Inhalt

[Solvis-SmartHome-Server / Fhem-Client 1](#_Toc27557008)

[Einführung 1](#_Toc27557009)

[Voraussetzungen 1](#_Toc27557010)

[Solvis Anlage, SolvisRemote 1](#_Toc27557011)

[Server 1](#_Toc27557012)

[SmartHome-System 2](#_Toc27557013)

[Verwendete Schnittstellen der Solvis-Anlage 2](#_Toc27557014)

[Interne Komponenten des SolvisSmartHomeServer 3](#_Toc27557015)

[Server 3](#_Toc27557016)

[Messwerte-Erfassung 3](#_Toc27557017)

[Auswertung und Steuerung über die SolvisControl-Bildschirme 3](#_Toc27557018)

[Ablauf des Programms nach dem Start 3](#_Toc27557019)

[Phase 1: Learning der Bildschirme (nur beim ersten Start) 3](#_Toc27557020)

[Phase 2: Learning der Status-Symbole (nur beim ersten Start) 4](#_Toc27557021)

[Phase 3: Auslesen der aktuellen Anlageparametern 4](#_Toc27557022)

[Besonderheiten 4](#_Toc27557023)

# Solvis-SmartHome-Server / Fhem-Client

## Einführung

Die SolvisRemote bietet einen Zugriff über mehrere http-Seiten an. Über diese kann der aktuelle Status der Anlage untersucht werden und Einstellungen der Anlage verändert werden.

Dieser Zugriff eigen sich jedoch nicht direkt zur Integration in ein SmartHome-System wie Fhem, ioBroker, OpenHAB u.a..

Bisher existierte für das SmartHome-System FHEM ein Modul, das die Messwerte der Anlage in FHEM zugänglich machen konnte sowie den Anlagenmodus (Tag/Nacht/Standby/Timer) verändern konnte. Letzteres funktionierte nicht zuverlässig.

Ziel des vorliegenden Projektes war, auch weitere Anlagenparameter von einem SmartHome-System einstellen zu können und möglichst eine Insel-Lösung nur für das Fhem-SmartHome-System zu vermeiden.

Bei dem neuen Modul handelt es sich daher um eine Server-Client-Lösung.

Der Server dient zum Auslesen der Messwerte und Auslesen/Einstellen der Anlagenparameter. Der Server ist in Java geschrieben, da aus meiner Sicht für größere Projekte eine Script-Sprache (wie Perl/JavaScript u.a.) wenig geeignet ist und ich in den letzten Jahren beruflich viel in Java programmiert hatte.

Mit diesem Server können sich mehrere SmartHomeClients gleichzeitig verbinden. Auf diese Weise kann man von verschiedenen SmartHome-Systemen den Server ansprechen. Die etwas CPU-zeitintensive Verarbeitung erfolgt nur an einer Stelle, dem Server.

Auf der SmartHome-Seite ist nur noch ein relativ einfacher an das verwendete System angepasster Client zu realisieren.

Der Datentransfer zwischen Server und Client erfolgt über das JSON-Format, welche recht einfach über eine Library eingelesen/erstellt werden können. Das Format im Einzelnen ist im vorliegenden Dokument im Anhang beschrieben.

## Features

Der neue Server bietet folgende Features:

* Auslesen der Messwerte der Sensoren
* Einstellung der Anlagenparameter wie Sollwerte, Raumabhängigkeiten etc.
* Monitoring der Solvis-Uhr und bei größeren Abweichungen (>40s) erfolgt eine entsprechende Nachjustierung.
* Anbindung über eine Client-Server-Verbindung, dadurch leichte Anpassung an andere Smarthome-System
* Daten zwischen Server-Client werden im JSON-Format ausgetauscht
* Es können sich max. 50 Clients mit dem Server verbinden
* Leichte Anpassungsmöglichkeit an vorhandene Anlage über ein XML-Files. Das Style-Sheet ist mit enthalten, so dass Anpassung mittels XML-Editor (z.B. integriert in Eclipse) stark vereinfacht wird

## Voraussetzungen

### Solvis Anlage, SolvisRemote

Grundvoraussetzung zur Verwendung der vorliegenden Lösung ist natürlich eine **Solvis-Anlage** sein, welche die **SolvisControl 2** (seit 9/2007) verwendet (SolvisMax, SolvisBen), die **SolvisRemote** muss noch zusätzlich vorhanden sein.

### Server

Der Server kann auf verschieden Systemen laufen, für die es ein Java-Run-Time-Environment gibt. Entwickelt wurde es auf einem Windows-System mit Oracle-JDK 8. Im Einsatz ist es auf einen Raspberry Pi 3 Modell B, getestet wurde es auf einem Raspberry Pi 2 Modell B, jeweils mit OpenJDK 9. Auf einem Raspberry Pi 2 kostet es etwa 0,5% der CPU-Zeit, höchst wahrscheinlich wird es auch noch auf einem Raspberry der ersten Generation lauffähig seine, was nicht getestet wurde.

Der Speicherverbrauch im Betrieb (einschl. OpenJDK) liegt etwa bei 60 MByte Speicher (auf dem Raspberry Pi 2, ermittelt mit *top*).

### SmartHome-System

Zusätzlich ist natürlich ein SmartHome-System mit einem entsprechenden Client notwendig.

Aktuell existiert nur für das SmartHome-System FHEM ein entsprechendes Modul.

Da ich mit dem Fhem-SmartHome-System nicht wirklich zufrieden bin (ich mag einfach kein Perl), spiele ich auf länger Sicht mit dem Gedanken auf ein anderes System evtl. OpenHAB umzusteigen.

Auf dem Server kann natürlich ebenfalls das SmartHome-System laufen.

## Verwendete Schnittstellen der Solvis-Anlage

Die Messwerte der Anlage, welche im Anlagenschema angezeigt werden, können noch recht gut unter der folgenden Adresse als Hex-String verpackt einem vereinfachten XML-Rahmen ausgelesen werden.

Dies kann man unter der folgenden Adresse auslesen:

http://<tcp-ip-Adresse der Solvis-Anlage>/sc2\_val.xml?

Dieser String wurde bisher durch das Fhem-Modul „73\_SolvisMax.pm“ ausgewertet um die Daten auf der FHEM-Oberfläche darstellen zu können.

Über diesen Weg lassen sich jedoch nicht die Anlagenparameter – wie Tag-/Nacht-Temperatur, Raumeinfluss etc. – verändern. Das geht nur über die SolvisControl, welche unter folgender Adresse zugänglich ist:

http://<tcp-ip-Adresse der Solvis-Anlage>/remote.html

Das bisherige FHEM-Modul „73\_SolvisMax.pm“ ließ hier nur sehr rudimentäre Zugriffe auf die SolvisConrol zu, es waren nur die Anlagenmodus Tag/Nacht/Timer/Standby wählbar. Ab und zu erkannte die SovisControl einer dieser Betätigungen nicht, so dass man es wiederholen musste, für ein zuverlässiges SmartHomeSystem nicht nicht geeignet.

Das vorliegende neue Modul nutzt auch die obigen beiden Wege, erweitert den Weg über die SolvisControl um die Interpretation des Bildschirminhaltes um in Abhängigkeit vom Bildschirminhalt die Buttons der SolvisControl passend zum einzustellenden Wert bedienen zu können. Der dann eingestellt Wert wird immer verifiziert, so dass verloren gegangene Button-Betätigungen erkannt werden.

Der Bildschirminhalt der SolvisControl wird folgender Http-Zugriff gelesen:

http://<tcp-ip-Adresse der Solvis-Anlage>/ display.bmp?

Der Ursprung des Koordiantensystems ist wie bei Bildern üblich oben links.

Die Betätigung der << Buttons wird folgender Http-Zugriff verwendet:

http://<tcp-ip-Adresse der Solvis-Anlage>/Taster.CGI?taste=links

Zum Betätigen eines angezeigten Buttons auf dem Bildschirm erfolgt über folgenden Http-Zugriff:

http://<tcp-ip-Adresse der Solvis-Anlage>/Touch.CGI?x=<x>&y=<y>

Hierbei sind <x> und <y> die Koordinaten des Buttons aus dem obigen Bild multipliziert mit zwei (auf der http-Seite wird das Bild der SolvisControl um den Faktor 2 vergrößerst dargestellt.

## Interne Komponenten des SolvisSmartHomeServer

Der SolvisSmartHomeServer besteht aus 3 Funktionseinheiten

### Server

Der eigentliche Server stellt die Schnittstelle nach außen dar. Er nimmt Verbindungen von bis max. 50 Clients entgegen, interpretiert deren Befehle und sendet die Solvis-Daten an die Clients.

### Messwerte-Erfassung

Dieser fragt regelmäßig (default alle 10s) den Solvis-Hex-String mit den Messwerten ab, interpretiert den Hexstring und sendet bei einer Änderung dem Client die gemessenen Werte. Für bestimmte Daten (Temperaturen) erfolgt eine Mittelwertbildung über eine Messwerte-Reihe (Default: 12, entspricht über einen Zeitraum von 2 Minuten). Erkennt dabei das Modul einen Wert, der vom Mittelwert stärker abweicht als die normale Schwankungsbereich des Sensors, wird der Wert bei der Mittelwertbildung doppelt gewichtet, so dass der vom Modul gelieferte Wert trotz Mittelwert-Bildung dem wirklichen Wert bei größeren Änderungen besser folgt (z.B. Aufheizung des Kesselwassers durch laufenden Brenner).

### Auswertung und Steuerung über die SolvisControl-Bildschirme

Zur Interpretation des Bildschirminhalts wurde ein abgespecktes OCR realisiert, das folgende Zeichen der SolvisControl erkennen kann:

+ - 0 1 2 3 4 5 6 7 8 9 ° C [ ] : . / h %

Zur Identifikation der einzelnen Screens werden gezielt bestimmte rechteckige Flächen der Screen untersucht. Zusätzlich kann auch der Identifikation das OCR herangezogen werden. So wird für Screens der Heizkreise nur die Überschrift herangezogen, welche einzelne Screen gerade angezeigt wird, wird durch die Detektion der

## Ablauf des Programms nach dem Start

Das Programm durchläuft nach dem Start vier verschiedene Phasen. Erst in der dritten Phase sind sämtliche Messwerte eingelesen und das Modul ist zur Steuerung der Solvis-Anlage bereit.

### Phase 1: Learning der Bildschirme (nur beim ersten Start)

Beim ersten Start des Programmes und nach Änderung des „control.xml“-Files müssen die Grafiken angelernt werden, die zur Identifikation der Bildschirme benötigt werden. Das erfolgt vom Server Großteils vollautomatisch, in dem es durch die verschiedenen Bildschirme der SolvisControl geht und sich dabei die zur Identifikation notwendigen Bildschirmbereiche merkt. Diese werden in die Datei „graficData.xml“ gespeichert, so dass die Learning-Phase nur beim ersten Starten des Moduls durchlaufen wird.

### Phase 2: Learning der Status-Symbole (nur beim ersten Start)

Beim ersten Start des Programmes und nach Änderung des „control.xml“-Files müssen die Grafiken angelernt werden, die zur Identifikation der Status benötigt werden. Das erfolgt durch das Modul Großteils vollautomatisch, in dem es durch die verschiedenen Bildschirme der SolvisControl geht und sich dabei die zur Identifikation des Status notwendigen Symbole merkt. Diese werden in die Datei „graficData.xml“ gespeichert, so dass die Learning-Phase nur beim ersten Starten des Moduls durchlaufen wird.

**Bei diesem Vorgang werden temporär die Betriebszustände der Anlage verändert!**

Der Server stellt nach dem Anlernen der Symbole des jeweiligen Betriebsmodus diesen wieder auf den ursprünglichen zurück, kurzzeitig befindet sich die Anlage jedoch in einem anderen Zustand, das sollte beachtet werden um entsprechend vorher oder nachher einzugreifen.

### Phase 3: Auslesen der aktuellen Anlageparametern

In dieser Phase beginnt das zyklische Auslesen der Messwerte. Der Client erhält entsprechend die Werte.

Gleichzeitig erfolgt das Auslesen der Anlageparameter von der SolvisControl. Dies kann je nach Konfiguration einige Minuten in Anspruch nehmen.

### Besonderheiten

Es gibt einige berechnete Werte, welche genauere Werte liefern, als die auch der SolvisControl angezeigten. Dazu gehören die Brennerlaufzeiten. Diese müssen aber regelmäßig mit den Werten abgeglichen werden, welche im SolvisControl angezeigt werden, da diese andernfalls auseinanderlaufen. Dazu erfolgt entsprechenden dem Messwert einen Synchronisation. Diese bewirkt, dass in dieser Phase die SolvisControl innerhalb eines kurzen Zeitintervalls auf der Anzeige mit dem zu synchronisierenden Werte eingestellt bleibt. Werden die Werte das erste Mal synchronisiert, kann diese Zeit länger dauern.

Schnittstelle Server – Client

Die Kommunikation zwischen Server und Client basiert auf JSON.

Die Schnittstelle basiert aktuell auf folgende Strukturen, welche im JSON-Format übertragen werden:

1. CONNECT Verbindungsaufbau Client -> Server
2. RECONNECT Wiederaufbau der Verbindung Client -> Server
3. CONNECTED Quittierung für den Aufbau der Verbindung Server -> Client
4. DISCONNECT Verbindungsabbau (nicht verwendet) Client -> Server
5. CONNECTION\_STATE Status der Verbindung (ALIVE etc.) Server -> Client
6. SOLVIS\_STATE Status der Solvis-Anlage (PowerOff etc.) Server -> Client
7. CHANNEL\_DESCRIPTIONS Beschreibung (Meta-Daten) der Kanäle Server -> Client
8. MEASUREMENTS Paket mit den Messwerten Server -> Client
9. SET Ändert einen Anlagenparameter Client -> Server
10. GET Stößt das Auslesen eines Anlagenparameters an Client -> Server
11. SERVER\_COMMAND Befehl für den Server (z.B. Backup der Messdaten) Client -> Server

**Der erste Verbindungsaufbau erfolgt wie folgt:**

1. Client -> Server CONNECT mit Namen der Solvis-Anlage
2. Server -> Client CONNECTED mit Client-ID
3. Server -> Client CHANNEL\_DESCRIPTIONS
4. Server -> Client MEASUREMENTS
5. Server -> Client SOLVIS\_STATE

Anschließend treffen beim Client dann neue Datenpakete ein, wenn sich die entsprechenden Messwerte geändert haben (MEASUREMENTS ) oder der Status der Solvis-Anlage geändert hat (SOLVIS\_STATE).

Der Client kann SET- und GET-Befehle senden.

Ist der Name der Solvis-Anlage unbekannt, liefert der Server ein CONNECTION\_STATE-Paket mit dem ConnectionStatus „CONNECTION\_NOT\_POSSIBLE“.

Nach einer Unterbrechung der Verbindung kann der Client die Verbindung wiederaufbauen oder einen ganz neue Verbindung initiieren. Letzteres entspricht dem obigen Ablauf.

**Wiederaufbau einer Verbindung:**

1. Client -> Server RECONNECT mit Client-ID
2. Server -> Client MEASUREMENTS
3. Server -> Client SOLVIS\_STATE

Ist der Wiederaufbau nicht erfolgreich (z.B. Client-ID unbekannt), wird ein CONNECTION\_STATE-Paket vom Server zum Client gesendet mit den ConnectionStatus „CLIENT\_UNKNOWN“.

Anschließend werden wieder GET/SET/MEASUREMENTS/SOLVIS\_STATE-Pakete ausgetauscht.

Die Schnittstelle zwischen Server und Client basiert auf JSON. Da das JSON-Format hierarchisch aufgebaut ist und nicht jedes SmartHome-System den JSON-Parser mittels eines Streams versorgen kann, so dass der Parser selber das Ende eines JSON-Files erkennt, wurde ein JSON\_File in einen Frame verpackt.

Dieser Frame sie wie folgt aus:

Übertragungs-Frame

1. 3 Byte mit der Länge des JSON-Files liegt wie in Netzwerken üblich im Big-Endian-Reihenfolge vor (MSB zuerst, LSB zuletzt).
2. JSON-Daten UTF-8 kodiert.

CONNECT

Der Connect-Datensatz ist wie folgt aufgebaut:

{"CONNECT":{"Id":"*Name der Anlage*"," Url ":"*Url der Anlage*","Account":"*Anlagen-Account*","Password":"*Anlagen-Password*"}}

Die Schlüssel Url, Account und Password können auch entfallen, wenn auf der Serverseite die Solvis-Anlage schon bekannt ist (aus der base.xml-Datei).

RECONNECT

Der RECONNECT-Datensatz ist wie folgt aufgebaut:

{"RECONNECT":{"Id":*Client-Id*}}

Die Client-Id ist eine Integer-Zahl, welche der Server im Datenpaket „CONNECTED“ liefert.

CONNECTED

Der CONNECTED-Datensatz ist wie folgt aufgebaut:

{"CONNECTED":{"ClientId":427735588,"ServerVersion":"00.01.00","MinClientVersion":"00.01.00"}}

XML-Files