Eigener Wetterserver für aktuelle Wetterdaten und Vorhersagen in Loxone

1. Einleitung

Loxone bietet mit dem Cloudservice „Weather“ einen professionellen Wetterservice an, der sich nahtlos in die Loxone Konfiguration integriert. Leider ist dieser Service jedoch noch nicht weltweit verfügbar.

Mit der hier vorgestellten Lösung kann man sich seinen eigenen Wetterservice aufbauen, der Wetterdaten von allen(?) weltweit verfügbaren Wetterstationen verwenden kann. Hierbei werden folgende Features unterstützt:

Features:

- Aktuelle Wetterdaten
- Tagesgenaue Wettervorhersage für die nächsten 4 Tage (Heute + 3 Tage im voraus)
- Stundengenaue Wettervorhersage für die nächsten 36 Stunden
- Wetterdaten von Wunderground (ehemals University of Michigan, jetzt Mitglied der The Weather Channel Companies). Beinhaltet z. B. die Wetterstationen des DWD.
- Vollständig kostenlos
- Wetteranzeige und Vorhersagen für "menschliche Benutzer" über den Webpage-Baustein

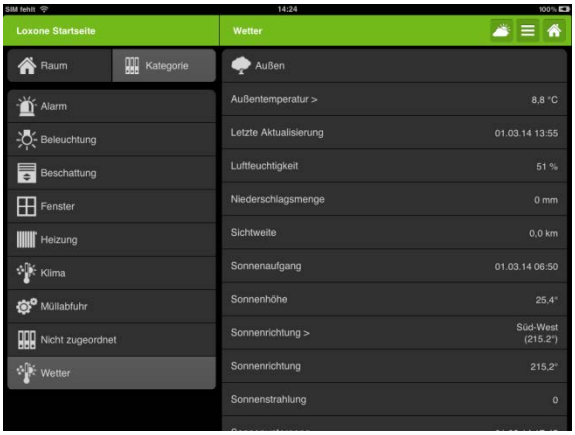
Voraussetzungen:

- Eigener Webspace, auf dem man CGI-Skripte (Perl) ausführen kann
- Alternativ: Eigener Linux- oder Windowsserver mit Webserver und Perl (z. B. RasberryPi oder ein NAS)
- Installiertes DateTime-Modul für Perl
- Kostenloser Wunderground-Account zum Zugriff auf die Wunderground-Wetter-API
- Etwas Erfahrung beim Installieren von CGI-Anwendungen (Perl)

2. Welche Daten kann man wie nutzen?

Wunderground bietet zahlreiche aktuelle Daten sowie auch umfangreiche und detaillierte Vorhersagedaten an. Getestet habe ich mit der Wetterstation des DWD in Dresden-Klotzsche. Alle Daten können als Variable in LoxoneConfig weiterverarbeitet werden (z. B. für Berechnungen oder gezielte Steuerungen z. B. einer Gartenbewässerungsanlage, Beschattung, o. ä.).

Des weiteren können alle Daten zur Visualisierung angezeigt werden. Eine eigene übersichtliche Wettervorhersage-Seite "für den menschlichen Benutzer" innerhalb der Visualisierung ist ebenfalls integriert. Diese Seite wird über den Webpage-Baustein eingebunden.



Alle Daten, die genutzt werden können inkl. Einer kurzen Erläuterung und dem Variablennamen, den ihr in LoxoneConfig verwenden müsst, findet ihr im Anhang an diese Anleitung!

3. Server-Installation

Vorbereitungen

Als erstes müsst ihr, falls noch nicht geschehen, dass DateTime- und das JSON-Modul für Perl nachinstallieren. Auf Debian-basierten Systemen (z. B. RaspberryPi) geschieht das mit dem folgenden Befehl:

```
sudo apt-get install libdatetime-perl libjson-perl
```

Auf anderen Linuxsystemen kann der Befehl abweichen (Doku/Google hilft weiter). Auf den meisten Webservern sollten die Module bereits installiert sein.

Die CGI-Skripte müssen nun vor der ersten Verwendung noch angepasst werden. Hierzu kann man einen normalen Texteditor verwenden. Achtung! Der mit Windows mitgelieferte Text-editor „Notepad“ und auch z. B. „Wordpad“ oder „Microsoft Word“ sind **nicht** geeignet. Bei den Dateien handelt es sich um reine ASCII Dateien, die zudem im Unixdateiformat (also mit Unix-Zeilenumbrüchen) abgespeichert sind. Die oben genannten Editoren zerstören die Dateien, sodass sie vom Perl-Interpreter nicht mehr gelesen werden können.

Ich empfehle daher für Windowsnutzer das kostenlose Programm „Notepad++“, welches auch als portable Version (also ohne notwendige Installation) verfügbar ist: http://notepad-plus-plus.org/download

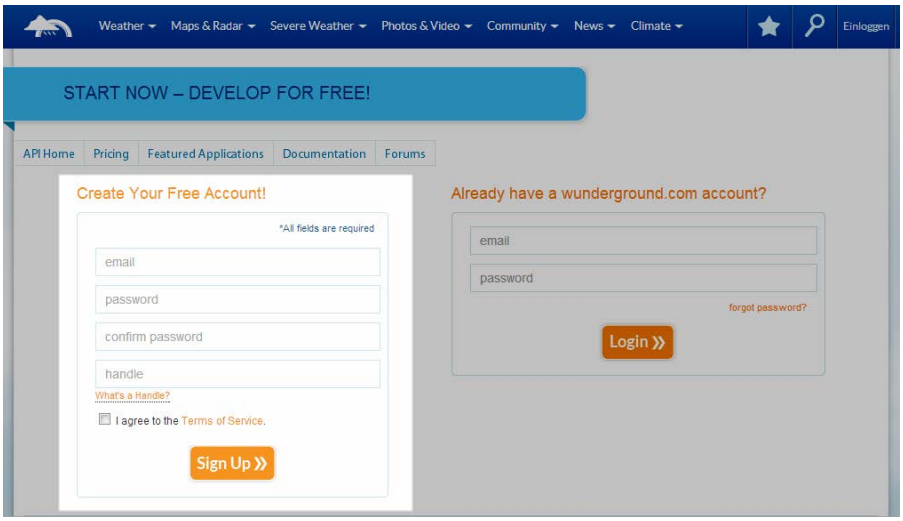
Linux/Unix-Nutzer können jeden x-beliebigen Editor verwenden – sie beherrschen alle das Editieren von ASCII-Dateien.

Bevor man mit der Bearbeitung der Dateien startet, benötigt man noch einige Daten der eigenen CGI-Umgebung auf dem Webspace. Hierzu schaut man am Besten in die FAQ oder auf die Support-Seiten des eigenen Providers. Dort sollten alle Daten verfügbar sein. Ansonsten kann man auch versuchen mit den hier aufgelisteten Angaben zu arbeiten (in 98% aller Fälle sollten diese funktionieren).

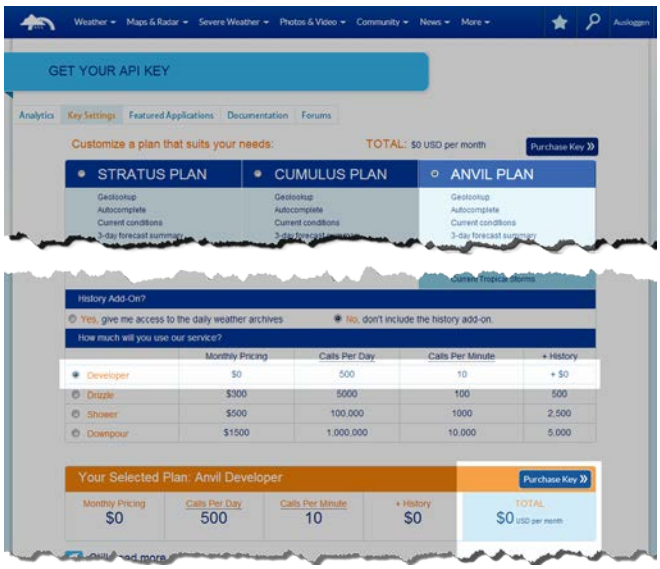
- Vollständiger Pfad des Perl-Interpreters: z. B. **`/usr/bin/perl`**
- Verzeichnis für CGI-Skripte auf dem Webspace: z. B. **`/cgi-bin/`**
- Verzeichnis für HTML-Dateien auf dem Webspace: z. B. **`/www/`**

Als nächstes benötigt man noch einen kostenlosen Account beim Wetterservice Wunderground. Innerhalb seines Accounts kann man dann einen sogenannten API-Schlüssel generieren (API-Key). Dieser Schlüssel berechtigt dann zum Zugriff auf die Wunderground-API und damit auf die Wunderground-Wetterdatenbank.

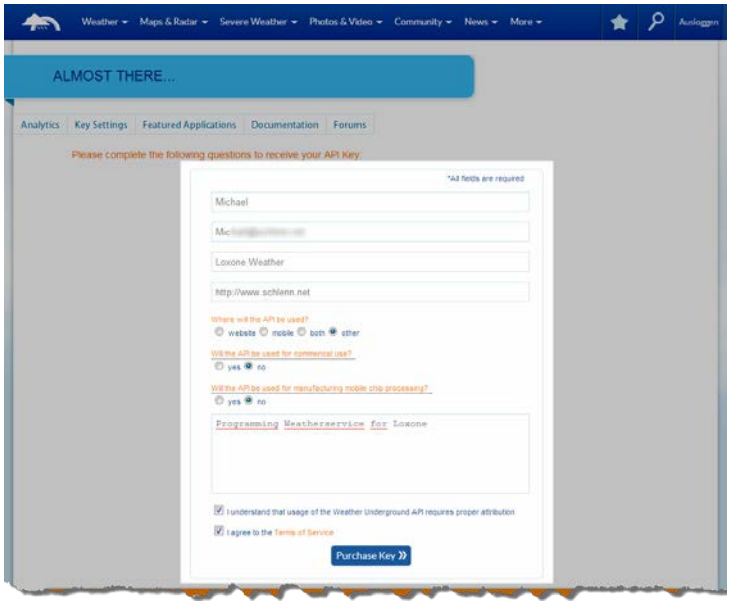
Dazu ruft Ihr die API-Webseite unter http://www.wunderground.com/weather/api/ auf und klickt auf „Sign up for free“. Unter „Create Your Free Account“ gebt Ihr dann Eure Emailadresse ein und vergebt ein Passwort. Fertig.



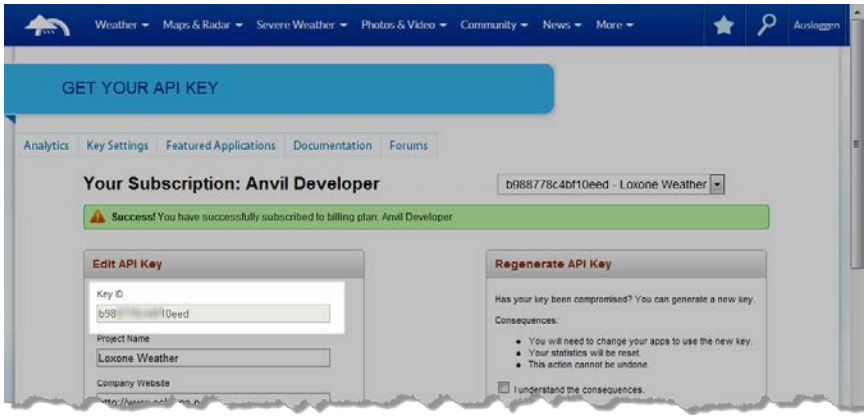
Anschließend erhaltet Ihr eine Email, in der Ihr noch auf den entsprechenden Bestätigungs-Link klicken müsst, um Euren Account zu aktivieren.
Jetzt loggt Ihr euch auf der gleichen Internetseite mit Eurem neuen Account ein und wechselt auf den Karteireiter „**Key Settings**“. Um einen neuen kostenlosen API-Key zu generieren wählt Ihr oben auf der Seite den „**ANVIL PLAN**“ aus (der die umfangreichsten Daten enthält) und unten unter „How much will you use our service?“ wählt Ihr „**Developer**“ aus. Der Developerschlüssel ist kostenlos und erlaubt 500 API-Zugriffe pro Tag und maximal 10 Zugriffe pro Minute. Für uns vollkommen ausreichend, da unser Server später die Daten selbst speichert und damit vom Miniserver so oft wie notwendig auf die Daten zugegriffen werden kann. Mit einem Klick auf „Purchase Key“ kauft Ihr Euren kostenlosen API-Key.



Auf der nächsten Seite müsst Ihr nun noch einige Angaben zu Eurem Projekt machen. Die Angaben sind beliebig und können frei gewählt werden:



Anschließend wird Euch Euer neuer API-Key angezeigt. Von hier aus kann man ihn dann bequem per „Copy & Paste“ kopieren. Man benötigt ihn später bei der Installation des Servers (siehe unten).



Es ist im Übrigen auch möglich sich mehrere API-Keys zu erzeugen. Auch alle weiteren „Developer“-Keys sind dabei kostenlos.

Installation der Skripte und der Datenbanken

Nach diesen Vorbereitungen geht es an die eigentliche Installation des Servers. Als erstes besorgt ihr Euch das Programmpaket „Loxone Weather“ von meiner Homepage: http://www.schlenn.net/download/loxone/loxoneweather

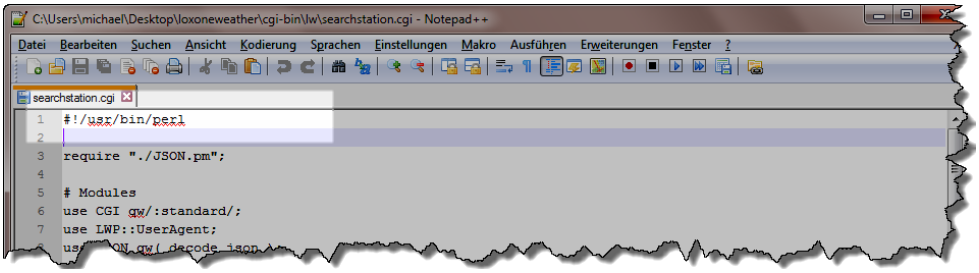
Jetzt entpackt ihr das Paket mit einem ZIP-Programm in ein beliebiges Verzeichnis auf Eurer Festplatte. Nach dem Entpacken solltet Ihr 2 neue Verzeichnisse haben:

- `cgi-bin`: Hier sind alle CGI-Skripte für den Server gespeichert
- `www`: Hier sind alle HTML- und Bilddateien für den Server gespeichert, die für die Wetterseite für den Webpage-Baustein notwendig sind

Wechselt in das Unterverzeichnis „cgi-bin“. Dort findet Ihr insgesamt 3 Dateien mit der Dateiendung „.cgi“ bzw. ".pl":

- `get.cgi`
- `fetch.cgi`
- `show.cgi`

Bei diesen Dateien handelt es sich um die CGI- bzw. Perl-Skripte. Öffnet alle 3 Dateien nacheinander mit dem Texteditor. Die allererste Zeile jedes Skripts muss nun angepasst werden. Die Zeile muss aus einem vorangestellten `#!` (Raute und Ausrufezeichen) gefolgt von dem vollständigen Pfad zu Eurem Perlinterpreter (siehe oben) bestehen. Zwischen dem Ausrufezeichen und der Pfadangabe darf sich kein Leerzeichen befinden! Also korrekt und vollständig z. B.: **`#!/usr/bin/perl`**



Nachdem Ihr die Pfadangabe angepasst habt, speichert Ihr jedes Skript ab.

Jetzt wechselt ihr noch in das Unterverzeichnis `bin`. Hier findet ihr ebenfalls 3 Perl-Dateien:

- `cachedver.pl`
- `fetch.pl`
- `send.pl`

Mit diesen beiden Dateien macht ihr genau das gleiche (also erste Zeile anpassen).

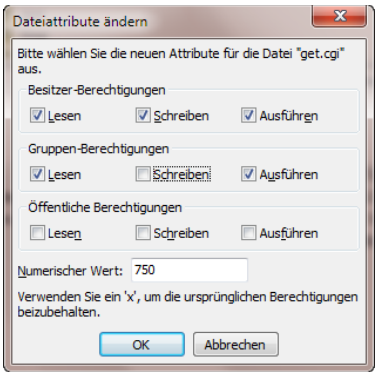
Anschließend muss der komplette Inhalt des `cgi-bin`-Verzeichnisses von Eurer Festplatte auf Euren Webspace hochgeladen werden. Wichtig ist, dass Ihr den Inhalt in das für CGI-Skripte vorgesehene Verzeichnis hochladet (siehe oben). Wenn Ihr wollt könnt Ihr der Übersicht halber natürlich auch ein Unterverzeichnis im `cgi-bin`-Verzeichnis anlegen (z. B. „`loxoneweather`“ o.ä.). Zum Hochladen könnt Ihr jedes beliebige FTP-Programm verwenden. Wichtig ist, dass das Programm die CGI-Dateien im ASCII-Modus überträgt. Das machen aber die meisten FTP-

Programme automatisch. Ein gutes und kostenloses FTP-Programm ist Filezilla: http://www.filezilla-project.org/

Zu guter Letzt müssen jetzt noch die korrekten Dateirechte für die einzelnen Dateien und Verzeichnisse vergeben werden. Dazu klickt man bei den meisten Programmen mit der rechten Maustaste auf die entsprechenden Dateien und wählt dann den Menüpunkt „Dateirechte“ o. ä. aus.

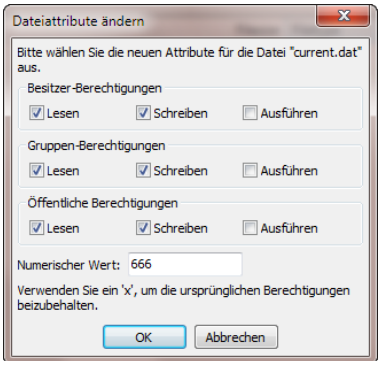
- Als erstes setzt ihr die Rechte für die 7 Skripte auf „750“:
  - `get.cgi`
  - `fetch.cgi`
  - `show.cgi`
  - `bin/cachedver.pl`
  - `bin/send.pl`
  - `bin/fetch.pl`
  - `bin/cron.sh`

  Die Rechte werden gemäß der folgenden Abbildung gesetzt:



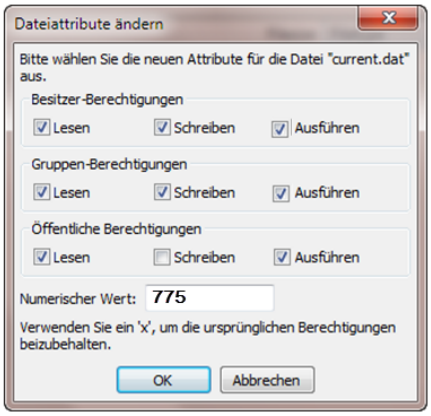
- Jetzt setzt Ihr im Unterverzeichnis „`data`“ noch die Berechtigungen für die Datenbankdateien auf „666“:
  - `current.dat`
  - `dailyforecast.dat`
  - `hourlyforecast.dat`

  Die Rechte dieser Dateien werden gemäß der folgenden Abbildung gesetzt



- Zuletzt werden noch die Rechte des Datenbank-Verzeichnisses "data" selbst angepasst. Die Berechtigung muss hier auf "775"

  Die Rechte dieser Dateien werden gemäß der folgenden Abbildung gesetzt:



- **Solltet Ihr die Skripte auf einem RaspberryPi installieren** und habt alle Dateien als Root auf den Server gespielt, solltet Ihr jetzt alle Dateien dem Benutzer des Webservers ("www-data") zuordnen, damit der Webserver auch die Berechtigung hat die Dateien zu verändern. Dazu gebt Ihr folgenden Befehl ein (Pfadangabe muss eventuell angepasst werden):

  ```
  chown -R www-data.www-data /usr/lib/cgi-bin/*
  ```

Damit sind alle notwendigen Schritte für die Installation abgeschlossen. Die CGI-Skripte sollten nun funktionieren. Zum Test ruft Ihr nun die folgende Internetseite auf:

http://www.euereinternetadresse.de/cgi-bin/get.cgi

Wenn alles korrekt eingerichtet wurde, dann solltet ihr einige Zeilen mit Variablen und Text zu Gesicht bekommen.

Gibt es irgendein Problem mit den Skripten und kann der Server die Skripte damit nicht ausführen, erscheint meist ein „Internal Server Error“ im Webbrowser. In diesem Fall ist irgendetwas bei der Einrichtung schief gegangen. In so einem Fall ist es gut, wenn der Provider den Zugriff auf die Error-Logdateien des Servers zulässt, damit man gezielt nach dem Fehler suchen kann. Hilft alles nichts weiter, so beginnt man am besten einfach noch einmal von vorne und geht die Anleitung nochmal Schritt für Schritt durch.

Die häufigsten Fehler wenn die Skripte nicht funktionieren sind:

- Pfad zum Perlinterpreter nicht korrekt gesetzt
- Beim Speichern der Skripte diese nicht im Unixformat abgespeichert (Editor)
- Beim Übertragen auf den FTP-Server nicht den Übertragungsmodus „ASCII“ verwendet
- Dateiberechtigungen nicht korrekt gesetzt

Einrichtung / Setup

Nachdem die Skripte korrekt installiert wurden gilt es jetzt noch einige Optionen korrekt zu setzen und vor allem den Skripten noch bekannt zu geben, für welche Wetterstation und mit welchem API-Key die Wetterdaten vom Wunderground-Server geholt werden sollen.

Wir suchen uns also als erstes unsere Wetterstation aus, von der wir Daten verarbeiten wollen. Dabei hat jede Wetterstation eine eindeutige Nummer/Kennzeichnung, die wir benötigen.

Wunderground unterhält auch ein Netz von privaten Wetterstationen (PWM: Personal Weather Stations). Am Besten sucht man über die sogenannte Wundermap eine Station in seiner Nähe.

Man benötigt die Station ID der Station. Die PWS-Stationen haben dann eine Bezeichnung wie "ISACHSEN118" o. ä.

Eine weitere Alternative ist sich einfach (z. B. aus LoxConfig) die Koordinaten des eigenen Standorts herauszusuchen (Längen- und Breitengrad). Auch diese kann man später als "Stationsnamen" verwenden und der Wundergroundserver sucht dann automatisch die nächstgelegene Wetterstation heraus.

Oder man verwendet "autoip" als Stationsnamen. Hierbei sucht der Wundergroundserver anhand der eigenen IP-Adresse die nächstgelegene Wetterstation.

Noch eine Variante ist einfach den Airport-Code des eigenen Flughafens zu verwenden (also z. B. "DRS" für Dresden).

Nun wechselt Ihr in das Verzeichnis „**`cgi-bin/data`**“ im Ordner auf der eigenen Festplatte, wo Ihr das Installationspaket entpackt hat, und öffnet wieder mit einem Texteditor die Datei „**`settings.dat`**“.

In dieser Datei werden einige Optionen gesetzt, die später von den CGI-Skripten verwendet werden. Dabei sind Zeilen mit einer vorangestellten Raute (#) Kommentare und werden ignoriert. Jede Option wird als Variable in der Form `our $variable = “Wert“;` gesetzt. Wichtig hierbei ist, dass der Wert jeweils in Anführungsstriche gesetzt werden muss und jede Zeile mit einem Semikolon abschließt.

Die Datei sollte selbsterklärend sein. Geht die ganze Datei durch und setzt die Optionen entsprechend eurer Installationsumgebung oder euren Wünschen.

Habt Ihr alle Angaben erledigt speichert Ihr die Datei ab und übertragt sie wie die CGI-Skripte weiter oben auch auf euren Webspace ins Verzeichnis „`cgi-bin/data`“. Die dort bereits existierende Datei muss überschrieben werden.

Testen

Jetzt ist der Server einsatzbereit und es kann losgehen! Der eigentliche Server besteht lediglich aus 3 CGI- bzw. Perl-Skripten:

- **`fetch.cgi`** und **`bin/fetch.pl`**: Dieses Skript holt die aktuellen Wetterdaten sowie die Wettervorhersage vom Wunderground-Server ab und speichert sie in der eigenen Datenbank ab.
- **`get.cgi`**: Mit diesem Skript kann man sich alle Werte anzeigen lassen. Die Datei dient dazu per Virtuellem HTML Eingang die Werte später in LoxoneConfig zu verwenden.
- **`show.cgi`**: Zeigt eine Wetterseite an, die per Webpage-Baustein in LoxoneConfig eingebunden werden kann (mehr dazu später)

Zum Testen kann man die Skripte direkt über den eigenen Webbrowser aufrufen. Später erfolgen die Aufrufe und die Auswertung dann direkt aus LoxoneConfig heraus.

Als erstes holt man sich den ersten Datensatz vom Wunderground-Server ab. Ruft dazu das Skript **`fetch.cgi`** direkt im Webbrowser auf. Als Bestätigung erhält man, wenn alles geklappt hat, ein simples „fetch@1“ im Browserfenster. Das Skript braucht etwas zum abarbeiten der ganzen Dinge. Wartet ca. 15-30 Sekunden ab, dann sollten aktuelle Wetterdaten in den Datenbanken im Verzeichnis **`data`** gespeichert sein.

Zur Kontrolle kann man sich nach dem Aufruf des Skriptes mit dem FTP-Programm noch die Datei `/cgi-bin/data/current.dat` herunterladen und in einem Texteditor betrachten. Die

Datei enthält nach dem Abruf vom Wundergroundserver die aktuellen Wetterdaten und sollte eine Zeile mit lauter Pipes („|“) enthalten.

Als nächstes versucht man die Wetter-Werte abzurufen. Wie schon erwähnt dient dazu das Skript **`get.cgi`**. Einfach im Browser aufrufen und es sollten euch alle aktuellen Wetterdaten sowie einige Vorhersagewerte angezeigt werden.

Ist ein Wert nicht verfügbar, gibt das Skript den Wert „-9999.00“ zurück.

Mehr zu den verfügbaren Werten findet Ihr später im Kapitel zur Programmierung unter Loxone.

Automatisch Wetterdaten holen

Der Server muss die Wetterdaten regelmäßig vom Wunderground-Server abholen. Hierzu dient das Skript „`fetch.cgi`“ bzw. „`/bin/fetch.pl`“ auf dem Server. Es gibt zwei Möglichkeiten das Holen der Daten anzutriggern:

PER CRON-JOB (Variante für einen Server „zu Hause“):

Hat man die Möglichkeit auf dem eigenen Webserver einen CRON-Job einzurichten (der „Taskmanager“ auf Linuxsystemen), so kann man das Fetch-Skript sehr einfach z. B. von einem regelmäßigen Cronjob aufrufen. Leider bieten nicht alle Provider die Möglichkeit an, eigene CRON-Jobs einzurichten. Wenn das der Fall ist, bleibt nur die zweite Möglichkeit übrig, das Fetch-Skript vom Miniserver aus anzutriggern (siehe nächster Abschnitt).

Als erstes sind wieder einige Anpassungen an einer Datei notwendig. Ruft in eurem Texteditor die Datei `bin/cron.sh` auf und passt die dortigen Pfadangaben an euer System an. Anschließend ladet ihr es wieder auf euren Webspace hoch.

Wenn noch nicht geschehen müssen nun noch die richtigen Dateirechten vergeben werden. Das geht genauso wie bei den CGI-Skripten (siehe oben) oder direkt auf der Kommandozeile. Das Skript benötigt die Dateirechte „750“:

```
cd /usr/lib/cgi-bin/bin/
chmod 750 cron.sh
```

Zur Einrichtung eines Cronjobs ruft man dann auf der Kommandozeile den Befehl `crontab -e` auf. Welche Optionen zur Verfügung stehen kann man in der Hilfeseite des Cron-Daemons nachlesen, die man mit `man 5 crontab` aufruft. Das folgende Beispiel holt jeweils alle 2 Minuten die Daten vom Wundergroundserver ab. Damit überschreitet man die Limitierungen des kostenlosen Wundergroundaccounts gerade so nicht und hat topaktuelle Wetterdaten.

Man lässt vom Crondaemon das gerade editierte Shellskript „`cron.sh`“ aufrufen. Die Pfadangaben eventuell anpassen!

```
# Alle 2 Minuten Wetterdaten holen
 */2 * * * * /usr/lib/cgi-bin/bin/cron.sh
```

Es macht Sinn den Befehl, den man hier angegeben hat, auch einfach mal direkt in der Kommandozeile auszuführen (am Besten per Copy&Paste). Dann sieht man auch, ob irgendwelche Fehlermeldungen kommen.

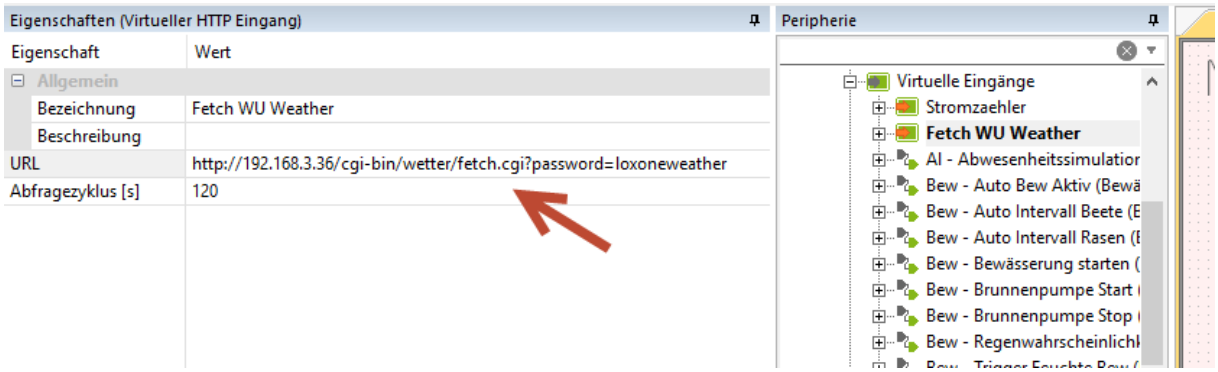
PER LOXONE Miniserver (Variante für einen Server „beim Provider“):

Hat man keine Möglichkeit auf dem eigenen Server/Webspace einen Cronjob einzurichten, so kann man auch vom Miniserver aus den Server dazu auffordern die Wetterdaten vom Wundergroundserver abzuholen.

Dazu legt ihr einen Virtuellen HTML Eingang an und gebt als URL das `fetch.cgi`-Skript an. Der Eingang ruft nun z. B. alle 2 Minuten das `fetch.cgi` Skript auf und der Server holt die Wetterdaten von Wunderground ab.

Wichtig ist noch, dass man aus Sicherheitsgründen noch das Passwort, welches man in der `settings.dat` festgelegt hat, dem `fetch.cgi`-Skript als Option mit übergeben muss. Das dient dazu eine kleine Hürde einzubauen, falls das Skript öffentlich aus dem Internet abrufbar ist. Wer möchte kann das Ganze natürlich noch zusätzlich per htaccess absichern. Anleitungen dazu finden sich über Tante Google. Die Option wird als „`?password=geheimespasswort`“ an den Skriptnamen in der Browseradresszeile angehängt. Also z. B. (Pfad natürlich anpassen!)

http://EUERSERVER/cgi-bin/fetch.cgi**?password=geheimespasswort**



4. Loxone-Programmierung - Per Virtuellem UDP-Eingang

Vorwort/Warnung:

Diese Möglichkeit bietet sich vor allem dann an, wenn der Wetterserver im eigenen Netzwerk steht! Allerdings ist die Loxone Firmware fehlerhaft (Stand: 7.4.4.14) und läasst bei zu vielen übertragenen Daten den Miniserver rebooten. Das scheint immer dann zu passieren wenn der Speicher knapp wird. Beispiel: Überträgt man alle Wetterdaten (alle aktuellen und alle Vorhersagedaten der nächsten 3 Tage und der nächsten 36 Stunden), so werden über 1000 Werte per UDP an den Miniserver übertragen und er rebootet bei mir immer. Übertrage ich nur die aktuellen Daten und die Vorhersagedaten der nächsten 2 Tage und keine Stundenvorher-sage, so sind es nur 120 Werte und alles klappt reibungslos.

Kontrolliert also Euren Miniserver nach Aktivierung am Anfang auf unplanmäßige Reboots!

Allgemeines

Eine Möglichkeit die Daten in LoxoneConfig zu nutzen ist sie vom Wetterserver aus per UDP zum MIniserver zu senden. Das funktioniert (sicher) aber nur, wenn der Wetterserver im lokalen Netzwerk betrieben wird (z. B. auf einem Rapsberry) oder wenn sicher gestellt ist, dass UDP-Pakete vom Webserver zum Miniserver durchgeroutet werden.

Vorbereiten

Als erstes muss man den Wetterserver so vorbereiten, dass er die Daten, die man in LoxoneConfig nutzen möchte, in regelmäßigen Abständen zum Miniserver sendet. Dazu dient das kleine Perl-Programm **`send.pl`**, welches sich im CGI-BIN-Verzeichnis auf dem Wetterserver befindet.

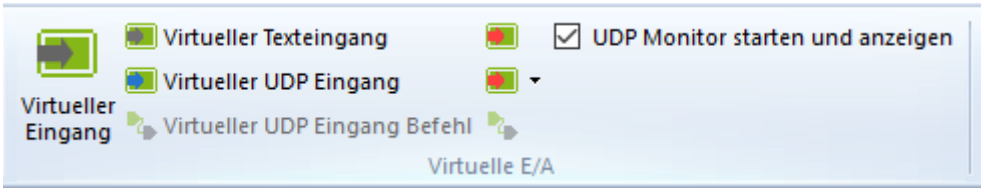
Wichtig in diesem Zusammenhang ist, dass man bei der Installation (siehe Kapitel 3: Server-Installation) in der `settings.dat`-Datei das Senden per UDP aktiviert hat, die korrekte IP-Adresse des Miniservers sowie den zu verwendenden UDP-Port mit angegeben hat. Zudem muss man in der `settings.dat`-Datei auch noch festlegen, welche Werte gesendet werden sollen. Generell werden immer die aktuellen Wetterdaten gesendet, welche Vorhersagedaten noch gesendet werden (also für welche Tage und für welche Stunden) kann man festlegen. Beschränkt es auf das, was ihr wirklich in LoxoneConfig weiterverarbeiten wollt (siehe Warnung oben).

Das Skript `send.pl` wird automatisch jedes Mal aufgerufen, wenn Daten vom Wundergroundserver abgeholt worden sind. Somit werden die Daten dann auch sofort an den Miniserver weitergesendet.

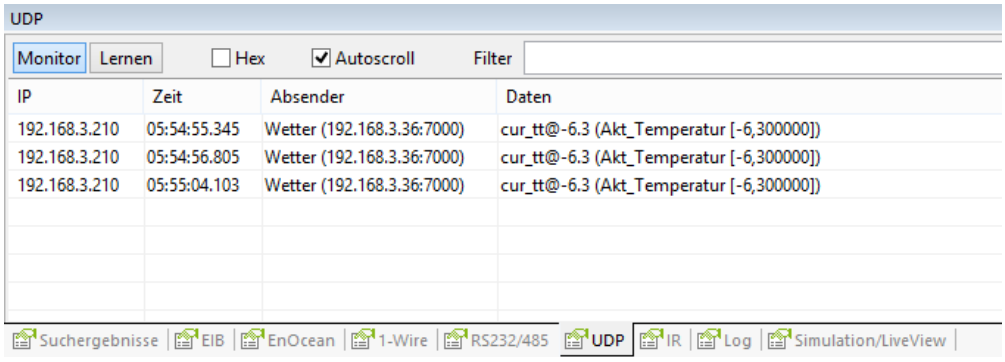
Ein Aufruf des Programms zum Senden kann natürlich auch auf der Kommandozeile manuell erfolgen (Pfad anpassen):

```
cd /usr/lib/cgi-bin/bin
./send.pl -v
```

Um zu schauen, ob die Daten im Miniserver auch ankommen startet man nun LoxoneConfig und verbindet sich mit dem Miniserver. Im Reiter "Miniserver" aktiviert man jetzt den UDP-Monitor (zusätzlich im Peripheriebaum auf Virtuelle Eingänge klicken, sonst erscheint die entsprechende Option nicht im Ribbon):



Dann ruft man das `send.pl`-Skript noch einmal auf. Es sollten dann entsprechenden Einträge im UDP-Log auftauchen:



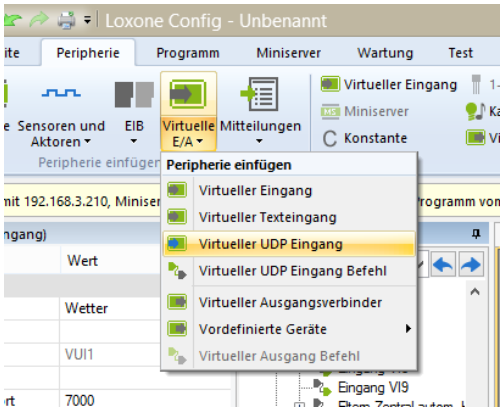
Wie gesagt achtet darauf, ob an dieser Stelle der Miniserver rebootet! Sollte das so sein reduziert die Vorhersagewerte, die gesendet werden oder wechselt auf den Virtuellen HTML Eingang und deaktiviert das Senden per UDP (siehe unten).

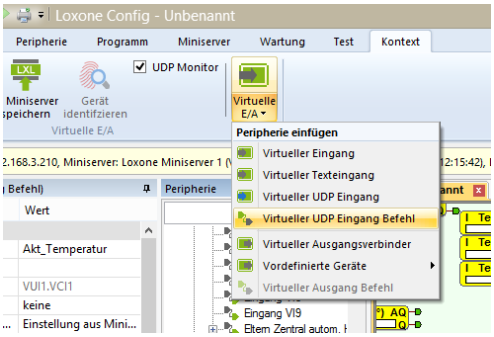
Virtuelle Eingänge in LoxoneConfig

Wenn alle Werte immer schön im UDP-Monitor unter LoxoneConfig auftauchen kann man sich an die Auswertung der Daten machen. Hierzu werden Virtuelle UDP-Eingangsbefehle verwendet. Für jeden Wert muss ein separater UDP-Eingangsbefehl angelegt werden.

Als erstes legt man unter Peripherie -> Virtuelle E/A (oder per Taste F4) einen virtuellen UDP-Eingang an und benennt diesen beliebig (z. B. Wetter). Den Port, auf den der Eingang hören soll, muss man in den Eigenschaften angeben (normalerweise 7000). Dieser muss mit dem Port, den man während der Serverinstallation festgelegt hat, natürlich übereinstimmen.

Anschließend legt man unterhalb des virtuellen Eingangs einen "Virtuellen UDP Eingang Befehl" an und benennt diesen ebenfalls beliebig, sinnvollerweise so, dass man erkennt welcher Wert sich dahinter verbirgt (also z. B. "Aktuelle Temperatur").

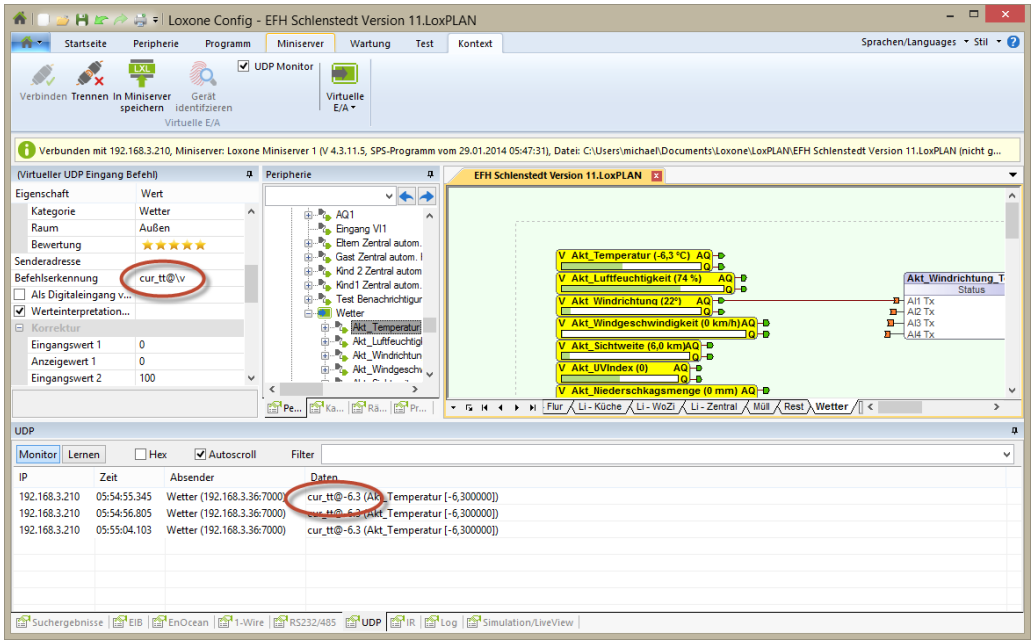




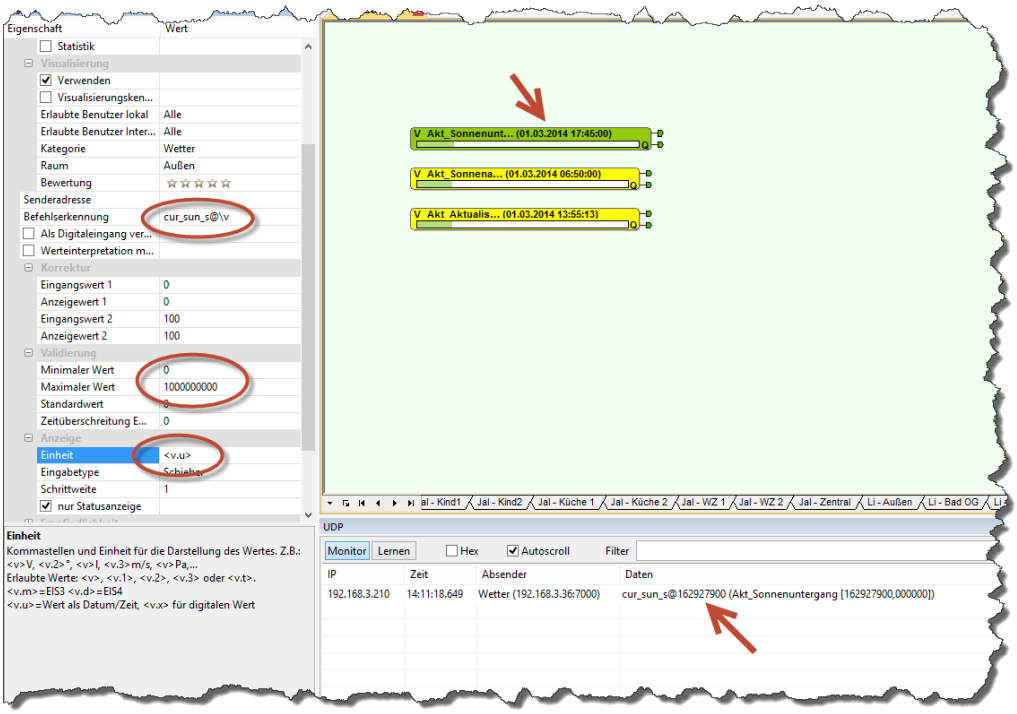
In den Eigenschaften dieses Befehls muss man unter "Befehlserkennung" nun noch eingeben, wie die empfangenen Daten verarbeitet werden sollen. Der Wetterserver sendet für jeden Wert dessen Typ, einen Unterstrich gefolgt von der Abkürzung, gefolgt von einem @-Zeichen und dem eigentlichen Wert. Also für die aktuelle Temperatur "`tt`" z. B. "`cur_tt@-3.9`". Somit gibt man unter Befehlserkennung an:

```
cur_tt@\v
```

Damit erkennt der Miniserver, dass es sich bei dem Wert um die gesendete aktuelle Temperatur handelt und setzt diese als Wert (alles, was nach dem @-Zeichen kommt (`\v`)). Diesen Schritt muss man nun für jeden Wert, den der Wetterserver sendet und den man verarbeiten will, wiederholen. Der Wert bleibt immer so lange aktuell bis der Miniserver einen neuen Wert per UDP übermittelt bekommt.



Eine Besonderheit gibt es bei Werten, die eine Datums-/Uhrzeitangabe enthalten, zum Beispiel die Zeit des Sonnenaufgangs. Hier sendet der Wetterserver die Zeitangabe als Wert in Sekunden seit 01.01.2009. Diese Angabe erwartet der Miniserver so. Im Virtuellen Eingang muss man angeben, dass der empfangene Wert als Datum/Uhrzeit interpretiert werden soll: **<v.u>**. Wenn man die Validierung verwenden möchte muss man unbedingt darauf achten, den Parameter "Maximaler Wert" hoch genug einzustellen (Ideal: 1000000000 = 1+9 Nullen)! Ich empfehle für den Anfang die Validierung zu deaktivieren.



Welche Werte man über den Virtuellen Eingang verwenden kann findet ihr ganz am Ende im Anhang.

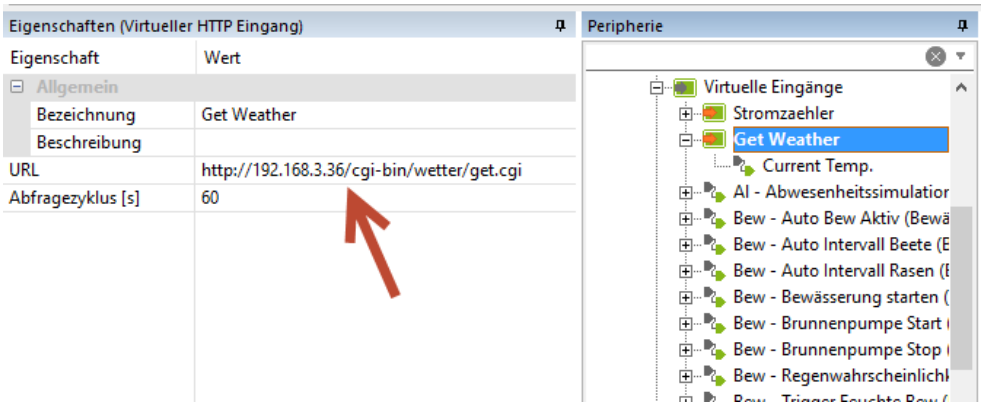
Hinweis: Mit aktuellen Stand der Firmware (7.4.4.14) können die Virtuellen Eingänge keinen Text auswerten (also z. B. „sonnig“)!

5. Loxone-Programmierung - Per Virtuellem HTTP-Eingang

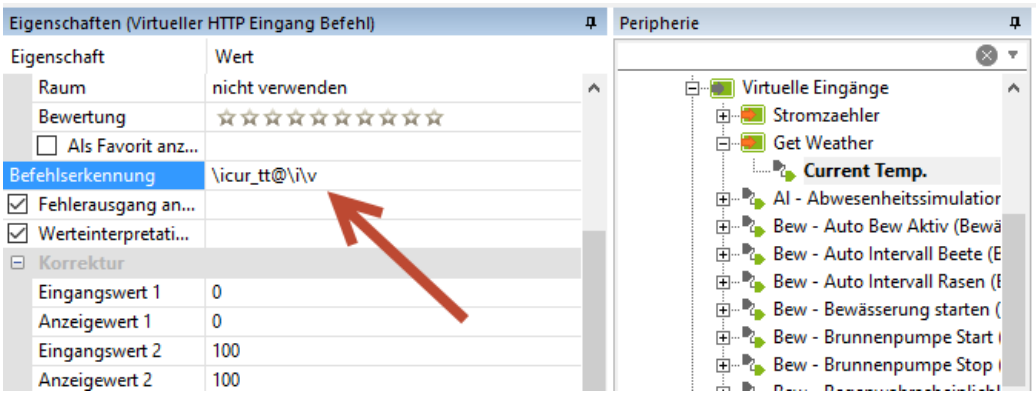
Anstelle von UDP kann man auch einen Virtuellen HTTP-Eingang verwenden. Vorteil ist, dass die Einrichtung im Gegensatz zu UDP häufig einfacher gelingt. Insbesondere wenn der Wetterserver nicht im eigenen Netzwerk betrieben wird (also z. B. auf einem Webserver) sollte man diese Methode verwenden, da man sich dann keine Gedanken über das Routing der UDP-Pakete durch den eigenen Router machen muss. Kleiner Nachteil: Es wird etwas mehr Netzwerk-Traffic verursacht (das fällt aber nicht ins Gewicht).

Die Einrichtung erfolgt analog zur Einrichtung der Virtuellen UDP-Eingänge (siehe oben), als Adresse für den HTTP-Eingang verwendet man das `get.cgi`-Skript auf dem Wetterserver, als Abfragezyklus bietet sich 60 Sekunden an (Pfad natürlich wieder anpassen!):

http://IPADRESSE/cgi-bin/get.cgi

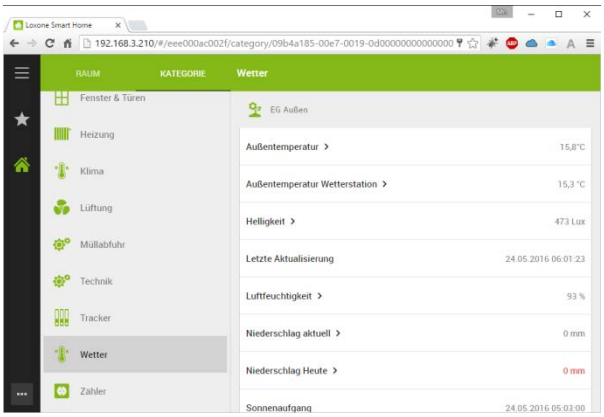


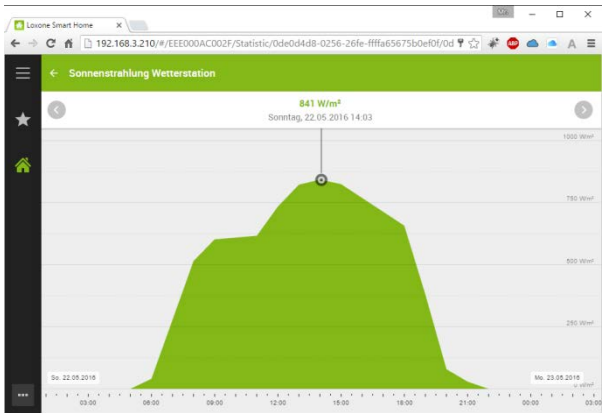
Unterhalb wird analog zum UDP-Eingang ein „Virtueller HTML Eingang Befehl“ angelegt. Nach dem entsprechenden Wetter-Wert sucht man dann mittels **\i .... \i** (siehe Loxone-Doku zum HTTP-Eingang), also für die aktuelle Temperatur z. B. mit diesem String `\icur_tt@\i\v:`



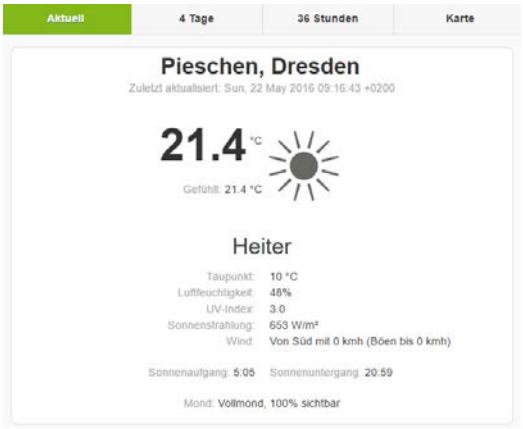
6. Anzeige der Wetterdaten im Browser und der Loxone-App

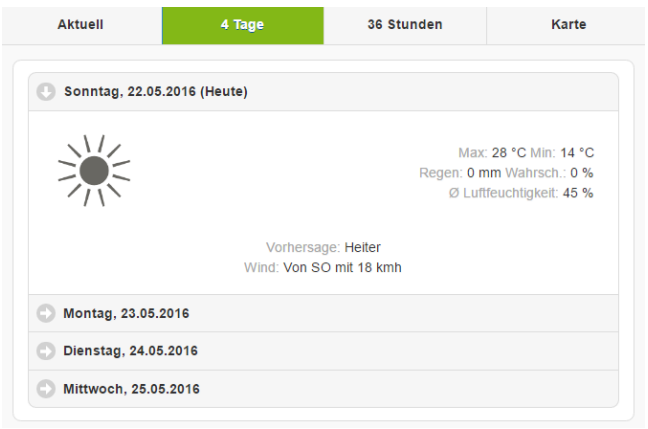
Natürlich kann man alle Werte, die man in LoxoneConfig verwenden kann, über z. B. den Statusbaustein auch in der Visualisierung verwenden. Das sieht z. B. wie folgt aus:





Richtig komfortabel für den „menschlichen Benutzer“ wird es aber erst, wenn die Wetterdaten visuell aufbereitet dargestellt werden können. So bietet es auch der von Loxone buchbare Wetterservice an. Auch LoxoneWeather bietet eine solche Ansicht über eine eingebundene Webseite an (per Webpage-Baustein):





Zur Anzeige dient das in Kapitel 3 installierte Skript "`show.cgi`", welches auf eurem Webserver liegen muss. Das Skript ruft die Wetterdaten aus der eigenen Datenbank ab und nutzt dann diverse Template- und Themedateien, um die Daten aufbereitet darzustellen. Diese Dateien liegen im Unterverzeichnis "`templates`" in eurem cgi-bin-Verzeichnis.

Zur schnelleren Abfrage über die Loxone-Visualisierung wird auch bei jedem Abruf der Wetterdaten vom Wunderground-Server eine gecachete Version der Wetterseite angelegt, damit der Aufruf aus der Visualisierung heraus schneller geht.

Die gecachete Version wird automatisch im WWW-Verzeichnis abgespeichert (näheres weiter unten).

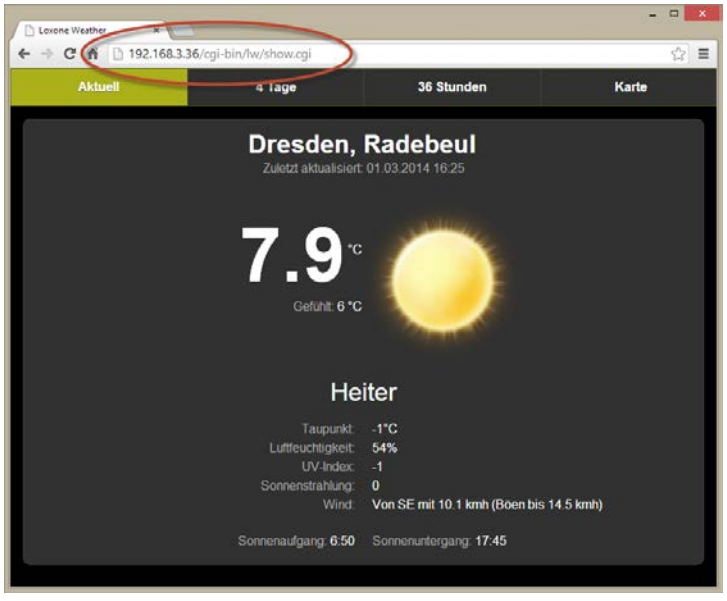
Installation und Test

Zusätzlich zu den Skripten, die ihr bereits in Kapitel 3 installiert habt, benötigt die Anzeige der Wetterdaten noch einige Dateien im normalen Webverzeichnis eures Webspaces oder Raspberrys. Die Dateien findet ihr im Verzeichnis "`www`" im LoxoneWeather-Programmarchiv. Am Besten ladet ihr das komplette Unterverzeichnis "loxoneweather" ins Wurzelverzeichnis auf euren Webspace hoch (Hinweis: Auf einem Raspberry ist das normalerweise das Verzeichnis `/var/www`). Wenn ihr das Verzeichnis anders benennt, müsst ihr die Optionen in der `settings.dat`-Datei entsprechend anpassen (siehe Kapitel 3).

Der Webserver muss teilweise Schreibrechte auf das Verzeichnis haben. Wenn Ihr auf einem Webserver arbeitet sollten keine Anpassungen notwendig sein, **Solltet Ihr die Skripte auf einem RaspberryPi installieren** und habt alle Dateien als Root auf den Server gespielt, solltet Ihr jetzt alle Dateien dem Benutzer des Webservers ("www-data") zuordnen, damit der Webserver auch die Berechtigung hat die Dateien zu verändern. Dazu gebt Ihr folgenden Befehl ein (Pfadangabe muss eventuell angepasst werden):

```
chown -R www-data.www-data /var/www/*
```

Anschließend sollte bereits alles erledigt sein und ihr könnt die Anzeige der Daten testen. Ruft dazu im Browser einfach das `show.cgi`-Skript im cgi-bin-Verzeichnis auf und ihr solltet eine entsprechende Ausgabe sehen (Pfadangabe natürlich wie immer anpassen!):



Dem Skript kann man noch 3 Parameter mit übergeben, um die Anzeige zu beeinflussen und seinen eigenen Wünschen anzupassen. Zum Einen kann man ihm das zu verwendende Theme mit übergeben. Aktuell habe ich 2 Themes erstellt: Eines im Design der Loxone Classic-App und eines im Design der neuen LoxoneApp ab Version 4. Wird keine Option angegeben, wird immer das Default-Theme verwendet (siehe dazu die Optionen in der `settings.dat`-Datei). Die notwendige Option wird als Parameter "`?theme=THEMENAME`" an die URL angehängt. Also z. B.

http://192.168.3.210/cgi-bin/show.cgi**?theme=LoxoneClassic**

Diese Themes stehen aktuell zur Verfügung:

LoxoneClassic: Im Design der Loxone Classic-App

LoxoneV4: Im Design der neuen LoxoneApp ab Version 4

Ein weiterer Parameter ist das Iconset, welches verwendet werden soll. Mitgeliefert werden 7 Iconsets. Das Iconset wird über den Parameter „`&iconset=dark`“ in der URL ausgewählt:

http://192.168.3.210/cgi-bin/show.cgi**?**theme=LoxoneClassic**&iconset=dark**

Diese Iconsets stehen aktuell zur Verfügung. Möchte man das Look&Fell des Original-Loxone-Wetterdienstes imitieren, eignen sich besonders gut das Iconset „dark“ für das Theme LoxoneV4 (heller Hintergrund) und das Set „light“ für das Theme LoxoneClassic (dunkler Hintergrund).

color:

dark:

flat:

green:

light:

realistic: (entnommen aus Kodi/XBMC, © Mominur Rahman (www.illuminatedimages.co.uk), licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.)

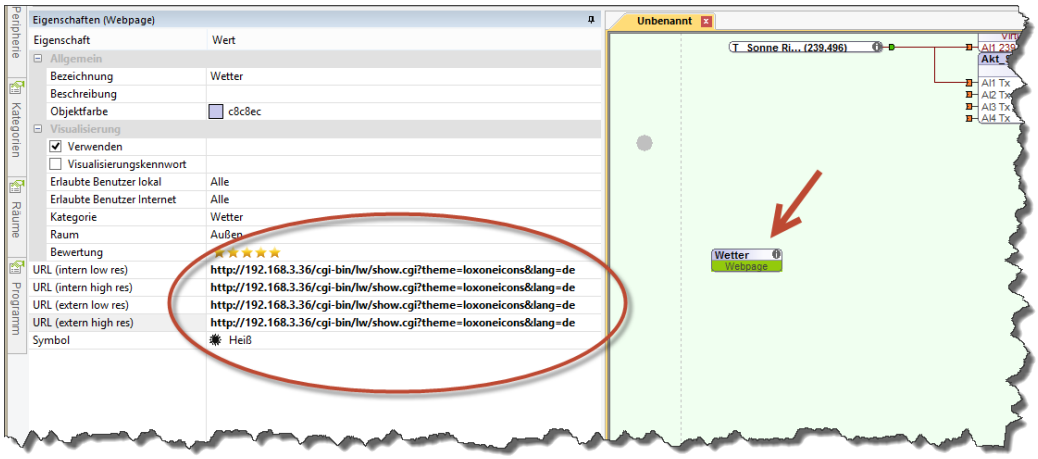
silver:

Der Name des Iconsets ist identisch mit dem Ordnernamen im WWW-Ordner, in dem sich das Iconset befindet (vgl. Ordner `/var/www/loxoneweather/icons`). So kann man ganz leicht auch eigene Iconsets hinzufügen. Die einzelnen Icons müssen nur identisch zu den vorhandenen Icons benannt werden und sich in einem eigenen Unterverzeichnis befinden. Es muss jeweils ein komplettes Set für die Tag-Darstellung (Unterordner „d“) und ein Set für die Nachtdarstellung (Unterordner „n“) existieren.

Als letztes kann man noch die zu verwendende Sprache mit `"&lang=de"` festlegen. Hier gibt es aktuell leider nur "de" als Option.

Integration in LoxoneConfig

Die Integration erfolgt über den Webpage-Baustein. Man findet ihn unter Programm -> Allgemein -> Webpage oder per "F5 -> Webpage". Hier muss man unter URL nur die gleiche URL eingeben, die man eben schon im Browser verwendet hat. Diese legt man für alle Auflösungen gleich fest.



Empfehlung:

Bei jedem Abruf vom Wundergroundserver wird eine gechachete Version dieser Wetterseite erstellt. Das hat den Vorteil, dass sie aus der Visu deutlich schneller angezeigt wird als wenn man sie „Live“ über das `show.cgi`-Skript erstellen lässt. Die gecachete Version findet ihr im WWW-Ordner unter `/var/www/loxoneweather/cachedweather.html`

Wenn ihr diese Seite anstelle des `show.cgi`-Sktiptes im Webpagebaustein verwendet, erfolgt der Aufruf der Seite deutlich schneller!

Todo

Die Ausgabe der Wetterdaten ist aktuell nur für das iPAD1 optimiert. Im Browser kann man die Anzeige ebenfalls verwenden. Für ein iPhone ist sie im Moment nicht zu gebrauchen. Die Integration erfolgt eventuell später noch. Auf einem Android-Tablet muss man sehen, wie die Themes optisch wirken.

Anpassen der Themes

Jeder kann die Themes, Kartendienste oder Übertzungen selbst anpassen oder weitere Themes erstellen. Alle Dateien liegen im Verzeichnis "`templates`" im `cgi-bin`-Verzeichnis.

Theme:

Jedes Theme besteht aus 4 Dateien. Diese können individuell gestaltet werden. Die Funktion sollte sich selbst ergeben.

Die Beispielthemes benutzen JQUERY zur Anzeige. Hier kann man mit ein wenig HTML-Kenntnissen die Anzeige an die eigenen Bedürfnisse anpassen. Variablen aus dem Wetterdienst werden in der Form `<!--$cur_tt-->` angegeben (hier für die aktuelle Temperatur -> siehe Tabellen im Anhang).

Die ersten beiden Buchstaben der Dateinamen stehen für die Sprache des Themes. So kann man auch Themes in unterschiedlichen Sprachen erstellen.

7. Anhang: Verfügbare Daten

Werte-Typ	Variable	Beschreibung	Einlesen per Virt. Eingang (UDP/HTML)	Verwendung im Theme durch show.cgi
Aktuelle Wetterdaten (**cur**rent)	cur_date	Date Epoche	Ja	Ja
	cur_date_des	Date RFC822	Nein	Ja
	cur_date_tz_des_sh	Timeezone Short	Nein	Ja
	cur_date_tz_des	Timezone Long	Nein	Ja
	cur_date_tz	Timezone Offset	Nein	Ja
	cur_day	Date Day	Ja	Ja
	cur_month	Date Month	Ja	Ja
	cur_year	Date Year	Ja	Ja
	cur_hour	Date Hour	Ja	Ja
	cur_min	Date Minutes	Ja	Ja
	cur_loc_n	Observation Location	Nein	Ja
	cur_loc_c	Location Country	Nein	Ja
	cur_loc_ccode	Location Coun.Code	Nein	Ja
	cur_loc_lat	Location Latitude	Ja	Ja
	cur_loc_long	Location Longitude	Ja	Ja
	cur_loc_el	Location Elevation	Ja	Ja
	cur_tt	Temperature	Ja	Ja
	cur_tt_fl	Feelslike Temp	Ja	Ja
	cur_hu	Rel. Humidity	Ja	Ja
	cur_w_dirdes	Wind Dir Description	Nein	Ja
	cur_w_dir	Wind Dir Degrees	Ja	Ja
	cur_w_sp	Wind Speed	Ja	Ja
	cur_w_gu	Wind Gust	Ja	Ja
	cur_w_ch	Windchill	Ja	Ja
	cur_pr	Pressure	Ja	Ja
	cur_dp	Dew Point	Ja	Ja
	cur_vis	Visibility	Ja	Ja
	cur_sr	Solar Radiation	Ja	Ja
	cur_hi	Heat Index	Ja	Ja
	cur_uvi	UV Index	Ja	Ja
	cur_prec_today	Precipitation Today	Ja	Ja
	cur_prec_1hr	Precipitation 1hr	Ja	Ja
	cur_we_icon	Weather Icon	Nein	Ja
	cur_we_code	Weather Code	Ja	Ja
	cur_we_des	Weather Description	Nein	Ja
	cur_moon_p	Moon: % Illuminated	Ja	Ja
	cur_moon_a	Moon: Age of Moon	Ja	Ja
	cur_moon_ph	Moon: Phase of Moon	Nein	Ja
	cur_moon_h	Moon: Hemisphere	Nein	Ja
	cur_sun_r	Sunrise	Ja (Zeit)	Ja
	cur_sun_s	Sunset	Ja (Zeit)	Ja
Tagesgenaue Vorhersage (**d**aily **f**ore**c**ast) HEUTE	dfc0_per	Period (0: Today…)	Ja	Ja
	dfc0_date	Date Epoche	Ja	Ja
	dfc0_day	Date DAY	Ja	Ja
	dfc0_month	Date MONTH	Ja	Ja
	dfc0_monthn	Date MONTHNAME	Nein	Ja
	dfc0_monthn_sh	Date MONTHN. Short	Nein	Ja
	dfc0_year	Date: YEAR	Ja	Ja
	dfc0_hour	Date: HOUR	Ja	Ja
	dfc0_min	Date: MINUTES	Ja	Ja
	dfc0_wday	Date: WEEKDAY	Nein	Ja
	dfc0_wday_sh	Date: WEEKD. Short	Nein	Ja
	dfc0_tt_h	High Temperature	Ja	Ja
	dfc0_tt_l	Low Temperature	Ja	Ja
	dfc0_pop	% of Precipitation	Ja	Ja

	dfc0_prec	Precipitation Forecast	Ja	Ja
	dfc0_snow	Snow Forecast	Ja	Ja
	dfc0_w_sp_h	Max. Wind Speed	Ja	Ja
	dfc0_w_dirdes_h	Max. Wind Dir Descript.	Nein	Ja
	dfc0_w_dir_h	Max. Wind Dir	Ja	Ja
	dfc0_w_sp_a	Ave. Wind Speed	Ja	Ja
	dfc0_w_dirdes_a	Ave. Wind Dir Descript.	Nein	Ja
	dfc0_w_dir_a	Ave. Wind Dir	Ja	Ja
	dfc0_hu_a	Ave. Humidity	Ja	Ja
	dfc0_hu_h	Max. Humidity	Ja	Ja
	dfc0_hu_l	Low. Humidity	Ja	Ja
	dfc0_we_icon	Icon Name	Nein	Ja
	dfc0_we_code	Weather Code	Ja	Ja
	dfc0_we_des	Weather Description	Nein	Ja
Tagesgenaue Vorhersage (**d**aily **f**ore**c**ast) +1 Tag +2 Tage + 3 Tage	Variablen beginnen mit dfc1_ dfc2_ usw. Sonst: Siehe oben	Siehe oben	Siehe oben	Siehe oben
Stundengenaue Vorhersage (**h**ourly **f**ore**c**ast) +1 Stunde	hfc1_per	Period (1: +1 Hour,…) Date	Ja	Ja
	hfc1_date	Epoche	Ja	Ja
	hfc1_day	Date: DAY	Ja	Ja
	hfc1_month	Date: MONTH	Ja	Ja
	hfc1_monthn	Date: MONTHNAME	Nein	Ja
	hfc1_monthn_sh	Date: MONTHN. Short	Nein	Ja
	hfc1_year	Date: YEAR	Ja	Ja
	hfc1_hour	Date: HOUR	Ja	Ja
	hfc1_min	Date: MINUTES	Ja	Ja
	hfc1_wday	Date: WEEKDAY	Nein	Ja
	hfc1_wday_sh	Date: WEEKD. Short	Nein	Ja
	hfc1_tt	Temperature	Ja	Ja
	hfc1_tt_fl	Feelslike Temperature	Ja	Ja
	hfc1_hi	Heat Index	Ja	Ja
	hfc1_hu	Humidity	Ja	Ja
	hfc1_w_dirdes	Wind Dir. Description	Nein	Ja
	hfc1_w_dir	Wind Dir.	Ja	Ja
	hfc1_w_sp	Wind Speed	Ja	Ja
	hfc1_w_ch	Windchill	Ja	Ja
	hfc1_pr	Pressure	Ja	Ja
	hfc1_dp	Dewpoint	Ja	Ja
	hfc1_sky	Sky %	Ja	Ja
	hfc1_sky_des	Sky Description / WX	Nein	Ja
	hfc1_uvi	UV Index	Ja	Ja
	hfc1_prec	Quant. Precipitation FC	Ja	Ja
	hfc1_snow	Snow Forecast	Ja	Ja
	hfc1_pop	% of Precipitation (%)	Ja	Ja
	hfc1_we_code	Weather Code	Ja	Ja
	hfc1_we_icon	Icon Name	Nein	Ja
	hfc1_we_des	Weather Description	Nein	Ja
Stundengenaue Vorhersage (**h**ourly **f**ore**c**ast) +2 Stunden ….. +36 Stunden	Variablen beginnen mit hfc2_ hfc3_ usw. Sonst: Siehe oben	Siehe oben	Siehe oben	Siehe oben