingerichtet:

```
root@ubilinux:~# apt-get install vnc4server
```

Nach der Paketinstallation muss der Server mit Einstellungen für die Displaygröße und die Farbtiefe eingerichtet werden, wobei die Konfiguration ein Passwort verlangt.

```
root@ubilinux:~# vnc4server -geometry 1024x768 -depth 24
Password: Passwort nach Wahl
Verify: Passwort erneut eingeben
```

Nun muss in den VNC-Einstellungen eine xfce4-Session gestartet werden. Dazu wird aus dem Home-Verzeichnis heraus die VNC-Konfigurationsdatei im Nano-Editor geöffnet:

```
root@ubilinux:~# nano .vnc/xstartup
```

In diese wird nun die fett markierte Zeile eingefügt:

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
xfce4-session
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
x-window-manager &
```

Nach der einmaligen Konfiguration kann der Server jederzeit gestartet und mit dem ‚VNC Viewer' darauf zugegriffen werden:

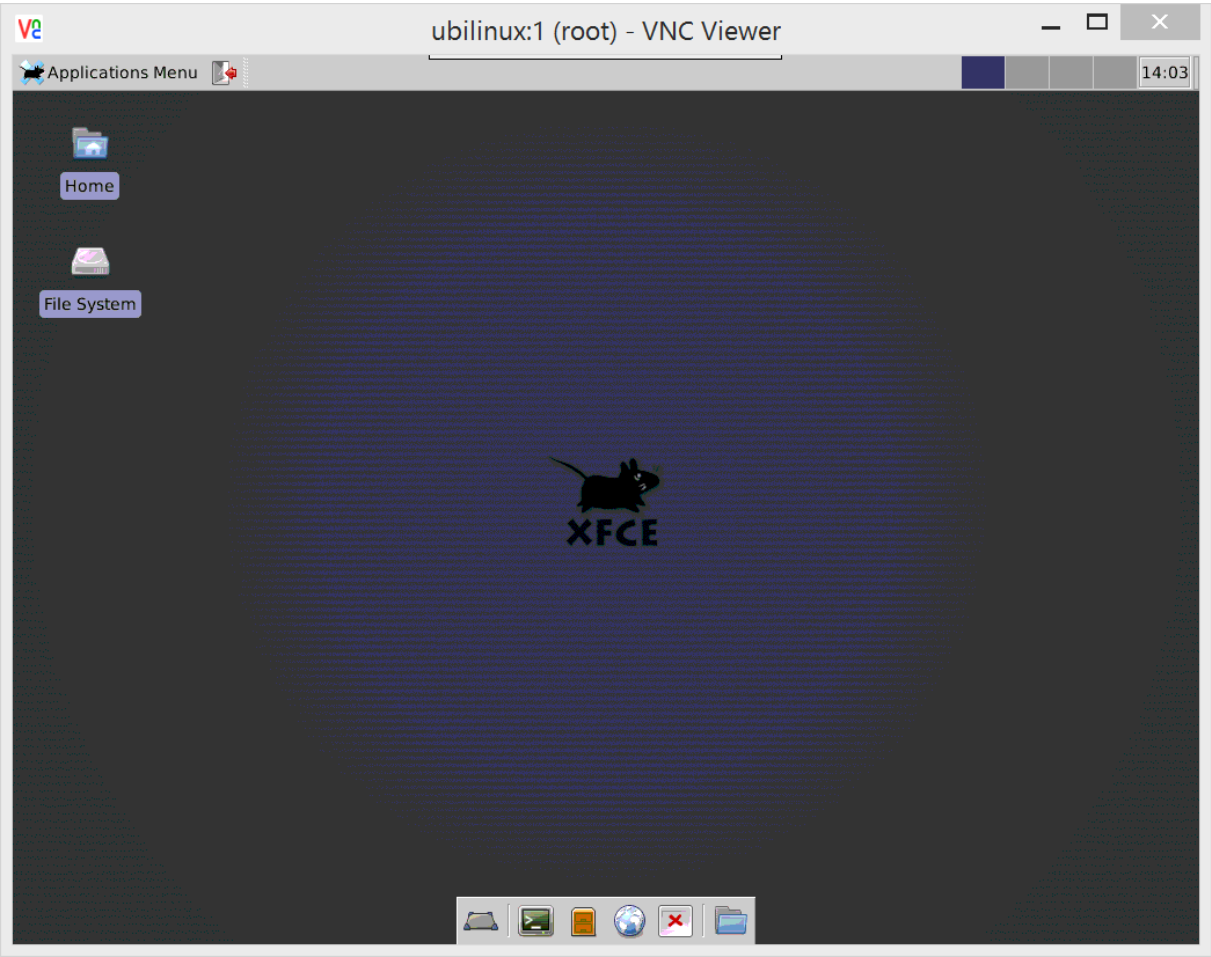
```
root@ubilinux:~# vnc4server
```

Dieser Befehl liefert eine Ausgabe zurück, welche ähnlich der Folgenden ist:

```
New 'ubilinux:2 (root)' desktop is ubilinux:2
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/ubilinux:2.log
```

Der ‚VNC Viewer' kann sich nun über die IP-Adresse des Edison und die Nummer der Desktopinstanz (aus Ausgabe des Serverstarts) auf den Server verbinden, wobei vor dem Verbindungsaufbau das Passwort des Edison eingegeben werden muss.

Wenn die Installation und Konfiguration korrekt und ohne Fehler durchgeführt wurde, erscheint der rudimentäre Desktop:



7.7 Erstellen des Autostartes

```
root@ubilinux:~# nano /etc/vncserver/vncservers.conf
```

Erstellen einer VNC-Session für den Benutzer root.

```
VNCSERVERS="1:root"
VNCSERVERARGS[1]="-geometry 1024x768 -depth 16"
```

Speichern und Beenden mit Strg+X. Falls zusätzliche Sessions benötigt werden können diese einfach in die Datei eingetragen werden.

Erstellen der Start Sequenz, diese startet jede Session welche in „vncserver.conf“ aufgeführt ist.

```
nano /etc/init.d/vncserver
```

Nach öffnen der Datei muss folgendes Skript eingefügt werden:

```
#!/bin/bash
unset VNCSERVERARGS
VNCSERVERS=""
[ -f /etc/vncserver/vncservers.conf ] && . /etc/vncserver/vncservers.conf
prog=$"VNC server"
start() {
 . /lib/lsb/init-functions
 REQ_USER=$2
 echo -n $"Starting $prog: "
 ulimit -S -c 0 >/dev/null 2>&1
 RETVAL=0
 for display in ${VNCSERVERS}
 do
 export USER="${display##*:}"
 if test -z "${REQ_USER}" -o "${REQ_USER}" == ${USER} ; then
 echo -n "${display} "
 unset BASH_ENV ENV
 DISP="${display%%:*}"
 export VNCUSERARGS="${VNCSERVERARGS[${DISP}]}"
 su ${USER} -c "cd ~${USER} && [ -f .vnc/passwd ] && vncserver :${DISP} ${VNCUSERARGS}"
 fi
 done
}
stop() {
 . /lib/lsb/init-functions
 REQ_USER=$2
 echo -n $"Shutting down VNCServer: "
 for display in ${VNCSERVERS}
 do
 export USER="${display##*:}"
 if test -z "${REQ_USER}" -o "${REQ_USER}" == ${USER} ; then
 echo -n "${display} "
 unset BASH_ENV ENV
 export USER="${display##*:}"
 su ${USER} -c "vncserver -kill :${display%%:*}" >/dev/null 2>&1
 fi
 done
 echo -e "\n"
 echo "VNCServer Stopped"
}
case "$1" in
start)
start $@
;;
stop)
stop $@
;;
restart|reload)
stop $@
sleep 3
```

```
start $@
;;
*)
echo $"Usage: $0 {start|stop|restart}"
exit 1
esac
```

7.8 Laden der Daten

Am Einfachsten ist die Daten mit einem Linuxsystem und der Funktion „SSH-copy“.

```
user@PC:~# scp /Quellpfad/* -r root@IP_edison:/Zielordner
```

```
user@PC:~# scp ~/Webseite/* -r root@IP_edison:/home/www/
```

Das „*“ am Ende des Quellpfades bedeutet das alle Dateien im Ordner kopiert werden, „-r“ bedeutet rekursiv also auch Unterordner werden mit kopiert.

Nach Abschluss des Kopiervorganges sollte man den Webserver nochmals neustarten.

```
root@ubilinux:~# service nginx restart
```

Mit einem Windowssystem muss man den „Putty Secure Copy Client“ installieren. Dieser wird dann über die Eingabeaufforderung in Windows benutzt und hat eine ähnliche Syntax.

7.9 Browser in den VNC-Session Start einbinden

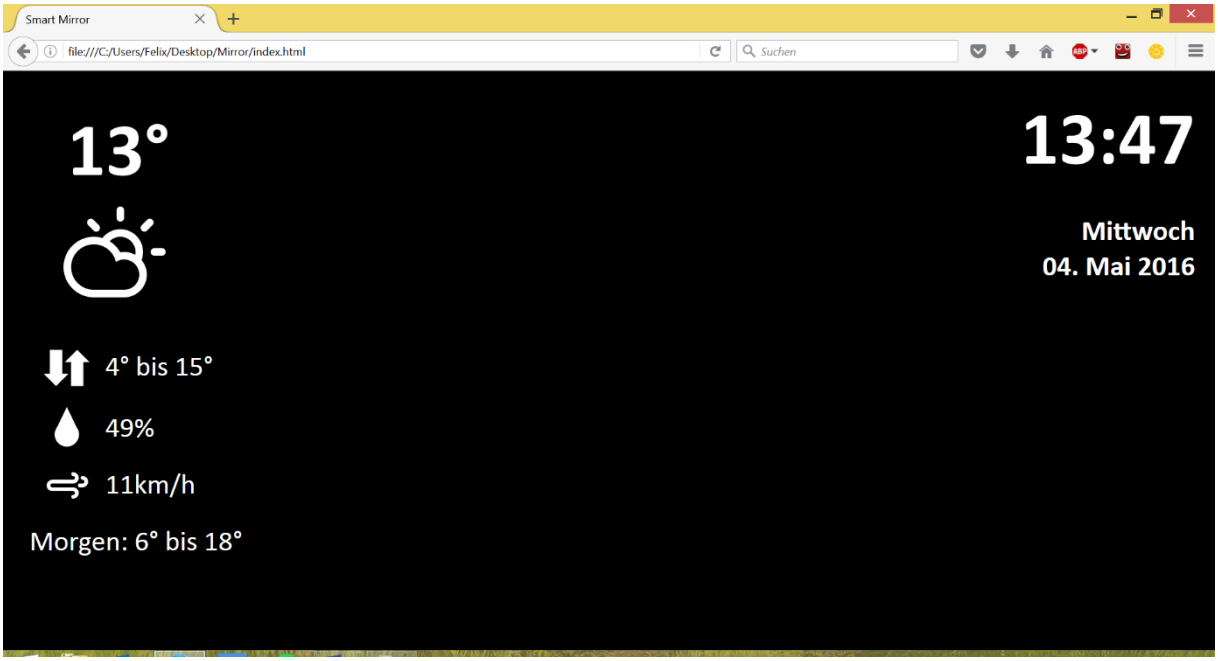
Es muss noch der Browser in den Start der VNC-Session eingebuden werden.

```
root@ubilinux:~# nano .vnc/xstartup
```

```
xset -dpms #Energiesparmodus deaktivieren
xset s off #Bildschirmschoner deaktivieren
xset s noblank #Bildschirmabschaltung deaktiven
sleep 5s &
midori --execute Fullscreen -a localhost #Midori im Vollbild öffen
```

8 Programmierung Smart Mirror

Die Grundfunktionen des Smart Mirror sind die Anzeige von Uhrzeit und Datum sowie des aktuellen Wetters an einem festgelegten Ort. Der Aufbau basiert auf einer HTML-Seite, das Design erfolgt mit Hilfe von CSS-Dateien. Außerdem werden nicht-statische Anzeigen in JavaScript ausgeführt.



8.1 index.html

Die erstellte HTML-Seite teilt sich in einen Header- und einen Body-Bereich auf.

Im Header erfolgt die Initialisierung der Seite, die CSS-Skripte und die API-Dokumente für die Anzeige des Wetters werden importiert, außerdem erfolgt die automatische Aktualisierung im 15-Sekunden-Takt aus dem Header-Bereich heraus:

```
<meta http-equiv="refresh" content="15" />
```

Der Body-Bereich besteht aus einer Tabelle mit drei Spalten. In der linken Spalte wird das Wetter angezeigt, die rechte Spalte zeigt die Uhrzeit und das Datum. Die mittlere Spalte dient lediglich als Platzhalter und bietet derzeit keinen Inhalt.

In den äußeren Spalten erfolgt derzeit der Aufruf der entsprechenden JavaScript-Dokumente und im Falle des Wetters die Ausgabe der gelesenen Daten. Die Uhrzeit und das Datum werden direkt aus der JavaScript-Datei ausgegeben.

```
<table class="Tabelle">
      <tr>
            <td class="ZelleAussen">
                  <script src="js/weather.js"></script>

                  <p class='temperature'></p>
                  <div class="climate_bg"></div>
                  <img class="dropicon" src="images/Droplet.svg">
                  <div class="humidity"></div>
                  <img class="windicon" src="images/Wind.svg">
                  <div class="windspeed"></div>
            </td>
            <td class="ZelleInnen"></td>
            <td class="ZelleAussen">
                  <script type="text/javascript" src="js/clock.js"></script>
            </td>
      </tr>
</table>
```

8.2 clock.js

Das File „clock.js“ beinhaltet den Aufruf und die Ausgabe von aktueller Uhrzeit und Datum. Dazu wird nach Aufruf des Scripts die Funktion „get_date()“ aufgerufen. In dieser wird die aktuelle Systemzeit ausgelesen und in eine Variable gespeichert:

```
var get_date = new Date;
```

Die ausgelesene Zeit hat den folgenden Aufbau:

```
Wed May 04 2016 09:38:11 GMT+0200
```

Diese Ausgabe wird nun in einer Variablen vom Typ string gespeichert und daraufhin an den Leerzeichen in ein Array gesplittet:

```
var date_string = String(get_date);
date_split = date_string.split(" ");
```

Im Folgenden werden für Uhrzeit und Datum jeweils Strings erstellt, die direkt in HTML ausgegeben werden können. Dazu werden die einzelnen Array-Felder in der entsprechenden Reihenfolge zusammengesetzt, wobei beispielsweise der Wochentag über eine Lookup-Table in das deutsche Wort gewandelt werden. Die Uhrzeit soll auf dem Smart Mirror nur mit Stunden und Minuten angezeigt werden, weshalb diese über die Funktionen „getHours“ und „getMinutes“ abgerufen werden, wobei den Minuten noch eine führende NULL angehängt wird, sofern die Minutenzahl einstellig ist:

```
date_german = "<p class = 'dateid'>"+get_wochentag(date_split[0])+"<br>"
  +date_split[2]+". "+get_monat(date_split[1])+" "+date_split[3]+"</p>";

time_german = "<p class = 'clockid'>"+get_date.getHours()+":"
  +(get_date.getMinutes()<10?'0':'')+get_date.getMinutes()+"</p>";

window.document.write(time_german);
window.document.write(date_german);
```

8.3 weather.js

Das JavaScript-File weather.js sammelt über die frei verfügbare API „Simpleweather“ Wetterdaten für einen beliebigen Ort und gibt diese an die HTML-Datei weiter, welche die Ausgabe übernimmt.

Die Wetterfunktion wird dabei mit einem Ort aufgerufen, der entweder fest eingegeben oder über eine Ortung mit der geolocation -Funktion ermittelt wird. In der Funktion „loadWeather()“ werden nun Informationen über die Wetterlage gesammelt.

Über die folgende Funktion werden die Wetterdaten im 60-Sekunden-Takt aktualisiert:

```
$(document).ready(function(){
	setInterval(getWeather,60000);
})
```

8.4 weather.css

Die Datei „weather.css“ wird im Header von „index.html“ geladen und ist für das Layout der Wetterdaten zuständig.

Wird beispielsweise die Temperatur aus HTML mit

```
<p class='temperature'></p>
```

ausgegeben, so ergeben sich deren grafische Eigenschaften aus der Klasse „temperature“ im weather.css-file:

```
.temperature {
	position: absolute;
	top: 0px;
	left: 80px;
	margin: 2% 0%;
	text-align: center;
	font-size: 80px;
	font-weight: bold;
}
```

Aus dem css-File lässt sich erkennen, dass die Position der Temperatur (wie auch die der restlichen Wetterdaten) absolut festgelegt ist (0 Pixel von oben, 80 Pixel von links) und um die Temperatur zusätzlich ein Rand gelegt wird, in dem kein anderes Objekt angezeigt wird, in diesem Fall ist dieser 2% der Bildschirmgröße nach oben und 0% nach links. Außerdem werden die Textausrichtung (mittig) und die Textgröße (80 Pixel, fett gedruckt).

Allgemeine Informationen, wie der schwarze Hintergrund und die weiße Schriftart werden in der Datei main.css festgelegt. Generell gilt, dass Informationen aus css-Files überschrieben werden können, würde also in der Klasse „temperature“ eine rote Schriftfarbe festgelegt werden, so würde dies die vorher gespeicherte Information aus der Einstellungen für den Body (main.css) ersetzt und die Temperatur rot dargestellt werden.

Analog erfolgen grafische Einstellungen für die Darstellung von Uhrzeit und Datum sowie allgemeine Einstellungen für die Grafikausgabe in den Files „main.css“ und „clock.css“.

9 Stückliste

Bauteilbezeichnung	Erläuterung
Intel Edison	CPU-Einheit zur Erstellung und Darstellung des Webservers
DisplayLink-Adapter	USB-DVI-Adapter für die Ausgabe einer grafischen Oberfläche über den intel Edison
USB OTG Adapterkabel	Adapterkabel, um DisplayLink-Einheit an Intel Edison anzuschließen
Aktiver USB-Hub	Der DisplayLink-Adapter zieht mehr Strom als der Intel Edison über die USB-Schnittstelle zur Verfügung stellen kann. Deshalb wird ein aktiver Hub benötigt, der den benötigten Strom zur Verfügung stellen kann.
12V-Steckernetzteil	Der Intel Edison kann mit dem Breakout-Board entweder über USB oder ein externes Netzteil (7-15 V) betrieben werden. Bei Verwendung der Schnittstelle als OTG entfällt aufgrund der USB-Spezifikation jedoch die Möglichkeit des Betriebs über USB weshalb ein externes Netzteil benötigt wird.
Monitor	Für die Bildausgabe wird ein Monitor mit DVI-Anschluss benötigt

10 Fazit und Ausblick

Trotz anfänglicher Schwierigkeiten bei der Implementierung einer Displayausgabe auf dem Intel Edison, konnte ein stabil laufendes System erstellt werden.

Wegen zu langer Lieferzeiten konnte das System leider nicht mit dem DisplayLink-Adapter betrieben werden, derzeit ist lediglich ein Zugriff über VNC oder der Aufruf des Webservers von einem anderen Rechner innerhalb des Netzwerks möglich.

Die SmartMirror-Seite wurde in Anbetracht der geringen Zeit bereits recht weit ausgebaut und übertrifft in Teilen (Beispielsweise der Wettervorhersage für den nächsten Tag) bereits die Idealerfüllung aus der Anforderungsliste. Dabei besteht an manchen Stellen noch Verbesserungsbedarf, beispielsweise wäre ein Laden der Seite wünschenswert, bei dem der Framebuffer erst nach Abschluss des Ladevorgangs überschrieben wird. Dadurch wären Leerzeiten auf der Seite vermeidbar.

Für eine mögliche Weiterführung des Projektes, wäre neben der Inbetriebnahme der Hardware, eine Erweiterung der SmartMirror-Software wünschenswert. Neben den beschriebenen Inhalten aus der Anforderungsliste wäre die Einführung einer Sensortechnik vorstellbar, welche beispielsweise den Monitor nur aktiviert, wenn sich eine Person im Raum befindet.