Ausführliches Webserver-Beispiel

Im folgenden soll eine vollständige Visualisierung eines Gebäudes erstellt werden. Diese Visualisierung ist im Foyer des Enertex-Gebäudes auf einem 27-Zoll HP – Android Panel aktiv.

Die vorliegende Visualisierung ist im Rahmen eines Praktikums entstanden und erklärt die wesentlichen Punkte bei der Programmierung und Erstellung der Webseiten.

Im Eingangsbereich bei der Enertex® Bayern GmbH

Abbildung 61: HP Android basierte Visualisierung

Ziel ist es,Statusanzeige mit einer zentralen Schaltmöglichkeit der Haussteuerung, sowie Wetterinformationen auf mehreren Seiten zu visualisieren.

Beginnen wir mit der Hauptseite, welche am Ende wie folgt dargestellt wird:

Hauptseite der Visualisierung

Abbildung 62: Hauptseite der Visualisierung

Zu Beginn wollen wir unseren Aufbau des im EibStudio einteilen, diese Einteilung ist für die Funktionalität der Anweisungen notwendig. Grundsätzlich gibt es verschiedene Sektionen die mit „[ ]“ gekennzeichnet sind. In diesen Sektionen werden wir Unterpunkte einführen, bei denen Schritt für Schritt mit auskommentierten (//) Beschreibungen, Variablen definiert, Webserver-Seiten hinzugefügt oder Anweisungen vergeben werden.

```
[MacroLibs]
/Bibliotheken/InternEnertexWebV3.lib
[ETS-ESF]
//
[WebServer]
//
[Macros]
//
[EibPC]
//// Deklarationen
//// Funktionen
```

Um nun unsere erste Seite im *EibPC* anzulegen, wird eine *SeitenID* benötigt. Diese sollte möglichst die Darstellungskomponente der Seite Spiegeln und wird in die Sektion [EibPC] geschrieben.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
```

Anfänglich muss man sich neben Namen (*Wetterseite*) und Kategorie (*Allgemein*) der Hauptseite, ebenso im Klaren sein, in welche Designrichtung der Webserver gehen soll. Wir entschieden uns für die Designvariante *black* und als optische Abrundung wurde eine Hintergrundgrafik implementiert, welche das Firmenlogo darstellt (vgl. Abbildung 62). **Achtung**: „ Hintergrundgrafiken werden nicht skaliert“. Bei einer Anordnung von 6 x 8 Elementen beträgt die perfekte Skalierung 1025x801 Pixel. Die Grafik muss dabei so angeordnet werden, dass die am Ende in der Visu freibleibende Fläche in der Mitte der Seite mit dem gewünschten Firmenlogo steht.

Ist das Hintergrundbild soweit bearbeitet, kann man es in die Sektion [WebServer] einbinden:

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]

[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
```

Damit der EibPC die Hintergrundgrafik auch unter dem angegebenen Pfad findet, muss diese exportiert werden. Dies geschieht mit Hilfe des *EibStudio* unter dem Reiter *EibPC* und dem Menüpunkt *Dateitransfer Dialog.*

Bilder hochladen

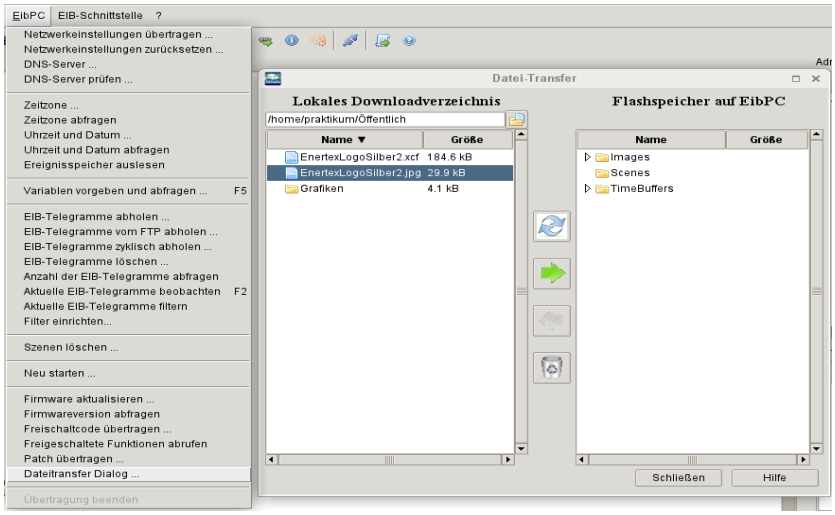

Abbildung 63: Dateitransferdialog

Mit der Möglichkeit, eigene Bilder in den Enertex EibPC zu laden, können Sie diese auch zur Gestaltung einer eigenen Anzeige für die Header und Footer Grafik nutzen. Zunächst müssen Sie die Grafiken wie beschrieben vorab in den Enertex® EibPC hochladen.

In unserem Beispiel setzten wir header und footer jeweils auf (0) und verzichteten somit auf Kopf- und Fußzeile, sowie der Möglichkeit, eigene Bilder dafür zu verwenden.

Nachdem die Designarbeiten soweit abgeschlossen sind, geht es an die Implementierung der Anzeigebutton. Diese können je nach belieben mit Gruppenadressen und Icons (Übersicht s. Seite 289) versehen werden. Viele der Button arbeiten mit vorgefertigten Makros, welche in den Enertex Bibliotheken vorhanden sind.

Grundsätzlich unterscheidet man **lokale** und **globale** Elemente bei der Anzeige. Global bedeutet, dass das Element auf verschiedenen Seiten verwendet werden kann, aber nur ein einziges mal erzeugt wurde. Ein Zugriff bzw. eine Veränderung des Elements durch das Anwendungsprogramm ist daher für alle Seiten gleich. Hingegen kann ein lokales Element nur in einer Seite verändert werden. Vom Design sind lokale und globale Elemente identisch, wobei erstere durch den Zusatz „p“ (page) gekennzeichnet sind.

Einige der Elemente (*z.B **shifter**(ClockID)[CLOCK] $Uhrzeit$* und ***shifter**(DateID)[DATE]$Datum$)* wurden auf unserer Hauptseite als Global initialisiert, da diese als durchgängige Informationen auf allen Seiten dienen sollen. Für diese Button müssen Webelement IDs initialisiert werden, welche ebenso in die Sektion [EibPC] geschrieben werden.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WindID = 1
WolkenID = 2
ClockID = 3
DateID = 4
```

Um im späteren Verlauf schneller arbeiten zu können empfiehlt es sich, funktionsorientierte ID Bezeichnungen zu wählen.

Die lokalen Elemente werden direkt in der Webserver-Konfiguration mit ID´s von (0 bis 39) erstellt, sind aber ebenso wie die globalen Elemente vorerst ohne Funktionalität.

Der Unterschied zwischen Button und Shifter ist rein die Größe der Elemente. Während ein Button einfache Breite besitzt, wird der Shifter mit doppelter Breite dargestellt.

Konfiguriert werden diese Elemente wie folgt:

shifter(ID)[Grafik1,Grafik2,Grafik3,Grafik4]$Text$	button(ID)[Grafik] $Text$
pshifter(ID)[Grafik1,Grafik2,Grafik3,Grafik4]$Text$	**p**button(ID)[Grafik] $Text$

Grafik 2 bis Grafik 4 sind optional

Nun wissen wir, wie die Elemente in die Sektion [WebServer] auf unserer Seite eingebunden werden. Anordnen lassen sie sich auf verschiedene Art und Weise als Schachbrettmuster. Durch die feste Vorgabe von eingebauten Icons, Längen und Variablen kann die Konfiguration ohne grafischen Aufwand rein textbasiert erfolgen.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(31)[TEMPERATURE]$Außentemperatur$\\
  pshifter(17)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  pbutton(5)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(15)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(14)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(13)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$
```

Kompilieren wir unser bisheriges Programm, so ergibt sich folgende Darstellung im Webserver.

Abbildung 64: Elemente im Schachbrettmuster

Die Anordnung der Elemente wird so dargestellt wie gewollt. Allerdings ist unser Logo auf der Hintergrundgrafik nicht zu erkennen, da es teilweise von den Elementen verdeckt wird, wir es aber auch für eine Darstellung von 6x8 Feldern mittig skaliert haben.

Um noch etwas fürs Auge zu schaffen, aber auch um mehr Informationen zu bekommen, binden wir zwei grafische Wettervorhersagen mit ein. Diese sollen unsere Hintergrundgrafik seitlich einrahmen und über der geschaffenen globalen Fußzeile aus Uhrzeit und Datum stehen. Zur Einbindung verwenden wir folgende Funktion:

picture(ID)[Höhe,Typ]($Beschriftung$,$www-LINK$)

Bei den ausgewählten Links ist möglichst darauf zu achten, dass diese unabhängig von Zeit und Datum sind. So aktualisieren die Seiten sich selbständig und es benötigt keinen weiteren Code, um einen refresh zu erzeugen.

Beispiele:

Gut:

$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$

Schlecht:

$http://www.wetteronline.de/?
daytime=day&diagram=true&fcdatstr=**20140916**&iid=DL&pid=p_city_local&sid=Pictogram$

Nach Auswahl der beiden Links, müssen wir unsere picture Funktionen ebenfalls in der Sektion [WebServer] einbinden.Um die seitliche Einrahmung des Logos zu erreichen, müssen zwischen den picture Funktionen zwei **none** Elemente eingesetzt werden. Diese werden im Webserver als Platzhalter verwendet und in keiner Weiße dargestellt.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(1)[TEMPERATURE]$Außentemperatur$\\
  pshifter(2)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  pbutton(3)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(4)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(5)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
picture(7)[DOUBLE,CENTERGRAF]($Durchschnittliche Tagestemperaturen$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$) \\
  none \\
  none \\
  picture(8)[DOUBLE,CENTERGRAF]($3 – Tagesvorhersage$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-3t_webL.jpg$)
// Vierte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(9)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$
```

Abbildung 65: Die Konfiguration der Buttons

Die Anordnung unserer Anzeige Button ist somit vorerst abgeschlossen und wir können uns der Funktionalität widmen. Dazu müssen wir im Code in der Sektion [Macros] abreiten. Hier lässt sich mit Hilfe der **Makro-Bibliothek(*InternEnertexWebV3*)** für jeden Button/Shifter die passende Funktionalität finden.

Beispiel Datumsanzeige global:

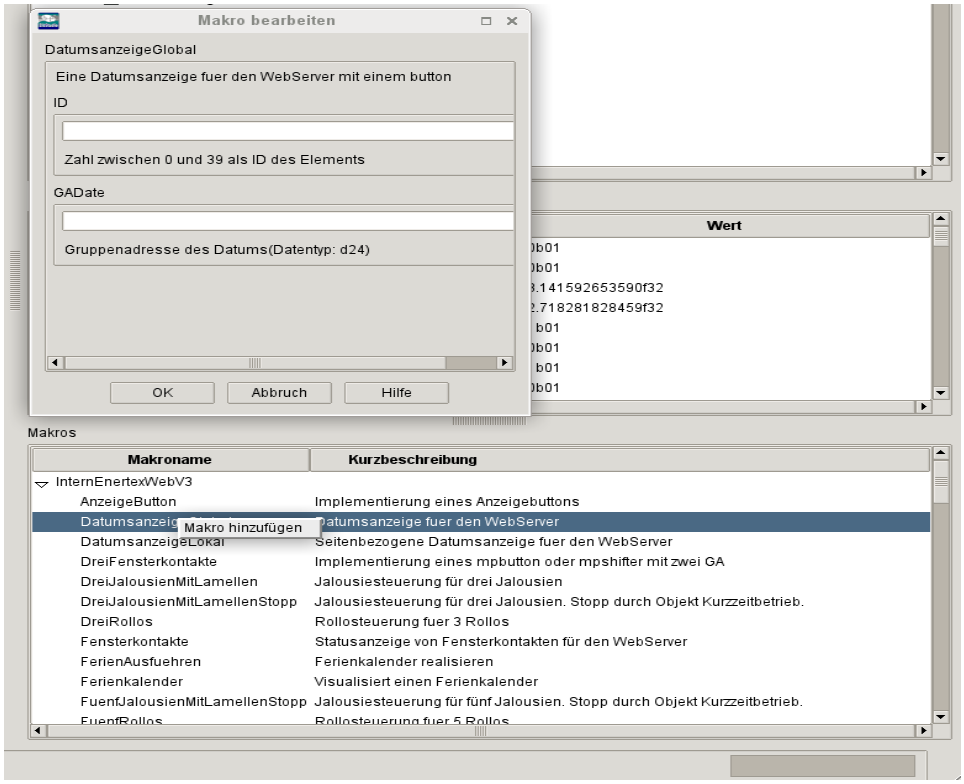

Abbildung 66: Makroauswahl

Wird für jeden Button/Shifter das richtige Macro angewendet und die jeweiligen Felder entsprechend ausgefüllt, so entstehen in der Sektion [Macros] folgende Zeilen

```
[Macros]
// Makros für die Wetterseite in der Kategorie Allgemein:
DatumsanzeigeGlobal(DateID,"Datum-7/0/0")
UhrzeitanzeigeGlobal(ClockID,"Uhr-7/0/1")
WindAnzeigeButtonGlobal(WINDID,"Wind-14/6/0")
MinMaxTemperaturAnzeigeButtonLokal(2,AllgemeinWetterseiteID,"Außentemperatur-14/6/3")
LichtAnzeigeButtonLokal(4,AllgemeinWetterseiteID,"Licht-14/6/4")
SonnenAufUntergangAnzeigeShifterLokal(5, AllgemeinWetterseiteID)
OnlineAnzeigeButtonLokal(9,AllgemeinWetterseiteID)
```

Nun ist allen Anzeige Button eine Funktion zugeteilt, bis auf

```
pbutton(1)[TEMPERATURE]$Außentemperatur$
pbutton(3)[OKCIRCLE]$Status HG$
pbutton(6)[OKCIRCLE]$Status NG$
```

Die ID´s (3) und (6) werden erst im späteren Verlauf belegt. Der *pbutton(1)* jedoch, soll ein dynamisches Icon erhalten, das sich je nach Außentemperatur einfärbt und diese auch anzeigt. So soll blau für kalt, grau für mäßig, dunkel rot für angenehm warm und rot für heiß stehen.

Dies realisieren wir durch die Funktion:

```
pdisplay(ID, Text, Icon, State, TextStil, PageID, [Mbutton])
```

Hiermit werden *pbutton* oder *pshifter* angesprochen

Zuerst muss die Unterteilung der verschiedenen Zustände angelegt werden. Wir entschieden uns für:

```
>= 27 °C = heiß
< 27 °C und >= 18 °C = angenehm warm
< 18 °C und >= 5 °C = mäßig
< 5 °C = kalt
```

Je nach Außentemperatursensor kann der Rückgabewert des Sensors variieren. Unsere Wetterstation liefert den Datenwert **f16** zurück. Somit gilt: 0f16 = 0°C

Die Zustandsanweisungen müssen nun noch in if-Anweisungen im Sektor [EibPC] verpackt werden und mit pdisplay übergeben werden.

```
[EibPC]
//// Funktionen
/// Temperaturanzeige dynamisch
if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=27f16 then
pdisplay(1,"Außentemperatur-14/6/3",TEMPERATURE,BRIGHTRED,BLINKRED,AllgemeinWetterseiteID)
endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=18f16 and "Außentemperatur-14/6/3"
<27f16  then pdisplay(1,"Außentemperatur-
14/6/3",TEMPERATURE,DARKRED,GREEN,AllgemeinWetterseiteID) endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3">=5f16 and "Außentemperatur-14/6/3"
<18f16  then pdisplay(1,"Außentemperatur-
14/6/3",TEMPERATURE,DISPLAY,GREY,AllgemeinWetterseiteID) endif


if change("Außentemperatur-14/6/3") and "Außentemperatur-14/6/3" < 5f16  then
pdisplay(1,"Außentemperatur-14/6/3",TEMPERATURE,ACTIVE,BLINKBLUE,AllgemeinWetterseiteID) endif
```

Nun „weiß“ der *pbutton(1)* wie er sein Icon bei der entsprechenden Außentemperatur darstellen soll.

Um auch der letzten Anzeige eine Funktionalität zu geben fehlt uns folgender Button:

```
button(WolkenID)[WEATHER]$Aktuelles Wetter$
```

Hier wollen wir eine Sonneneinstrahlungsabhängige (*Luxwert*) Anzeige der aktuellen Bewölkung implementieren. Es wird unterschieden in Sommerzeit (Monat Mai bis Monat September) und Winterzeit (Monat Oktober bis Monat März), da der Sonnen-Höhenwinkel (Elevation) und somit die Intensität der Einstrahlung variiert. Sollte es regnen, wird eine Regenmeldung ausgegeben.

Nun muss auch hier eine Einteilung der verschiedenen Zustände angelegt werden. Wir entschieden uns für folgende Einteilung:

Sommerzeit:
Einstrahlung <= 20000 Lux und dies länger als 1 Minute, dann bedeckt
20000 Lux < Einstrahlung <= 45000 Lux und dies länger als 1 Minute, dann wolkig
45000 Lux < Einstrahlung <= 70000 Lux und dies länger als 1 Minute, dann sonnig
70000 Lux < Einstrahlung und dies länger als 1 Minute, dann starker Sonnenschein
Winterzeit:
Einstrahlung <= 3500 Lux und dies länger als 1 Minute, dann bedeckt
3500 Lux < Einstrahlung <= 9000 Lux und dies länger als 1 Minute, dann wolkig
9000 Lux < Einstrahlung <= 15000 Lux und dies länger als 1 Minute, dann sonnig
15000 Lux < Einstrahlung und dies länger als 1 Minute, dann starker Sonnenschein

Achtung:
Sommerzeit und Winterzeit ergeben zwei Bedingungen in der if-Anweisung. Somit muss Sommerzeit und Winterzeit jeweils als Variable definiert werden. Da die Variablen entweder wahr oder falsch bzw. EIN oder AUS sein können definieren wir einen Zeitraum mit Hilfe der month(dd,mm) Funktion. Dies geschieht ebenfalls in der Sektion [EibPC] unterhalb der Webelement ID´s.

```
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WINDID = 1
WolkenID = 2
ClockID = 3
DateID = 4
//Variablen
Sommerzeit=month(01,05) and !month(30,09)
Winterzeit =month(01,10) and !month(30,04)
```

Nun realisieren wir die Anzeige mit den Funktionen:

webdisplay(ID,Text,Icon,Zustand,TextStil)
delay(Signal, Zeit)

Tragen wir nun unsere Bedingungen als if-Anweisung in die Sektion [EibPC] ein, ergibt sich folgender Code:

```
[EibPC]
//// Funktionen
/// Wetter
//Sommerzeit
if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==EIN) then webdisplay(WolkenID, $Es
regnet$,WEATHER,STATE6,GREY) endif

if( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" <= 20000f16,60000u64))
then webdisplay(WolkenID,$bedeckt$,WEATHER,STATE5,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 20000f16 and delay("Licht-
14/6/4" <= 45000f16,60000u64)) then webdisplay(WolkenID, $wolkig$,WEATHER,STATE4,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 45000f16 and delay("Licht-
14/6/4" <= 70000f16,60000u64)) then webdisplay(WolkenID, $sonnig$,WEATHER,DISPLAY,GREY) endif

if ( Sommerzeit == EIN and "Regenmeldung-14/6/1"==AUS  and  delay("Licht-14/6/4" > 70000f16,60000u64))
then webdisplay(WolkenID, $starker Sonnenschein$,WEATHER,BRIGHTRED,BLINKRED) endif
//Winterzeit
if (Winterzeit == EIN and "Außentemperatur-14/6/3" < 0.0 and  "Regenmeldung-14/6/1"==EIN) then
webdisplay(WolkenID, $Es schneit$,ICE,ACTIVE,BLINKBLUE) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" <= 3500f16,60000u64))
then webdisplay(WolkenID,$bedeckt$,WEATHER,STATE5,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 3500f16 and delay("Licht-
14/6/4" <= 9000f16,60000u64)) then webdisplay(WolkenID, $wolkig$,WEATHER,STATE4,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and "Licht-14/6/4" > 9000f16 and delay("Licht-
14/6/4" <= 15000f16,60000u64)) then webdisplay(WolkenID, $sonnig$,WEATHER,DISPLAY,GREY) endif

if (Winterzeit == EIN and "Regenmeldung-14/6/1"==AUS and delay("Licht-14/6/4" >15000f16,60000u64)) then
webdisplay(WolkenID, $starker Sonnenschein$,WEATHER,BRIGHTRED,BLINKRED) endif
```

Um die Seite zu vollenden, benötigen wir erst unsere zwei weiteren Seiten von Status Hauptgebäude und Status Nebengebäude.Diese Seiten sollen als zentrale Schaltmöglichkeiten dienen und anzeigen,ob und wenn ja in welcher Etage noch Fenster offen sind.Zudem werden hier Weiterleitungen in die entsprechende Etage eingebaut werden.

Diese Weiterleitungen fehlen uns ebenso auf der Hauptseite. Im Falle eines offenen Fensters oder einer leuchtenden Lampe, in einem beliebigen Raum im Gebäude, soll der jeweilige Status Button (*Status HG*) auf rot wechseln und aus dem √ ein **X** machen.

Wir binden also zwei Verlinkungen in Form von plink(ID)[Grafik] [PageSprungIndex] $Text$ in den Webserver mit ein, die auf (*Status HG)* und (*Status NG*) weiterleiten sollen.

```
[WebServer]
//Webelemente für die Wetterseite in der Kategorie Allgemein:
page(AllgemeinWetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$[$/upload/EnertexLogoSilber2.jpg$]
header(0)
footer(0)
// Erste Zeile
button(WINDID)[WIND]$Wind in km/h$ \\
  pbutton(1)[TEMPERATURE]$Außentemperatur$\\
  pshifter(2)[TEMPERATURE,TEMPERATURE]$Aussentemp.: Min und Max in °C$\\
  plink(10)[RIGHT][StatusHGID]$Zur Statusseite Hauptgebäude$\\
  pbutton(3)[OKCIRCLE]$Status HG$
// Zweite Zeile
pbutton(4)[WEATHER]$Licht in Lux$ \\
  button(WolkenID)[WEATHER]$Aktuelles Wetter$\\
  pshifter(5)[WEATHER,NIGHT]$Sonnenauf- und Sonnenuntergang$\\
  plink(11)[RIGHT][StatusNGID]$Zur Statusseite Nebengebäude$\\
  pbutton(6)[OKCIRCLE]$Status NG$
// Dritte Zeile
picture(7)[DOUBLE,CENTERGRAF]($Durchschnittliche Tagestemperaturen$,
$http://wetter.tagesschau.de/import/wetter-cms/vorhersagen/img/de-vs-tt_webL.jpg$) \\
  none \\
  none \\
  picture(8)[DOUBLE,CENTERGRAF]($3 – Tagesvorhersage$,$http://wetter.tagesschau.de/import/wetter-
cms/vorhersagen/img/de-vs-3t_webL.jpg$)
// Vierte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$\\
  pshifter(9)[INFO]$Online$\\
  shifter(DateID)[DATE]$Datum$

page(StatusHGID)[$Allgemein$,$Statusseite Hauptgebäude$]
design $black$
header(0)
footer(0)

page(StatusNGID)[$Allgemein$,$Statusseite Nebengebäude$]
design $black$
header(0)
footer(0)

[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
StatusHGID = 2
StatusNGID = 3
```

Die Verlinkung der neuen Seiten funktioniert nun bereits. Um diese optisch noch etwas zu verbessern verwenden wir in der [EibPC] Sektion die Funktion:

plink(ID, Text, Icon, Iconzustand, PageID, PageSprungIndex)

Damit lässt sich der Status des Icon von plink verändern.

```
[EibPC]

////Funktionen
// plinks Hauptseite
plink(10,$Zur Statusseite Hauptgebäude$,RIGHT,ACTIVE,AllgemeinWetterseiteID,StatusHGID)
plink(11,$Zur Statusseite Nebengebäude$,RIGHT,ACTIVE,AllgemeinWetterseiteID,StatusNGID)
```

Nachdem alles in die verschiedenen Sektionen geschrieben wurde und wir den Code erfolgreich kompiliert haben, ist unsere Webserver-Hauptseite wie in Abbildung 62 gestaltet.

Die Statusseite des Hauptgebäudes wird nun erstellt. Sie soll Ausschaltmöglichkeiten für alle Lichter in einer Etage sowie des ganzen Hauptgebäudes beinhalten und anzeigen, ob ein Fenster in der jeweiligen Etage geöffnet ist.

Statusseite des Hauptgebäudes:

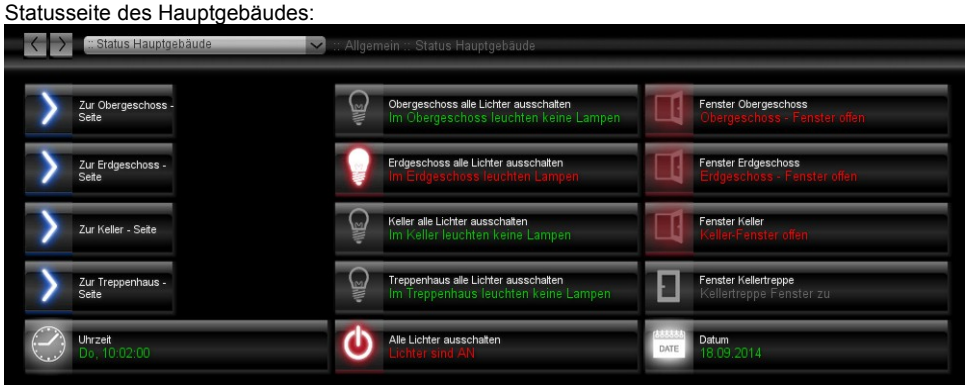

Abbildung 67: Statusanzeige mit Sprungzielen

Um wieder schneller in die Einzelnen Etagen zu gelangen, binden wir wieder plinks ein.Dort werden nun die einzelnen Räume aufgeführt und sind somit separat ansteuerbar. Sobald nun in einem Raum eine Lampe an ist, wird auf der Statusseite des Hauptgebäudes ein Lampensymbol leuchtend rot und der passende Text erscheint in der Anzeige. Betätigt man nun den Shifter, so werden alle Lampen der Etage ausgeschaltet. Will man dies bei mehreren oder allen Etagen mit nur einem Klick durchführen, so lässt sich das durch die Betätigung des Hauptschalters in der untersten Zeile, zwischen Uhrzeit- und Datumsanzeige, realisieren.

Die Seite wird nun wieder Schritt für Schritt aufgebaut und beginnt wie unsere Hauptseite mit der der Einbindung der Shifter und plinks. Darauf erhalten diese ihre Funktionalität und Designakzente.Um die Funktionalität jedoch zu gewährleisten, sollten Sie **davor** Ihre verschiedenen **Etagenseiten** mit allen Lichtern und Fenstern aufbauen. Dies führt zu einer besseren Übersicht der Gruppenadressen und erleichtert die Fehlersuche, falls bei den if-Anweisungen etwas schief läuft.

Beispiel Etagenseite:

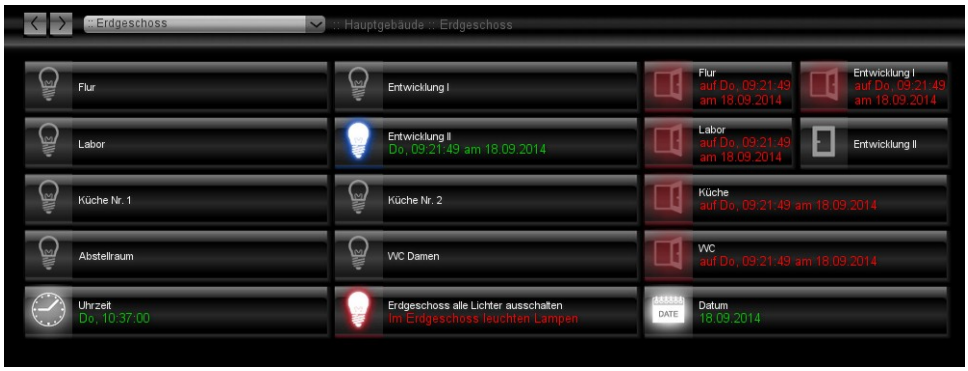

Abbildung 68: Detaillierte Anzeigen

Ist dies abgeschlossen können Sie sich wieder der Statusseite zuwenden.

Wie schon erwähnt erstellen wir zuerst unser Grundgerüst mit Button und Shifter. Die Etagenschalter haben wir als Global implementiert, da wir diese auf die Statusseite und Etagenseite einbinden. Das Beispiel behandelt nur den Hauptschalter des Erdgeschosses, die anderen können eins zu eins assoziiert werden.

```
[Webserver]
//Webelemente für Status Hauptgebäude
page(StatusID)[$Allgemein$,$Status Hauptgebäude$]
design $black$
header(0)
footer(0)
// Erste Zeile
plink(5)[RIGHT][ObergeschossID]$Zur Obergeschoss – Seite$
  none
  shifter(HauptschalterObergeschoss)[LIGHT]$Obergeschoss alle Lichter ausschalten$
  pshifter(11)[WINDOW]$Fenster Obergeschoss$
// Zweite Zeile
plink(6)[RIGHT][ErdgeschossID]$Zur Erdgeschoss-Seite$
  none
  shifter(HauptschalterErdgeschoss)[LIGHT]$Erdgeschoss alle Lichter ausschalten$
  pshifter(10)[WINDOW]$Fenster Erdgeschoss$
// Dritte Zeile
plink(7)[RIGHT][KellerID]$Zur Keller – Seite$
  none shifter(HauptschalterKellergeschoss)[LIGHT]$Keller alle Lichter ausschalten$ pshifter(9)[WINDOW]
$Fenster Keller$
// Vierte Zeile
plink(15)[RIGHT][TreppenhausID]$Zur Treppenhaus-Seite$
  none
  shifter(HauptschalterTreppenhaus)[SWITCH]$Treppenhaus alle Lichter ausschalten$
  pshifter(31)[WINDOW]$Fenster Kellertreppe$
//Fünfte Zeile
shifter(ClockID)[CLOCK] $Uhrzeit$
  pshifter(20)[SWITCH]$Alle Lichter ausschalten$
  shifter(DateID)[DATE]$Datum$
[EibPC]
//// Deklarationen
// Initialisierung der PageIDs für den Visalisierungsassistenten
AllgemeinWetterseiteID = 1
//Initialisierung der Webelement IDs
WINDID = 1
WolkenID = 2
ClockID = 3
DateID = 4
HauptschalterErdgeschoss = 5
```

Zur Funktion der Etagenschalter:

Prinzipiell besteht jeder Etagenschalter aus einer Variable die *wahr* oder *falsch* sein kann (Typ b01). Die Variable wird durch die Status-Gruppenadressen der Lampen geschrieben und ist wahr, sobald in einem Raum ein Licht an ist. Beispiel :

> *Wenn (Status_Raum1 == AUS oder Status_Raum2 == AUS oder Status_Raum 3 == AUS) dann soll die Variable Licht_Erdgeschoss auf EIN gesetzt werden, ansonsten auf EIN stehen.*

Im EibStudio wird dies dann für alle gewünschten Etagen wie folgt umgesetzt:

```
[EibPC]
//// Deklarationen
//Variablen
EGAlleLichter = 0b01

//// Funktionen
//Erdgeschoss Alle Lichter
if("Labor_1_Status-0/1/18" == AUS and "Büro_2_Status-0/1/19" == AUS and "Konstruktion_3_Status-0/1/20"
== AUS and "Konstruktion_4_Status-0/1/21" == AUS and "Flur_5_Status-0/1/22" == AUS and
"Küche_6_Status-0/1/23" == AUS and "Küche_7_Status-0/1/24" == AUS and "Abstellraum 8_Status-0/1/25"
== AUS and "WCDamen_9_Status-0/1/26" == AUS) then EGAlleLichter = AUS else EGAlleLichter = EIN
endif
```

Damit in unserem Webserver die Anzeige auch zu dem Wert der Variable passt, passen wir unseren Shifter über die Funktion webdisplay(ID, Text, Icon, State, TextStil,[Mbutton]) an.

```
[EibPC]
////Funktionen
/// Webdisplay Lichter globale Variablen
// EG
if EGAlleLichter == EIN then webdisplay (HauptschalterErdgeschoss,$Im Erdgeschoss leuchten
Lampen$,LIGHT,BRIGHTRED,BLINKRED) else webdisplay(HauptschalterErdgeschoss,$Im Erdgeschoss
leuchten keine Lampen$,LIGHT,INACTIVE,GREEN) endif
```

Damit die Ausführung des Shifters nun auch zum gewünschten Effekt führt, dem Ausschalten aller Lichter der Etage, muss unser Shifter noch mit einem fertigen Macro zugewiesen werden. Das Macro findet sich unter folgender Bezeichnung:

UmschaltShifterAUSGlobal

```
[Macros]
UmschaltShifterAUSGlobal(HauptschalterErdgeschoss,"EGalleLichter-
0/5/1",EGAlleLichter,LIGHT)
```

Wurde das Macro erfolgeich eingebunden, schalten sich nun alle Lichter der gewünschten Etage aus und die Icondarstellung des Shifters verändert sich von rot auf grau mit grüner Statusschrift.

Diese Prozedur führt man mit neuen Variablen ebenso für die Fensterkontakte aus. Der Unterschied zu den Lichtern besteht in der Ausführung als lokal und in der Funktionalität der Shifter. Wenn Sie keine Fenster haben, die über einen Stellmotor geschlossen werden können, entfällt die Einbindung eines Macros am Ende. Der Shifter dient also so nur als Anzeige.

```
[EibPC]
//// Deklarationen
//Variablen
EG_FensterOffen = 0b01
////Funktionen
//Erdgeschoss Fenster
if "Büro2+3Fensterkontakt-4/1/0" or"Konstruktion4+5Fensterkontakt-4/1/2" or "Küche7+8Fensterkontakt-4/1/4"
or "WCDamen10Fensterkontakt-4/1/6" or "Labor1Fensterkontakt-4/1/7" or "Flur6+9Fensterkontakt-4/1/8" then
EG_FensterOffen = 1b01 endif
/// pdisplay Fenster lokale Variablen
//EG
if EG_FensterOffen then pdisplay(10,$Erdgeschoss - Fenster
offen$,WINDOW,BRIGHTRED,BLINKRED,StatusID) else pdisplay(10,$Fenster
zu$,WINDOW,INACTIVE,GREY,StatusID) endif
```

Um alle Etagen bzw. das komplette Hauptgebäude schalten zu können, muss in der ETS eine Gruppenadresse angelegt werden, die alle gewünschten Lampen schaltet. Diese bindet man dann in das Macro *UmschaltShifterAUSLokal* ein. Nun kann man mit einem Klick das Gebäude schalten. Um das Icon anzupassen verwenden wir *pdisplay* und stellen es je nach Zustand dar.

```
[EibPC]
//// Deklarationen
//Variablen
AlleLichterHG = 0b01
////Funktionen
/// pdisplay Lichter lokale Variablen
//Hauptschalter
if AlleLichterHG == EIN  then pdisplay(20,$Lichter sind
AN$,SWITCH,BRIGHTRED,BLINKRED,StatusID) else pdisplay(20, $Lichter sind
AUS$,SWITCH,INACTIVE,GREEN,StatusID)endif
```

Damit auf unserer Hauptseite erkennbar ist, ob der Status von Lampen und Fenstern in Ordnung, beziehungsweise aus und geschlossen sind, verbinden wir die Variablen der Einzelnen Etagen (*EgAlleLichter,EG_FensterOffen,...*) mit einer neuen Status Variable *StatusHG*.

```
[EibPC]
//Variablen
StatusHG = 0b01
// if-Status
if ( Treppe == EIN or OGAlleLichter== EIN or EGAlleLichter == EIN or KGAlleLichter == EIN or
EG_FensterOffen == EIN or KG_FensterOffen ==EIN or OG_FensterOffen == EIN ) then StatusHG = EIN
else StatusHG = AUS endif


if StatusHG == EIN then pdisplay(5,$Status
kontrollieren$,CROSSCIRCLE,BRIGHTRED,BLINKRED,AllgemeinWetterseiteID) endif
if StatusHG == AUS then pdisplay(3,$Status OK$,OKCIRCLE,ACTIVE,GREEN,AllgemeinWetterseiteID) endif
```

Wird dies auch für das Nebengebäude erledigt, so erhalten wir bei jeweils einer leuchtenden Lampe der beiden Gebäude folgende Darstellung auf unserer Hauptseite:

Somit wäre unsere Zentrale Steuerseite fertig. Der Status unserer Lichter wird auf der Hauptseite angezeigt und wir haben die Möglichkeit mit nur einem Klick in auf die zentrale Schaltseite zu gelangen. Mit einem Blick erhalten wir Informationen über Außentemperatur, Bewölkung, 3-Tages Wettertrend, Windgeschwindigkeit, Datum, Uhrzeit sowie Sonnenauf- und Sonnenuntergang.

Was jetzt noch fehlt ist ein richtiger Blick in die Zukunft. Wir kreieren eine zusätzliche Wetterseite, die uns alles über die kommenden Stunden und Tage verraten wird.

```
[Webserver]
page(WetterseiteID)[$Allgemein$,$Wetterseite 2$]
design $black$
header(0)
footer(0)
```

Dazu benötigen wir eine große Datenbank von Wetterdaten.Um an diese zu gelangen ist ein Benutzerkonto auf Weather Wundergroud erforderlich.

Haben Sie dies erfolgreich abgeschlossen, müssen Sie sich unter dem Reiter Key Settings einen Key generieren. Hier genügt die kostenfreie Version für Developer. Jedoch müssen Sie berücksichtigen, dass die kostenfreie Version eine begrenzte Anzahl an Anfragen hat. Diese beträgt am Tag 500 und in der Minute maximal 10 Anfragen.

Haben Sie ihren Key generiert, so müssen Sie ihn in unser Macro Wettervorhersage einbauen.

```
[Macros]
Wettervorhersage(Key,Wetterstation,Germany,after(systemstart(),1300u64) or cycle(10,00))
```

Für die Wetterstation müssen Sie ihre Stadt eintragen, oder suchen Sie nach einer passenden Station für Ihre Gegend.

Sind die Daten alle eingegeben, sehen sie nach Systemstart in der Variablenabfrage (F5) eine Liste von neuen Variablen, welche Weather Underground zur Verfügung stellt. Überprüfen Sie ob Ihre angegebene Wetterstation auch die Daten liefert, welche Sie implementieren möchten. Wenn alles passt, lassen sich die Daten nun mit Hilfe eines Button oder Shifter anzeigen.

Beispiel:

```
[Webserver]
page(WetterseiteID)[$Allgemein$,$Wetterseite 2$]
design $black$
header(0)
footer(0)


button(WINDID)[WIND]$Wind in km/h$ pbutton(14)[WEATHER]$Wetter$ pbutton(25)[WEATHER]
$Regenwahrscheinlichkeit$ pbutton(24)[RAIN]$Feuchtigkeit$  pbutton(12)[TEMPERATURE]$Aussentemp.:
Min$ pbutton(13)[TEMPERATURE]$Aussentemp.:Max$


[EibPC]
///Wounderworld Wetter
//Wettervorhersage Heute
if after(systemstart(),3000u64) or  change(Wettervorhersage_Heute_Wind_Max_Richtung) then
pdisplay(11,$aus$ + convert(Wettervorhersage_Heute_Wind_Max_Richtung,$
$),WIND,ACTIVE,BLINKBLUE,WetterseiteID) endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Temperatur_Min) then
pdisplay(12,convert(Wettervorhersage_Heute_Temperatur_Min,$$)+$
°C$,TEMPERATURE,ACTIVE,BLINKBLUE,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Temperatur_Max) then
pdisplay(13,convert(Wettervorhersage_Heute_Temperatur_Max,$$)+$
°C$,TEMPERATURE,BRIGHTRED,BLINKRED,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Wetter) then
pdisplay(14,convert(Wettervorhersage_Heute_Wetter,$$),WEATHER,STATE4,GREY,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Regenwahrscheinlichkeit) then
pdisplay(25,convert(Wettervorhersage_Heute_Regenwahrscheinlichkeit,$
$),WEATHER,STATE6,BLINKBLUE,WetterseiteID)endif


if after(systemstart(),3000u64) or change(Wettervorhersage_Heute_Feuchtigkeit) then
pdisplay(24,convert(Wettervorhersage_Heute_Feuchtigkeit,$
$),RAIN,ACTIVE,BLINKBLUE,WetterseiteID)endif
```

Mit der picture Funktion, fügen wir noch eine Windanzeige ein, welche die Windrichtungen in Deutschland anzeigt.

```
[Webserver]
picture(9)[DOUBLE,ZOOMGRAF]($Windrichtungen$,$http://wwwdyn.zdf.de/ext/weather/wind-brd-0.jpg$)
```

Daneben platzieren wir mit der Funktion timechart einen Temperaturverlauf der letzten 24 Stunden.

```
[Webserver]
mtimechart(8)[QUAD,NOAUTOSCALE,48,-10,45]( $Aussentemperaturverlauf$,LEFT, ChartBuffer1)
```

Damit der timechart weiß, welche Werte er darstellen soll, müssen wir folgende Konfigurationen vornehmen:

timebufferconfig(ChartBufferID, MemTyp, Laenge, DataTyp)

MemTyp gibt an, ob der Speicher im Ring (0) oder linear (1) beschrieben wird. Die Länge der max. Aufzeichnung der Zeitreihe wird mit Laenge (0u16 bis 65565u16) angegeben. DataTyp stellt eine repräsentative Zahl der Zeitreihe dar, z.B. 0f16 für 16-Bitzahlen oder 3% für u08 Werte.

Nun müssen die Zeitreihen noch mit Daten „befüllt“ werden. Die Funktion

timebufferadd(ChartBufferID, Daten)

erledigt diese Aufgabe.

Nachdem die Zeitreihe über einige Zeit im Enertex® EibPC aufgenommen wurde, muss sichergestellt werden, dass diese auch beim Neueinspielen des Programms oder Neustart die Werte nicht verloren gehen. Die Funktionen

timebufferstore(ChartBufferID)

timebufferread(ChartBufferID)

sind für diese Aufgabe geschaffen. timebufferstore legt die Werte des Timebuffers mit der ChartBufferID permanent in den Flashspeicherdes Enertex® EibPC ab, timebufferread liest einen abgespeicherten Puffer zurück.

```
button(ELEVATION)[WEATHER]$Höhenwinkel in °$ button(Sonnenstand)[UP]$Sonnenstand$
button(AZIMUTH)[WEATHER]$AZIMUTH$ pbutton(23)[CLOCK]$Sonnenhöchstand:$
[EibPC]
//mtimechart
Len=35040u16
Datatyp=3.3f16
MemTyp=0
timebufferconfig(ChartBuffer1, MemTyp, Len,"Außentemperatur-14/6/3" )

if mtime(0,0) or mtime(15,0) or mtime(30,0) or mtime(45,0) then
{timebufferadd(ChartBuffer1,"Außentemperatur-14/6/3")
} endif

if chtime(01,00,00) then {
timebufferstore(ChartBuffer1)
} endif

if systemstart() then {
timebufferread(ChartBuffer1)
} endif
```

Somit wäre der Temperaturverlauf korrekt angezeigt.Damit wir den timechart noch besser ausnutzen, fügen wir noch den Sonnenstandsverlauf hinzu. Dafür benötigen wir den Azimutwinkel und den Höhenwinkel der Sonne. Diese beiden Daten bekommen wir durch die Funktionen

azimuth()

Höhenwinkel : elevation()

Der Enertex® EibPC muss die geographische Länge und Breite des betreffenden Ortes kennen. Diese können im Enertex® EibStudio eingegeben werden(s.Seite.156). Um an Ihre Koordinaten und weitere Informationen zu gelangen, könnte Ihnen diese Seite hier weiterhelfen.

Den timechart und die Konfiguration müssen Sie nun wie folgt erweitern:

```
[Webserver]
mtimechart(8)[QUAD,NOAUTOSCALE,48,-10,45,-10,65]( $Aussentemperaturverlauf$,LEFT,
ChartBuffer1,$Sonnenstand$,RIGHT,ChartBuffer2
[EibPC]
Len=35040u16
Datatyp=3.3f16
MemTyp=0
timebufferconfig(ChartBuffer1, MemTyp, Len,"Außentemperatur-14/6/3" )
timebufferconfig(ChartBuffer2, MemTyp, Len,elevation() )
if mtime(0,0) or mtime(15,0) or mtime(30,0) or mtime(45,0) then
{timebufferadd(ChartBuffer1,"Außentemperatur-14/6/3");
timebufferadd(ChartBuffer2,elevation() )
} endif
```

Die beiden Winkelanzeigen, sowie einen Statusbutton des Sonnenverlaufes, der ausgibt, ob der Sonnenstand steigt oder fällt, können Sie nun auch einfügen. Dafür benötigen Sie den Azimutwinkel.

Der Sonnenhöchststand ist immer bei Azimutwinkel = 180 Grad, davor steigt der Sonnenstand. Um zu wissen wann der Sonnenhöchststand war, lassen wir uns bei Azimutwinkel 180 ° die Uhrzeit ausgeben. Somit ergibt sich folgender Code.

```
[Webserver]
button(ELEVATION)[WEATHER]$Höhenwinkel in °$ button(Sonnenstand)[UP]$Sonnenstand$
button(AZIMUTH)[WEATHER]$AZIMUTH$ pbutton(23)[CLOCK]$Sonnenhöchstand:$
[EibPC]
if change(azimuth()) then webdisplay(AZIMUTH,(convert(convert(azimuth(), 0.0 ), $$c14 ) + $ Grad $c14),
WIND, ACTIVE, GREEN) endif
if change(elevation()) then webdisplay(ELEVATION,(convert(convert(elevation(), 0.0 ), $$c14) + $ Grad$c14),
WIND, ACTIVE, GREEN) endif


if (change(elevation()) and azimuth()<175f32 and elevation()>0f32 ) then webdisplay(Sonnenstand, $steigt$,
UP, BRIGHTRED, BLINKRED) endif


if (change(elevation()) and azimuth()>175f32 and elevation()<185f32 )then webdisplay(Sonnenstand,$Sonne
im Zenit$, WEATHER, BRIGHTRED, BLINKRED) endif


if(change(elevation()) and azimuth()>185f32 and elevation()>0f32 )then webdisplay(Sonnenstand,$fällt$ ,
DOWN, ACTIVE, BLINKBLUE) endif


if (change(elevation()) and azimuth()>185f32 and elevation()<0f32 )then webdisplay(Sonnenstand,$Nacht$ ,
NIGHT, DISPLAY, GREY) endif


if azimuth()>=185f32  then pdisplay(23,settime(),CLOCK,ACTIVE,GREEN,WetterseiteID) endif
```

Als nächstes fügen wir ein Regenradar als picture ein. Wir entschieden uns für diese Version, welche für alle Bundesländer verfügbar ist. Als erstes muss man das Radar als Grafik anzeigen lassen, womit man diese Ansicht erhält. Alle fünf Minuten wird ein neues Bild aufgenommen, welches wir als eine Variable abspeichern und dann im Webserver angezeigen. Insgesamt benötigen wir also 12 Variablen. Die Konfiguration schaut somit wie folgt aus:

```
[Webserver]
 picture(4)[DOUBLE,CENTERGRAF]($RegenRadar$,$www.google.de$)
[EibPC]
URL1 = $$
URL2 = $$
URL3 = $$
URL4 = $$
URL5 = $$
URL6 = $$
URL7 = $$
URL8 = $$
URL9 = $$
URL10 = $$
URL11 = $$
URL12 = $$


if mtime(01,05) then picture(4,$Regenradar$,WetterseiteID,URL1) endif
if mtime(06,05) then picture(4,$Regenradar$,WetterseiteID,URL2) endif
if mtime(11,05) then picture(4,$Regenradar$,WetterseiteID,URL3) endif
if mtime(16,05) then picture(4,$Regenradar$,WetterseiteID,URL4) endif
if mtime(21,05) then picture(4,$Regenradar$,WetterseiteID,URL5) endif
if mtime(26,05) then picture(4,$Regenradar$,WetterseiteID,URL6) endif
if mtime(31,05) then picture(4,$Regenradar$,WetterseiteID,URL7) endif
```

```
if mtime(36,05) then picture(4,$Regenradar$,WetterseiteID,URL8) endif
if mtime(41,05) then picture(4,$Regenradar$,WetterseiteID,URL9) endif
if mtime(46,05) then picture(4,$Regenradar$,WetterseiteID,URL10) endif
if mtime(51,05) then picture(4,$Regenradar$,WetterseiteID,URL11) endif
if mtime(56,05) then picture(4,$Regenradar$,WetterseiteID,URL12) endif
```

Mit diesem Code erneuert sich alle fünf Minuten das Radarbild, allerdings fehlt noch die Belegung der einzelnen URL´s. Dazu benötigen wir die URL der Grafik, welche um 09.45 wie folgt zurückgegeben wird:

Die Fett markierten Zahlen sind für das Datum zuständig, kursiv und unterstrichen gibt die Uhrzeit an.

http://www.wetteronline.de/?pid=p_radar_map&ireq=true&src=radar/vermarktung/p_radar_map/wom/**2014/09/30**/Intensity/BAY/grey_flat/**20140930***0745*_BAY_Intensity.gif#1412074528673

Es fällt auf, dass unsere URL 2 Stunden voraus ist. So müssen wir also von unserer Systemzeit 2 Stunden abziehen um die richtige anzeige zu bekommen.

```
if cycle (01,00) then Weatherhour=convert(hour()-2,$$) endif
```

Ein weiteres Problem ist die 10-Uhr Grenze. Wir benötigen als Stundenrückgabe immer folgenden Datentyp: 08; 09; 10; 11;....

Da wir bei convert(hour()**-2**,$$) vor 10 Uhr jedoch immer nur eine einzelne Zahl bekommen,müssen wir eine Unterscheidung machen, ob es vor oder nach 10 Uhr in der URL ist.

```
if chtime(11,59,59) then NachZehn=1 else NachZehn=0 endif
```

Für den Fall NachZehn=1 und somit wahr folgt folgender veränderlicher Code:

```
if NachZehn == 1 and mtime(01,00) then URL1=$http://www.wetteronline.de/?pid=p_radar_map&ireq=true&src=radar/vermarktung/p_radar_map/wom/$ + convert(Jahr,$$) + $/$ + convert(Monat,$$) + $/$ + convert(Tag,$$) + $/Intensity/BAY/grey_flat/$ + convert(Jahr,$$) + convert(Monat,$$) + convert(Tag,$$) + convert(Weatherhour,$$)+$00$+$_BAY_Intensity.gif$ endif
```

convert(Weatherhour,$$)+00 ergibt nun also in einer realen Zeit von 12:00 Uhr in der URL 10:00 Uhr

Wenn es noch nicht nach 10 Uhr ist, wird eine 0 vor ***convert(Weatherhour,$$)+00*** benötigt, um wieder das richtige Format zu erhalten.

```
if NachZehn == 0 and mtime(36,00) then URL8=$http://www.wetteronline.de/?pid=p_radar_map&ireq=true&src=radar/vermarktung/p_radar_map/wom/$ + convert(Jahr,$$) + $/$ + convert(Monat,$$) + $/$ + convert(Tag,$$) + $/Intensity/BAY/grey_flat/$  + convert(Jahr,$$) + convert(Monat,$$) + convert(Tag,$$) + $0$ + convert(Weatherhour,$$)+$35$ +$_BAY_Intensity.gif$ endif
```

Hier ist also eine URL mit folgender Uhrzeit dargestellt: 0X:35 Uhr. Die Anzeige erfolgt mit mtime(36,00) jedoch erst eine Minute später, da es teilweise zu Verzögerungen der Seite kommen kann und dann keine Grafik mehr geladen wird.

Damit wir jetzt jedoch noch einen richtigen Wetterfilm darstellen können, bauen wir noch einen Schalter ein, der bei Betätigung den Film abspielen soll.

```
page(WetterseiteID)[$Allgemein$,$Wetterseite$]
design $black$
header(0)
footer(0)
pshifter(10)[PLAY]$Wetterfilm abspielen$
```