Solvis-SmartHome-Server

(+ Fhem-Client, IOBroker-Objektlisten)

11.01.2020, letzte Änderung: 21.07.2020

Stefan Gollmer

Inhalt

1 Einführung

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

Dieser Zugriff eigen sich jedoch nicht so ohne weiteres 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 sowie den Anlagenmodus (Tag/Nacht/Standby/Timer) verändern konnte. Letzteres funktionierte nicht zuverlässig. Ein ähnliches gibt es wohl auch für ioBroker.

Ziel des vorliegenden Projektes war, auch weitere Anlagenparameter von einem SmartHome-System einstellen zu können und zusätzlich 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 u.a.) weniger geeignet ist und ich in den letzten Jahren beruflich viel in Java programmiert hatte.

Mit diesem Server können sich mehrere SmartHomeSysteme 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 oder man nutzt dort die MQTT-Schnittstelle des Servers.

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 B beschrieben.

Neben der Server-Client-Lösung ist wie gesagt zusätzlich eine MQTT-Client-Schnittstelle integriert. Mit dieser Lösung kann man der SolvisSmartHomeServer auch ohne einen speziellen Client von den meisten SmartHome-Systemen aus ansprechen, da die meisten SmartHome-System ein entsprechendes Modul besitzen. Die definierten Topics sind im Anhang D beschrieben.

2 Features

Der neue Server bietet folgende Features:

- Auslesen der Messwerte der Sensoren
- Einstellung der Anlagenparameter wie Temperatur-Sollwerte, Raumabhängigkeiten etc.
- Monitoring der Solvis-Uhr und mit entsprechender Nachjustierung.
- Es werden Anwender und Service-Zugriffe auf den Touchscreen der SolvisControl erkannt und nach beenden die möglicherweise veränderten Anlagenparameter wieder erneut gelesen. Für Service-/normaler Anwender existieren unterschiedliche Wartezeiten, ehe dieser Vorgang abläuft.
- 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
- Zusätzliche Möglichkeit der Anbindung über MQTT, damit ist das System in SmartHome-Systeme ohne speziellen Client möglich. Voraussetzung ist nur eine MQTT-Schnittstelle(-Erweiterung) im SmartHome-System.
- Leichte Anpassungsmöglichkeit an vorhandene Anlage über XML-Files. Die XML Schema sind mit enthalten, so dass Anpassung mittels XML-Editor (z.B. integriert in Eclipse) stark vereinfacht wird
- Erkennen eines Fehlerzustandes der Anlage. Im Fehlerfall kann auch optional einen Mail versendet werden. Wird der Fehler als Fehlerscreen von der Anlage angezeigt, wird an die Mail die Hardcopy des Bildschirmes angehängt.

3 Voraussetzungen

3.1 Solvis Anlage, SolvisRemote

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

3.2 Server (auch MQTT-Client)

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 und Raspbian Stretch. 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 sein, was nicht getestet wurde.

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

3.3 SmartHome-System

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

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

Neben der Server-Client-Schnittstelle besitzt der SolvisSmartHomeServer noch die Möglichkeit der Nutzung der MQTT-Schnittstelle. Da viele SmartHome-System einen entsprechenden Client oder gar Broker (Server) beinhalten, ist der Betrieb auch ohne einen speziellen Client möglich.

Für den MQTT-Client des IoBrokers liegt noch eine Objektliste bei.

Auf dem Computer auf dem der Server läuft, kann natürlich ebenfalls das SmartHome-System laufen.

4 Installation

4.1 Java

Der SolvisSmartHome-Server benötigt die Laufzeitumgebung von Java (JRE). Auf Windows-Systemen stehen Installtionsdateien unter folgendem Link zur Verfügung:

https://www.java.com/en/download/

Unter Linux gibt es ebenfalls Oracle-JRE. Ich habe OpenJDK verwendet, es soll zwar weniger performant sein, wegen der neuen Lizenz-Politik von Oracle verwende ich OpenJDK, das wie folgt installiert werden kann (unter Raspbian Stretch, sollte bei anderen Distributionen ähnlich sein):

Mit

apt search openjdk-.-jre**

kann man sich die aktuellen JRE-Versionen für das aktuell installierte Betriebssystem ansehen.

Die aktuell stabile Version 9 für Raspbian (hier Stretch) wird dann wie folgt installiert:

apt install openjdk-9-jre

Die installierte Version kann dann über

java -version

angezeigt werden.

Getestet wurde es auf Windows mit der Oracle-JRE-Version jre1.8.0_251, auf der Linux-Seite (Raspbian) mittels OpenJDK 9.

4.2 SolvisSmartHome-Server Linux-Installation

4.2.1 Dateien des Linux-Installationspaketes

Das Installationspaket beinhaltet folgende Dateien:

• Makefile	Zur Installation
• SolvisSmartHomeServer.jar	Das eigentliche Programm
• SolvisSmartHomeServer.pdf	Diese Dokumentation
• base.xml.new	Steuerdatei mit den Basis-Daten
• base.xsd	XML Schema hierzu
• SolvisSmartHomeServer.service	SystemD-Datei für die Service-Einrichtung
• DebugSolvisSmartHomeServer.service	SystemD-Datei für die Service-Einrichtung im Debug-Mode (Remote-Debugging)
• CHANGES.txt	Datei mit den Änderungen
• mqtt-client.0.SolvisSmartHomeServer.json	JSON-Objekt-Liste für IoBroker unter Verwendung des MQTT-Client
• 73_SolvisClient.pm	FHEM-Modul

4.2.2 Anpassung der Datei *base.xml*

Im Paket liegt die Datei „base.xml.new“. Diese ist in „base.xml“ umzubenennen und dann entsprechend anzupassen.

Die Datei „base.xml“ enthält die wesentlichen Daten, welche vom Anwender anzupassen sind.

Es enthält aktuell (Stand 29.04.2020) folgenden Inhalt:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:BaseData xmlns:tns="http://www.example.org/control"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.example.org/control base.xsd ">
    <tns:ExecutionData
        timeZone="Europe/Berlin"
        port="10735"
        writablePathLinux="/opt/fhem"
        writeablePathWindows="C:\JavaPgms\SolvisSmartHomeServer\log"
        echoInhibitTime_ms="2000"
    />
    <tns:Units>
        <tns:Unit
                id="mySolvis"
                type="SolvisMax"
                configOrMask="0x00"
                fwLth2_21_02A="true"
                account="account"
                passwordCrypt="AES-coded"
                password="password"
                url="http://aaa.bbb.ccc.ddd"
                defaultAverageCount="12"
                measurementHysteresisFactor="4"
                defaultReadMeasurementsInterval_ms="10000"
                forcedUpdateInterval_ms="3600000"
                doubleUpdateInterval_ms="10000"
                bufferedInterval_ms="60000"
                watchDogTime_ms="30000"
                releaseBlockingAfterUserAccess_ms="300000"
                releaseBlockingAfterServiceAccess_ms="3600000"
                delayAfterSwitchingOnEnable="true"
                ignoredFrameThicknesScreenSaver="3">
            <tns:ClockAdjustment
                    enable="true"
                    fineLimitUpper_ms="2000"
                    fineLimitLower_ms="-5000"
                    aproximatlySetAjust_ms="1000"
                    burstLength="0" />  <!—burstLength = 0: fine tuning disabled →
            <tns:Features>
                    <tns:ClockTuning>false</tns:ClockTuning>
                    <tns:HeatingBurnerTimeSynchronisation>false
                        </tns:HeatingBurnerTimeSynchronisation>
                    <tns:UpdateAfterUserAccess>false</tns:UpdateAfterUserAccess>
                    <tns:DetectServiceAccess>false</tns:DetectServiceAccess>
                    <tns:PowerOffIsServiceAccess>false</tns:PowerOffIsServiceAccess>
                    <tns:ClearErrorMessageAfterMail>true</tns:ClearErrorMessageAfterMail>
                    <tns:OnlyMeasurements>true</tns:OnlyMeasurements>
             </tns:Features>
        </tns:Unit>
    </tns:Units>
    <tns:Mqtt
        enable="true"
        brokerUrl="192.168.1.75"
        port="1883"
        subscribeQoS="1"
        publishQoS="1"
        idPrefix="SolvisSmartHomeServer"
        topicPrefix="SolvisSmartHomeServer" />
    <tns:ExceptionMail
        port="587"
        passwordCryp="AES-coded"
        name="Vorname Nachname"
        securityType="TLS"    <!--SSL oder TLS möglich -->
        provider="securesmtp.t-online.de"
        from="Mailadresse@t-online.de">
        <tns:Recipients>
            <tns:Recipient
                address="Mailadresse1@t-online.de"
                type="TO" />   <!—TO, CC oder BCC möglich -->
            <tns:Recipient
                address="Mailadresse2@gmail.com"
                type="TO" />
        </tns:Recipients>
    </tns:ExceptionMail>
</tns:BaseData>
```

Von der Version 1.00.02 an sind die Passwörter in diese Datei AES-256-verschlüsselt einzutragen. Nur für die Solvis-Anlage kann noch das Password im Klartext drin stehen. Um diese Werte zu generieren, ist folgender makefile-Aufruf zu nutzen:

sudo make crypt

Es wird anschließend nach dem zu verschlüsselnden Wort gefragt. Im Anschluss erfolgt die Ausgabe des verschlüsselten Wortes.

Die obigen *markierten* Einträge müssen entsprechend der beim Anwender vorliegenden Voraussetzungen angepasst werden.

writablePathLinux:		Pfad (für Linux), in dem der Server seine Dateien ablegt
writeablePathWindows:		Pfad (für Windows), in dem der Server seine Dateien ablegt
id:		ID der Solvis-Anlage, entspricht in FHEM dem Gerätenamen
fwLth2_21_02A:		Die Solvis-RemoteControl-Firmwares vor der Version 2.21.02A haben einen Bug, der durch Setzen dieses Attributes auf „true“ umgangen wird. Da bei der Umgehung dieses Bugs eine Fehlermeldung ignoriert wird, sollte man auf eine neuere Firmware updaten.
Type:		Typ der Anlage, aktuell nur „SolvisMax“ oder „SolvisBen“ möglich
configOrMask		Dient dazu, die Konfiguration der Solvis-Anlage genauer zu bestimmen. Aktuell sind folgende Werte möglich: 0x00 Standard 0x10 Solaranlage mit externem Plattenwärmetauscher und zwei Pumpen A01 und A07
account:		Account der Solvis-Anlage
passwordCrypt:		Password der Solvis-Anlage AES-256 verschlüsselt
password:		Passwort der Solvis-Anlage **(depricated)**
url:		Url der Solvis-Anlage
ClockTuning:	true:	Die Uhr wird automatisch eingestellt, Genauigkeit ± 31s
tns:EquipmentTimeSynchronisation:	true:	Synchronisation der sekundengenauen Laufzeiten von bestimmten Geräten (z.B. Brenner, Pumpen) der Anlage mit den stündlichen der SolvisControl
UpdateAfterUserAccess:	true:	Nach jedem Anwenderzugriff werden automatisch die Anlagenparameter abgefragt
DetectServiceAccess:	true:	Es werden die Service-Screens "Schornsteinfeger", "Nutzerauswahl", "Nutzerauswahl-Code" besonders behandlt, werden diese erkannt, sind sämtliche Gui-Actionen für 1h deaktiviert. Jeder Anwender-Eingriff in dieser Zeit triggert die Stunde erneut.
PowerOffIsServiceAccess	true:	Wird ein Power-Off der Anlage erkannt, wird der wie ein Zugriff auf die Service-Screens behandelt (siehe vorheriges Element)
SendMailOnError	True:	Wird das Fehlerfenster oder der Fehler-Button erkannt, wird einen Mail versendet
ClearErrorMessageAfterMail	true:	Im Fehlerfall der Solvis-Anlage erscheint eine Message-Box auf dem Bildschirm. Wenn die Mail

		eingerichtet ist wird eine Hardcopy dieser Messagebox per Mail versendet. Ist das erfolgt wird mit der „<“-Taste die MessageBox verlassen und die GUI-Steuerung bleibt erhalten. Im anderen Fall ist die GUI-Steuerung unterdrückt.
OnlyMeasurements:	true:	Es werden sämtliche Gui-Aktivitäten verhindert. Das Modul verhält sich so wie das bisherige, ein aktiver Eingriff vom Server auf die SolvisControl erfolgt nicht

Zu beachten:

Mit der Version ausgelieferten base.xml sind sämtliche neuen Features des Servers (bezogen auf die alte reine Fhem-Lösung) deaktiviert, da die einzelnen Features überraschende Effekte haben können (u. U. unerwartete Reaktionen der SolvisControl “wie von Geisterhand” bis hin zum möglichen Absturz der SolvisControl beim Feature „ClockFineTuning”). Die Features müssen im Features-Element einzeln freigeschaltet werden. Bitte dabei nochmals den Kommentar im XML-File beachten.

Die übrigen Parameter können vom User angepasst werden und haben folgende Bedeutung:

Port:	Port des Servers, über den sich der Client mit dem Server verbinden kann
echoInhibitTime_ms	Je nach SmartHome-System-Realisierung kann es sein, dass vom Server gesendete Werte als Stellwerte vom SmartHome-System zurückgesendet werden. Der Server ignoriert über diese echoInhibitTime die zurückgesendeten Stellwerte mit dem identischen Wert.
defaultAverageCount:	Anzahl der Messwerte, über die der Mittelwert gebildet wird
measurementHysteresisFactor:	Hysteresis-Faktor für Messungen, über die ein Mittelwert gebildet wird. Bei 0 führt jede Änderung zu einer Messwertausgabe, bei != 0 wird abhängig vom Sensorrauschen die Hysteresis eingestellt.
forcedUpdateInterval_ms:	Nach Ablauf dieser Zeit werden sämtliche Messwerte zum Client gesendet. Zum Disablen ist hier 0 einzutragen.
doubleUpdateInterval_ms:	Symmetrisch um den Zeitpunkt, der durch *forcedUpdateInterval_ms* vorgegeben ist, erfolgen Updates sämtlicher Messwerte. Dadurch füllen die Messkurven sauber das FHEM-Diagramm von links nach rechts, ohne Anfangs- und Endlücken.
bufferedInterval_ms:	Bei bestimmten Messwerten kann hiermit die minimale Zeit zwischen der Ausgabe von 2 Messwerten bestimmt werden. Zum Disablen ist hier 0 einzutragen.
releaseBlockingAfterUserAccess_ms	Zeit, in der die GUI-Steuerung des Servers nach Erkennung eines User-Eingriffs deaktiviert ist (default: 5 Minuten)
releaseBlockingAfterServiceAccess_ms	Zeit, in der die GUI-Steuerung des Servers nach Erkennung eines Service-Eingriffs deaktiviert ist (default: 2 Stunden)
delayAfterSwitchingOnEnable:	Bestimmte Sensoren benötigen bei meiner Anlage nach dem Einschalten relativ viel Zeit, bis sie stabile

	Werte anzeigen (Der Raumfühler benötigt 25 min). Mit diesem Parameter kann dieses Verhalten deaktiviert werden.
defaultReadMeasurementsInterval_ms:	Gibt die Zeit vor, in welchem Abstand die Messwerte der Solvis-Anlage gelesen werden.
ignoredFrameThicknesScreenSaver	Da bei manchen Anlagen am Rande des Bildschirms Fragmente zu sehen sind, muss der Rand-Bereich zur Bildschirmschoner-Erkennung ausgeblendet werden. Die Breite dieses Bereichs wird durch diesen Parameter bestimmt (Default: 3 Pixel).

Will man die MQTT-Schnittstelle nutzen, muss man im Mqtt-Tag noch folgende Parameter anpassen:

enable:	True enabled die MQTT-Schnittstelle
brokerUrl:	Url des Brokers (ohne Protokoll)
port	Port dees Brokers, Standard ist 1883
subscribeQoS	Quality of service zum Empfang (0 … 2)
publishQoS	Quality of service zum Sendeng (0 … 2)
idPrefix	Prefix der Mqtt-Id des Servers (wird mit einer eindeutigen Zufallsnummer ergänzt)
topicPrefix	Prefix der MQTT-Topics

Zusätzlich gibt es noch das ExceptionMail-Element, mit dem man die Zugangsdaten des Mail-Providers definiert, sowie die zu benachrichtigenden Adressen.

port:	Port-Nummer des Providers
passwordCrypt:	Passwort AES-256 verschlüsselt
name:	Name des Absenders (z.B. Max Mustermann)
securityType:	TLS oder SSL möglich (TLS bisher nur getestet)
provider:	Url des Providers, z.B. securesmtp.t-online.de
from:	Mailadresse, z.B. Mailadresse1@t-online.de
address:	Empfängeradresse, z.B. Mailadresse2@gmail.com
type:	Art (TO, CC oder BCC)

4.2.3 Installation des Servers mittels Make

Zur Installation dient das beiliegende Makefile.

Es wird wie folgt aufgerufen:

sudo make Installationsart

Die Aufrufmöglichkeiten werden im Anhang D näher aufgeführt, hier werden nur die wichtigsten genannt, welche zur Erstinstallation und später zum Update notwendig sind.

Bei der Erstinstallation wird das Makefile wie folgt starten:

sudo make install

Nur wenn der SolvisSmartHome-Server nicht auf dem gleichen System wie Fhem läuft, sind die anderen installationsarten interessant (Fhem-Client wird nicht installiert bei “sudo make installSolvis”).

4.2.4 Generieren der AES-Schlüssel für das Solvis-Zugriffspasswort sowie für die Mail

Ab Version 1.00.02 sind die Passworte AES-256-Verschlüsselt in die base.xml einzutragen. Nur das Passwort der Solvis-Anlage kann – wegen der Abwärtskompatibilität - NOCH im Klartext eingetragen werden. Auf längere Sicht sollte jedoch auch bei der Solvis-Anlage auf das verschlüsselte Password übergegangen werden. Um die verschlüsselten Passworte zu erhalten, ist das Makefile wie folgt aufzurufen:

sudo make crypt

Es wird dann anschließend nach dem zu verschlüsselnden Wort gefragt. Danach erfolgt die Ausgabe des verschlüsselnden Wertes.

Diesen Befehl kann man auch schon aufrufen, bevor man die base.xml fertig bearbeitet hat.

4.2.5 Senden einer Testmail

Um die Funktionalität der Mail zu testen, kann das Makefile wie folgt gestartet werden:

sudo make testmail

Anschließend wird mit den Daten der base.xml-Datei eine Testmail verschickt. Diese Make-Funktion arbeitet lokal in dem Verzeichnis, in dem das Makefile liegt und berücksichtigt die dort liegende base.xml.

Sollte man beim Austesten des Mail-Versands die base.xml geändert haben, ist anschließend noch ein „sudo make update“ notwendig.

4.2.6 Automatisches Anlernen der Grafiken

Nach „sudo make install“ oder “sudo make installSolvis” kann der Server auf dem System ausgeführt werden.

Wie unter 7.1 und 7.2 erwähnt wird, muss als Erstes der Server die Grafiken der Solvis-Anlage lernen.

Dazu ist erst die Solvis-Anlage wie folgt einzurichten:

- **Unter „Sonstig./Anlagenstatus“ ist der Bildschirm des Warmwasser-Heizkreises auszuwählen**

Nun kann der Learn-Modus mittels des Makefiles wie folgt gestartet werden:

sudo make learn

Nun beginnt das Lernen der Grafiken. Dieses Anlernen ist etwas kritisch. Man sollte dabei die Terminal-Ausgaben als auch den Solvis-Bildschirm im Auge behalten. Da die SolvisControl leider manchmal Touchs verschluckt, kann der Learn-Vorgang fehlerhaft sein. Ich habe sehr viel Aufwand dazu verwendet, dass dies erkannt wird und in den meisten Fällen wird auch richtig darauf reagiert und der fehlgelaufene Lern-Vorgang wird wiederholt. 100%ig ist es noch nicht, ganz selten muss man den Lernvorgang wiederholen.

Sollte er nicht durchlaufen, bitte nochmal versuchen, wenn kein prinzipielles Problem (z.B. mit der speziellen Anlagenkonfiguration) vorliegt, sollte er dann fehlerfrei durchkommen.

Zu beachten:

Während des Anlernens sollte niemand unter der Dusche stehen, da kurz die Warmwasser-Pumpe ein- /ausgeschaltet wird um die Grafik einer eingeschalteten Pumpe zu lernen (ca. 4s). Je nach Kesseltemperatur kann das zu Überraschungen führen.

4.2.7 Starten des Servers auf der Console

Will man die Meldungen des Servers direkt sehen, kann man den Server auch in der Console mittels folgenden Make-Aufrufs starten:

sudo make foreground

Das empfiehlt sich auch für die ersten Beobachtungen nach einer Neuinstallation.

4.2.8 Einrichten des Servers als Service

Ist der Lernvorgang erfolgreich, ist der Server als Service einzurichten. Das erfolgt mit folgendem Makefile-Aufruf:

make installService

Hierbei werden die Datei „SolvisSmartHomeServer.service“ bzw. „DebugSolvisSmartHomeServer.service“ im Verzeichnis /etc/systemd/system abgelegt, anschließend der Service enabled und gestartet.

Der Server sollte nicht gleichzeitig im debug und normalem Mode gestartet sein. Nur der zuerst gestartete würde dann laufen.

U.a. stehen folgende Befehle zur Verfügung, hierbei ist <*server*> im normalen mode „SolvisSmartHomeServer“, im Debug-Mode „DebugSolvisSmartHomeServer“:

- *sudo systemctl status <server>*
  liefert den aktuellen Status des Servers
- *sudo systemctl start <server>*
  Startet den Server
- *sudo systemctl stop <server>*
  Stoppt den Service
- *sudo systemctl restart <server>*
  Beendet den Server und startet ihn wieder neu
- *sudo systemctl disable <server>*
  Disabled den Dienst, der Server wird nach dem nächsten Boot-Vorgang nicht mehr automatisch gestartet
- *sudo systemctl enable <server>*
  Enabled den Dienst, der Server wird nach dem nächsten Boot-Vorgang wieder automatisch gestartet

Der Server sollte nun bei jedem Bootvorgang gestartet werden.

4.2.9 Einrichten des Fhem-Clients

Die Datei „73_SolvisClient.pm“ wurde durch das Makefile in das FHEM-Verzeichnis kopiert.

Läuft FHEM auf einem anderen System kann das durch

sudo make installFHEM

erfolgen.

Nach einem FHEM-Neustart kann das Modul mittels folgender Define-Anweisung in FHEM eingebunden werden:

define <*Anlagen-Id wie in base.xml*> SolvisClient <*tcp-ip-Adresse des Servers*>:10735

In der Regel wird der Server auf dem System des SmartHome-Systems laufen. Dann reicht folgende Anweisung:

define <*Anlagen-Id wie in base.xml*> SolvisClient localhost:10735

Zum ersten Einrichten reicht das schon. Die Readings der Solvis-Anlage sollten jetzt schon erscheinen. Auch die entsprechenden Pull-Down-Menüs für die möglichen SET/GET-Befehle sollten jetzt auswählbar sein.

Die Dokumentation des FHEM-Moduls ist wie üblich über das Web-Interface von FHEM ereichtbar und ist daher nicht Bestandteil dieser Dokumentation.

4.2.10 Update des Servers

Solange der Server noch nicht als Service in das System eingebunden ist (mittels *sudo make installService*) reicht es, einfach *sudo make install* aufzurufen.

Ist der Service integriert, sollte folgender Make-Aufruf genutzt werden:

sudo make update

Der bewirkt, dass zuerst der Service gestoppt wird, dann die Dateien ausgetauscht werden und zum Schluss der Service wieder gestartet wird.

Falls der Fhem-Client „73_SolvisClient.pm“ upgedatet worden sein sollte, sicherheitshalber Fhem mittels

shutdown restart

neu starten.

Make beachtet die Zeitstempel der Dateien. Daher werden bei einem Update immer nur die neuen Dateien ersetzt. Ich beachte, dass im Installationspaket nur geänderte Dateien einen neuen Zeitstempel erhalten.

Will man das Verhalten umgehen, müsste man mittels des Linux-Befehls *touch <Dateiname>* die Zeitstempel neu setzen.

Zu beachten:

Sollten sich die Grafik-Definitionen im neuen Programmpaket geändert haben, ist statt des Update-Aufrufs der Learn-Aufruf wie folgt auszuführen:

4.2.11 Erneutes Anlernen der Grafik

Sollte jemand die „control.xml“ an seine eigenen Wünschen anpassen oder enthält das Update-Paket ein aktualisiertes „control.xml“ ist immer ein neuer Lern-Vorgang auszuführen.

Dazu ist folgender Make-Aufruf notwendig:

sudo make learn

Dieser Aufruf bewirkt bei einem installierten Server ein Stoppen des Servers, ein Austausch der geänderten Dateien und anschließend wird der Lernvorgang gestartet Am Ende (bei fehlerfreiem Durchlauf) wird dann der Server wieder als Server gestartet.

4.2.12 Deinstallation

Das Programmpaket kann mit folgendem Make-Befehl deinstalliert werden:

sudo make uninstall

Dieser Befehl löscht die vom Makefile angelegten Dateien. Auch der Dienst wird abgemeldet.

4.2.13 Veränderungen am System durch die Installation und durch das Programm

- Die Datei “73_SolvisClient.pm” wird in den Ordner “/opt/fhem/FHEM” kopiert
- Es wird ein Ordner „/opt/SolvisSmartHomeServer“ angelegt
- In diesen werden die Dateien base.xml, base.xsd und SolvisSmartHomeServer.jar kopiert und mit entsprechenden Rechten versehen
- In den Ordner „/etc/systemd/system“ wird die Datei „SolvisSmartHomeServer.service“ sowie „DebugSolvisSmartHomeServer.service“ kopiert
- Durch den Befehl „systemctl enable SolvisSmartHomeServer“ erstellt das System einen Link zur Datei „/etc/systemd/system/SolvisSmartHomeServer.service“ im Ordner „/etc/systemd/system/multi-user.target.wants“

- Durch den Befehl „systemctl enable DebugSolvisSmartHomeServer“ erstellt das System einen Link zur Datei „/etc/systemd/system/DebugSolvisSmartHomeServer.service“ im Ordner „/etc/systemd/system/multi-user.target.wants“

Das Programm selber legt in das Verzeichnis „<*writablePathLinux*>/SolvisServerData folgende Dateien an:

control.xml, control.xsd, graficData.xml, graficData.xsd, log4j2.xml, measurements.xml, measurements.xsd

Sowie die Log-Dateien

solvis.log, solvis.log.*, solvis-error.log

4.2.14 Änderungen des „control.xml“-File

Die „control.xml“-Datei enthält die Definitionen der Screens, der Messwerte, der GUI-Commands etc. Wer sich hier selber heranwagen will:
Sämtliche Attribute und Elemente dieser Datei sind in der XML Schema Definition „control.xsd“ beschrieben und auch kommentiert.

Es empfiehlt sich daher einen XML-Editor zu verwenden, der mit XSD-Datei mit berücksichtigt. Ich verwende den Editor aus Eclipse. Der führt beim editieren der xml-Datei Werteprüfungen durch, macht Attribut-/Element-Vorschläge, „required“-Attribute werden automatisch ergänzt und Kommentare zu den einzelnen Elementen/Attribute werden als ToolTip angezeigt.

Diese Datei wird beim ersten Start und bei jedem Update, bei dem diese Datei verändert wurde zusammen mit der control.xsd-Datei in das „writablePathLinux“ bzw. „writablePathWindows“ geschrieben. Muss bei einem Update diese Datei durch einen neuere ersetzt werden, wird die ältere in „control.xml.1“ umbenannt.

Man muss dann die selber vorgenommenen Änderungen in die neue Datei nachtragen. Wenn es neue Definitionen sind, welche bisher noch nicht existierten, würde ich die auch in die neue Version übernehmen. Daher am besten mir einen PM mit der neuen Definition schicken.

4.3 SolvisSmartHome-Server Windows-Installation

4.3.1 Dateien des Windows-Installationspaketes

Das Installationspaket beinhaltet folgende Dateien:

• SolvisSmartHomeServer setup.exe	Installationsprogramm
• SolvisSmartHomeServer.pdf	Diese Dokumentation
• mqtt-client.0.SolvisSmartHomeServer.json	JSON-Objekt-Liste für IoBroker unter Verwendung des MQTT-Client
• 73_SolvisClient.pm	FHEM-Modul

4.3.2 Aufruf des Installationsprogramms

Durch Aufruf des Programmes *SolvisSmartHomeServer setup.exe* erfolgt die Vorinstallation des Servers. Das Installationsprogramm fragt folgendes ab:

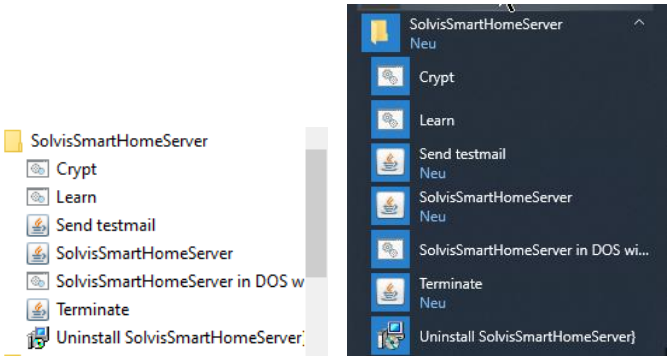
- Das Installationsverzeichnis (Default: C:\JavaPgms\SolvisSmartHomeServer)
- Der Pfad im Startmenü
- Ob der Server mit Windows oder bei Benutzeranmeldung gestartet werden soll

4.3.3 Vom Installationspaket angelegte Dateien

Dabei werden folgende Dateien im dabei auszuwählenden Installationsverzeichnisses angelegt:

- SolvisSmartHomeServer.jar Das eigentliche Programm
- base.xml.new Steuerdatei mit den Basis-Daten
- base.xsd XML Schema hierzu
- CHANGES.txt Datei mit den Änderungen
- Startup.bat Batch-Datei für die Aufrufsvarianten
- Task.xml XML-Datei welche in die Aufgabenplanung importiert wird
- unins000.dat Datei zur Deinstallation
- unins000.exe Deinstallationsprogramm

Neben diesen Dateien werden auch folgende Einträge im Start-Menü erstellt.



Die Einträge im Start-Menü haben folgende Bedeutung:

- Crypt Aufruf zum Verschlüsseln der Passwörter
- Learn Aufruf zum Anlernen der Solvis-Grafik
- Send testmail Sendet eine Testmail
- SolvisSmartHomeServer Manueller Start des SolvisSmartHomeServer
- SolvisSmartHomeServer in DOS window Server im DOS-Fenster starten
- Terminate SolvisSmartHomeServer kontrolliert beenden
- Uninstall SolvisSmartHomeServer Server-Paket deinstallieren

Während des Installationsvorganges wird zusätzlich nachgefragt, ob der Server mit Windows oder beim Anwender-Login gestartet werden soll. Wurde diese Checkbox angeklickt, wird zusätzlich in der Windows-Aufgabenplanung ein Task mit dem Namen *SolvisSmartHomeServerTask* erstellt.

4.3.4 Anpassung der Datei *base.xml*

In dem Installationsverzeichnis liegt folgende Datei:

base.xml.new

Diese Datei ist nach *base.xml* zu kopieren.

Anschließend ist diese Datei wie unter beschrieben anzupassen.

Folgende Unterschiede zwischen Linux und Windows sind zu beachten:

writeablePathWindows: Hier ist ein beschreibbarer Pfad einzutragen

Die Ermittlung der verschlüsselten Passwörter, welche in die base.xml einzutragen sind, erfolgt durch den Aufruf „Crypt“ im Startmenü. Nach dem Aufruf erscheint dann ein DOS-Fenster mit der Aufforderung das zu verschlüsselnde Passwort einzugeben.

Das verschlüsselte Passwort ist dann zu kopieren und in die base.xml einzutragen.

4.3.5 Automatisches Anlernen der Grafiken

Nachdem die *base.xml* angepasst wurde, muss der Server die Grafiken der Solvis-Anlage anlernen. Dazu ist er erst mal mittels des Startmenü-Eintrages *Learn* zu starten.

Dabei öffnet sich ein DOS-Fenster, auf dem man die Server-Ausgaben beobachten kann.

Dieses Anlernen ist etwas kritisch. Man sollte dabei die Fenster-Ausgaben als auch den Solvis-Bildschirm im Auge behalten. Da die SolvisControl leider manchmal Touchs verschluckt, kann der Learn-Vorgang fehlerhaft sein. Ich habe sehr viel Aufwand dazu verwendet, dass dies erkannt wird und in den meisten Fällen wird auch richtig darauf reagiert und der fehlgelaufene Lern-Vorgang wird wiederholt. 100%ig ist es noch nicht, ganz selten muss man den Lernvorgang wiederholen.

Sollte er nicht durchlaufen, bitte nochmal versuchen, wenn kein prinzipielles Problem (z.B. mit der speziellen Anlagenkonfiguration) vorliegt, sollte er dann fehlerfrei durchkommen.

Zu beachten:

Während des Anlernens sollte niemand unter der Dusche stehen, da kurz die Warmwasser-Pumpe ein- /ausgeschaltet wird um die Grafiken der Pumpe zu lernen (ca. 4s). Je nach Kesseltemperatur kann das zu Überraschungen führen.

4.3.6 Starten des Servers im DOS-Fenster

Will man die Meldungen des Servers direkt sehen, kann man den Server auch in der im DOS-Fenster mittels des Start-Menüs *SolvisSmartHomeServer im DOS-Fenster*. Dann werden mögliche Fehlermeldungen direkt angezeigt und man kann das *base.xml*-File entsprechend korrigieren.

Evtl. kommt beim Start auch einen Meldung der Firewall. Der lokale Netztwerkzugriff muss für den SolvisSmartHomeServer erlaubt werden.

4.3.7 Anpassungen der Aufgabenplanung

Nach der Installation und falls die Checkbox selektiert wurde, dass der Server mit Windows oder mit dem Anwender-Login gestartet werden soll, ist in der Aufgabenplanung der Task *SolvisSmartHomeServerTask* automatisch angelegt worden.

Die Start-Optionen können nachträglich über die Aufgabenplanung geändert werden.

Wichtig:

> Der Task muss über die Aufgabenplanung einmal manuell gestartet werden, um die Firewall-Einstellung richtig setzen zu können.

4.3.8 Update des Servers

Um den Server zu aktualisieren, ist es erforderlich ihn zuerst zu beenden. Das erfolgt über das Startmenü *Terminate*. Anschließend reicht es das Installationsprogramm *SolvisSmartHomeServer setup.exe* der neuen Version aufzurufen.

Möglicherweise ist das *base.xml* noch an die neue Version anzupassen.

Es empfiehlt sich den Server über den Startmenüeintrag *SolvisSmartHomeServer im DOS-Fenster* zu starten, damit man mögliche Fehlermeldungen direkt erkennen kann und evtl. die *base.xml* entsprechend zu erweitertn/ändern.

Zu beachten:

Sollten sich die Grafik-Definitionen im neuen Programmpaket geändert haben, ist statt des Update-Aufrufs der Learn-Aufruf - wie unter 4.3.5 beschrieben - erneut anlernen.

4.3.9 Deinstallation

Um den Server zu aktualisieren, ist es erforderlich ihn zuerst zu beenden. Das erfolgt über das Startmenü *Terminate*.

Das Programmpaket kann danach mit den üblichen Windows-Bordmitteln wieder deinstalliert werden. Dabei bleibt das Installationsverzeichnis bestehen, da dort noch die manuell erstellte Datei *base.xml* liegt. Wenn man wirklich auch später den Server nicht mehr benötigt, kann man den kompletten Installationspfas manuell löschen.

4.3.10 Änderungen des „control.xml“-File (normalerweise nicht notwendig)

Die *control.xml*-Datei enthält die Definitionen der Screens, der Messwerte, der GUI-Commands etc. Wer sich hier selber heranwagen will:
Sämtliche Attribute und Elemente dieser Datei sind in der XML Schema Definition „control.xsd“ beschrieben und auch kommentiert.

Es empfiehlt sich daher einen XML-Editor zu verwenden, der mit XSD-Datei mit berücksichtigt. Ich verwende den Editor aus Eclipse. Der führt beim editieren der xml-Datei Werteprüfungen durch, macht Attribut-/Element-Vorschläge, „required“-Attribute werden automatisch ergänzt und Kommentare zu den einzelnen Elementen/Attribute werden als ToolTip angezeigt.

Diese Datei wird beim ersten Start und bei jedem Update, bei dem diese Datei verändert wurde zusammen mit der *control.xsd*-Datei in das *writablePathWindows* geschrieben. Muss bei einem Update diese Datei durch einen neuere ersetzt werden, wird die ältere in „control.xml.1“ umbenannt.

Man muss dann die selber vorgenommenen Änderungen in die neue Datei nachtragen. Wenn es neue Definitionen sind, welche bisher noch nicht existierten, würde ich die auch in die neue Version übernehmen. Daher am besten mir einen PM mit der neuen Definition schicken.

5 Verwendete Schnittstellen der Solvis-Anlage

5.1 Bisherige genutzte Schnittstellen

Die Messwerte der Anlage, welche im Anlagenschema der Web-Seiten 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 auch 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, deren Web-Seite 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 SolvisControl zu, es waren nur die Anlagenmodus Tag/Nacht/Timer/Standby wählbar. Ab und zu erkannte die SolvisControl einer dieser Betätigungen nicht, so dass man es wiederholen musste, für ein zuverlässiges SmartHomeSystem nicht geeignet.

5.2 Neue Schnittstellen

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 und entsprechend darauf automatisch reagiert wird (z. B. durch erneute Betätigung, hilft das nicht, wird das Einstellmenü erneut angefahren).

Der Bildschirminhalt der SolvisControl wird mittels folgenden Http-Zugriffs gelesen:

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

Der Ursprung des Koordinatensystems ist wie bei Bildern üblich oben links. Der Bildschirm, der über diesen Url ausgelesen werden kann, ist halb so groß, wie der, der über das Web-Interface der Solvis-Anlage angezeigt wird. Koordinaten des Programmes (in den XML-Dateien zu finden) basieren immer auf diesem kleineren Bild. Es hat die Größe 240 *128 Punkte.

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 Web-Seite wird das Bild der SolvisControl um den Faktor 2 vergrößerst dargestellt).

6 Interne Komponenten des SolvisSmartHomeServer

Der SolvisSmartHomeServer besteht folgenden Funktionseinheiten

6.1 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.

6.2 MQTT Client

Neben der Server-Client-Schnittstelle kann der „Server“ auch als MQTT-Client in das „IoT“ eingebunden werden. Auf diese Weise ist es möglich ohne einen speziellen Client den Server mit einem SmartHome-System zu betreiben. Voraussetzung für das Smarthome-System ist, dass für diesen ein MQTT-Client/Server zur Verfügung steht.

6.3 Messwerte-Erfassung

Diese 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 mehrfach gewichtet(abhängig von der Abweichung), so dass der vom Modul gelieferte Wert trotz Mittelwert-Bildung dem wirklichen Wert (trotz Mittelwertbildung) bei größeren Änderungen besser folgt (z.B. bei Aufheizung des Kesselwassers durch laufenden Brenner in der höchsten Stufe).

6.4 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 Nummern rechts von der Überschrift erkannt (z.B. 14/5, Heizkreis 1, Bild 4 von 5) Das in Wirklichkeit vorhandene Leerzeichen wird vom OCR nicht beachtet, da es zur Unterscheidung nicht notwendig ist und ohne wirklichen Nutzen ein höhere Aufwand in der OCR-Erkennung notwendig gewesen wäre.

6.5 Weitere Auswertungen des Solvis-Bildschirms

Ändert sich der Bildschirminhalt der Solvis-Anlage ohne dass der Server dies initiiert hat, so wird die erfolgte Änderung in folgender Weise analysiert:

6.5.1 Bildschirmschoner

Es wird untersucht, ob die Änderung durch den Bildschirmschoner erfolgt ist. Den Bildschirmschoner erkennt der Server anhand der relativen Lage der Zeit- und Datumsfelder.

Ein erkannter Bildschirmschoner bewirkt keine weiteren Aktionen.

6.5.2 Meldungs-Box-Erkennung

Der Bildschirm wird in den Ruhephasen auch auf eine Meldungs-Box untersucht. Dazu werden die Lage der Umrandung der Box sowie der Strich unter der Überschrift der Meldung analysiert. Hier ein Beispiel einer solchen Message-Box:

Wird eine Meldung erkannt, so wird der Status der Anlage im Server auf den Status „ERROR. Ist die ExceptionMail in der base.xml eingerichtet, wird zusätzlich eine Hardcopy des Bildschirminhalts an die in base.xml eingetragenen Mail-Empfänger versendet.

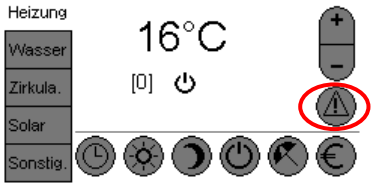
Anschließend gibt es noch zwei unterschiedliche Behandlungen, je nachdem das Feature „ClearErrorMessageAfterMail“ im base.xml gesetzt ist.

- Im Falle von ClearErrorMessageAfterMail = “false”, bleibt die Meldungsbox bestehen, die GUI-Steuerung ist dadurch blockiert!
- Im Falle von ClearErrorMessageAfterMail = “true”, wird nach erfolgreichem Mail-Versand der Hardcopy mit dem < -Button zum HomeScreen zurückgegangen. Die GUI-Steuerung ist nicht blockiert.

Der Status ERROR des Servers bleibt auch nach dem Verschwinden der Meldungsbox noch bestehen. Es wird automatisch in den Homescreen gewechselt und der dortige Button mit dem Warndreieck untersucht. Erst wenn dieses Warndreieck verschwunden ist, wird der ERROR-Status gelöscht (siehe auch 6.5.4).

6.5.3 Error-Button-Erkennung auf dem HomeScreen

Im Fehlerfall der Anlage wird anstelle der Datums/Uhranzeige ein Button mit einem Warndreieck eingeblendet. Hier ein Beispiel:



In Wirklichkeit erkennt der Server nicht den Button (das würde ein Provozieren eines Fehlers in der Lern-Phase erfordern) sondern die fehlende Einblendung von Zeit und Datum.

Dieser Button ist quasi der Summen-Button für alle gemeldeten Fehler der Anlage. Ist noch ein Fehler aktiv, dessen Meldung schon weggeklickt wurde, dann wird das durch diesen Button kenntlich gemacht.

Erkennt der Server diesen Button, geht der Status des Servers auf ERROR (falls er nicht schon ist). War vorher der ERROR-Status noch nicht gesetzt, wird auch einen Mail mit der Hardcopy verschickt (falls die Exception-Mail eingerichtet ist).

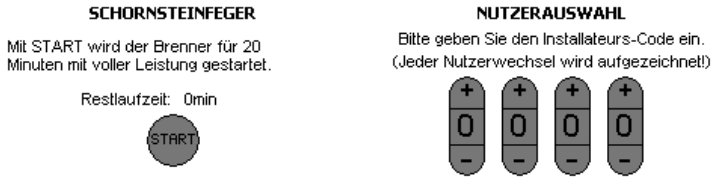
Ist der ERROR-Status gesetzt werden GUI-Befehle immer ausgeführt. Nach der Ausführung wird immer wieder zum Home-Screen zurück gegangen und es wird weiter der Error-Button beobachtet. Erst wenn dieser verschwindet wird der ERROR-Status des Servers zurück gesetzt und es wird zur letzten vom User ausgewählte Seite gegangen.

6.5.4 Anwender-/Service-Erkennung

Erfolgte eine Änderung des Bildschirms ohne vom Server veranlasst ist und es werden weder der Bildschirmschoner noch der beiden Fehler-Darstellungen erkannt, so geht der Server davon aus, dass der Anwender in die Anlage eingegriffen hat.

Er sperrt dann für eine bestimmte Zeit (in base.xml definiert) die GUI-Command-Ausführung.

Werden eins der folgenden Bildschirme erkannt, dann geht der Server von einem Eingriff eines Service-Mitarbeiter aus:



In diesem Fall wird ebenfalls die GUI-Steuerung gesperrt, dies jedoch deutlich länger (default 2h, in der base.xml konfigurierbar). Auf diese Weise wird verhindert, dass ein Service-Mitarbeiter durch den Eingriff des Servers irritiert wird und von einem falschen Fehler ausgeht.

6.6 Logging

Zur Fehlersuche werden wesentliche Infos und Fehlermeldung in Log-Dateien geschrieben. Diese werden in den im base.xml definierten beschreibbaren Pfad geschrieben. Seit der Version 1.00.11 ist die Namenskonvention dieser Log-Dateien folgende:

`solvis-tiny.log.*`	In diese Datei werden sämtliche Logmeldungen geschrieben
`solvis-error.log.*`	In diese Datei werden nur die Fehlermeldungen geschrieben

Die Unterscheidung erfolgt, weil normalerweise in die normale Log-Datei deutlich mehr geschrieben wird, als in die Error-Log-Datei. Da die Log-Dateien begrenzt sind (5MByte für die Standard-, 2 MByte für die Error-Datei), kann es sein, dass wichtige Fehlermeldungen in dieser Datei schon nicht mehr vorhanden sind. Wenn man den Log-Level auf Debug setzt, kann das – je nach Konfiguration - schon innerhalb eines Tages erreicht sein.

7 Ablauf des Programms

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.

Anwerkung:

Die Phasen 1 und 2 werden im Learning-Mode ausgeführt. Dazu ist der Server mit den Parameter -- *server-learn* aufzurufen. Ist der Lernvorgang (Phase 1 und 2) beendet, wird das Programm beendet.

Die Phase 3 und 4 werden dann ausgeführt, wenn der Server normal gestartet wird. Das kann auch im Hintergrund erfolgen.

7.1 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 **Screens** 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.

Dieser Teil erzeugt auf der Console folgende Einträge (Stand 26.02.2020, bei einem Heizkreis):

```
2020-02-26 11:51:45,182|LEARN|Learning started.
2020-02-26 11:51:57,351|LEARN|Screen grafic <Home> learned.
2020-02-26 11:52:00,954|LEARN|Screen grafic <Solar> learned.
2020-02-26 11:52:04,141|LEARN|Configuration mask: 0x1000001
2020-02-26 11:52:12,278|LEARN|Screen grafic <Nachttemperatur> learned.
2020-02-26 11:52:12,324|LEARN|Screen grafic <NachttemperaturNotSelected> learned.
2020-02-26 11:52:15,984|LEARN|Screen grafic <NachttemperaturSelected> learned.
2020-02-26 11:52:22,706|LEARN|Screen grafic <Warmwasser> learned.
2020-02-26 11:52:34,070|LEARN|Screen grafic <Tagestemperatur> learned.
2020-02-26 11:52:34,114|LEARN|Screen grafic <TagestemperaturNotSelected> learned.
2020-02-26 11:52:37,755|LEARN|Screen grafic <TagestemperaturSelected> learned.
2020-02-26 11:52:44,474|LEARN|Screen grafic <Zirkulation> learned.
2020-02-26 11:52:51,250|LEARN|Screen grafic <Sonstiges> learned.
2020-02-26 11:52:51,294|LEARN|Screen grafic <Sonstiges 1> learned.
2020-02-26 11:52:54,960|LEARN|Screen grafic <Heizkreise> learned.
2020-02-26 11:52:58,701|LEARN|Screen grafic <Heizkreis> learned.
2020-02-26 11:53:08,389|LEARN|Screen grafic <Anlagenstatus WW> learned.
2020-02-26 11:53:12,062|LEARN|Screen grafic <Anlagenstatus HK> learned.
2020-02-26 11:53:14,031|LEARN|Screen grafic <Anlagenstatus Solar> learned.
2020-02-26 11:53:22,364|LEARN|Screen grafic <Schornsteinfeger> learned.
2020-02-26 11:53:29,106|LEARN|Screen grafic <Sonstiges 2> learned.
2020-02-26 11:53:32,730|LEARN|Screen grafic <Sonstiges_Nutzerauswahl> learned.
2020-02-26 11:53:36,282|LEARN|Screen grafic <Nutzerauswahl> learned.
2020-02-26 11:53:46,083|LEARN|Screen grafic <Zaehlfunktion> learned.
2020-02-26 11:53:52,750|LEARN|Screen grafic <Sonstiges 3> learned.
2020-02-26 11:53:56,420|LEARN|Screen grafic <Uhrzeit / Datum> learned.
2020-02-26 11:54:00,080|LEARN|Screen grafic <Zeiteinstellung> learned.
```

Wichtig:

Vor dem Start des Programms muss in der SolvisControl unter „Sonstig./Anlagenstatus“ der Bildschirm mit dem Warmwasser-Kreis ausgewählt worden sein.

7.2 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 wieder auf den ursprünglichen zurück, kurzzeitig befindet sich die Anlage jedoch in einem anderen Zustand, das sollte beachtet werden um entsprechend vorher oder nachher (wenn in dieser einzugreifen.

Dieser Teil erzeugt auf der Console folgende Einträge (Stand 26.02.2020):

```
2020-02-26 11:54:33,701|LEARN|Screen grafic <Zeiteinstellung_YYYY> learned.
2020-02-26 11:54:40,844|LEARN|Screen grafic <Zeiteinstellung_MM> learned.
2020-02-26 11:54:47,917|LEARN|Screen grafic <Zeiteinstellung_DD> learned.
2020-02-26 11:54:54,867|LEARN|Screen grafic <Zeiteinstellung_hh> learned.
2020-02-26 11:55:02,051|LEARN|Screen grafic <Zeiteinstellung_mm> learned.
2020-02-26 11:55:17,628|LEARN|Screen grafic <ModeTag> learned.
2020-02-26 11:55:20,223|LEARN|Screen grafic <ModeNacht> learned.
2020-02-26 11:55:22,748|LEARN|Screen grafic <ModeStandby> learned.
2020-02-26 11:55:25,350|LEARN|Screen grafic <ModeTimer> learned.
2020-02-26 11:55:34,115|LEARN|Screen grafic <WWPumpeAus> learned.
2020-02-26 11:55:36,629|LEARN|Screen grafic <WWPumpeAn> learned.
2020-02-26 11:55:39,272|LEARN|Screen grafic <WWPumpeAuto> learned.
2020-02-26 11:55:42,445|LEARN|Learning finished.
```

Anmerkung: Die Bildschirm-Ausschnitte der Uhreinstellung werden auch in dieser Phase erkannt. Das hat Programm-interne Gründe.

7.3 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.

7.4 Phase 4

In dieser Phase sind alle Anlagenparameter ausgelesen und die Messauswertung erfolgt zyklisch. Anlageparameter werden immer erst auf Anforderung gelesen/verändert. In dieser Phase wird auch analysiert, ob der Screen-Saver aktiv ist, ob ein Zugriff durch den Anwender selber erfolgt ist oder der Fehlerbildschirm angezeigt wird. Diese können entsprechende Events dann im SmartHome-System auslösen (außer Screen-Saver).

Wurde ein Eingriff durch ein Anwender direkt an der SolvisControl (oder über SolvisRemote) erkannt, werden alle Anlagenparameter erneut gelesen, wenn der Anwenderzugriff beendet ist.

7.5 Besonderheiten

Es gibt einige berechnete Werte, welche genauere Werte liefern, als die auf der SolvisControl angezeigten. Dazu gehören die Brennerlaufzeiten. Diese müssen regelmäßig mit den Werten abgeglichen werden, welche von der SolvisControl angezeigt werden, da diese andernfalls auseinanderlaufen. Dazu erfolgt entsprechenden dem Messwert eine 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, dauert dies solange, bis sich der Wert in der Solvis-Anzeige ändert.

Wird der Server das erste Mal genutzt, dauert diese Synchronisationsphase länger und bewirkt, dass solange der Brenner läuft die Zählfunktion-Anzeige auf dem Display der SolvisControl erscheint.

Da die Uhr der Solvis-Anlage nicht besonders genau ist (meine geht mehr als 1 Minute/Woche vor), wird diese vom Server ständig beobachtet. Weicht sie mehr als 30s von der Uhr des Systems ab, auf dem der Server läuft, wird sie automatisch korrigiert.

Es gibt auch noch die Möglichkeit die Uhr genauer zu trimmen, indem man mehrfach Dummy-Einstellungen der Uhr durchführt. Diese bewirken ein kurzzeitiges Stehenbleiben der Uhr, so dass damit noch genauer die Uhr gestellt werden kann. Wer will kann diesen Modus über die Datei „base.xml“ aktivieren.

Der Bildschirm der SolvisControl wird etwa alle 30s untersucht, on ein Anwenderzugriff erfolgt ist.

8 Anbindung an verschiedene SmartHome-Systeme

8.1 FHEM-Anbindung

Für das Smarthome-System FHEM existiert ein spezielles Modul, das die Client-Server-Schnittstelle nutzt. Es ist im Programmpaket enthalten und sollte in das FHEM-Verzeichnis kopiert werden.

Wenn das FHEM auf einem Linux-Rechner installiert ist, kann es mittels Make - wie unter 4 beschrieben – in das System eingebunden werden.

8.2 IOBroker-Anbindung

Für IOBroker existiert (bisher) kein spezieller Adapter. Zu Anbindung kann der Adapter „MQTT Broker/Client „ oder „MQTT-Client“ verwendet werden.

8.2.1 Verwendung vom MQTT Broker/Client

Leider habe ich mit der aktuellen Version 2.1.2 des MQTT Broker/Clients keinen stabilen Betrieb. Dieses Problem scheinen auch andere zu besitzen:

https://forum.iobroker.net/topic/32143/mqtt-problem

Wenn der „stable“-Version des MQTT Broker wieder stabil ist, werde ich mich damit beschäftigen. Es kann natürlich daher kommen, dass ich IoBroker auf einer virtuellen Maschine laufen lasse und der Server auf dem Entwicklungs-PC läuft. Da ich selber als MQTT Broker den Mosqitto schon jahrelang verwende, hat das aktuell eine niedrige Priorität.

8.2.2 Verwendung vom MQTT Client

Dazu muss ein MQTT Broker im System existieren. Ich empfehle den Mosquitto. Mit diesem läuft mein SmartHome-System schon seit Jahren problemlos. Den MQTT-Test des SolvisSmartHomeServers erfolgte ebenfalls mit diesem Broker.

Der MQTT-Client ist wie folgt einzurichten:

Einstellung	Wert	Bedeutung
MQTT Broker IP	192.168.0.71 (Beispiel)	IP-Adresse des MQTT Brokers, z.B. Mosquitto (Server)
Client ID	IoBrokerSolvisClient (Beispiel)	Im MQTT-System eindeutige ClientId
Topic bei Verbindung	*Topic*/*Client*/online	Topic, der bei Verbindung gesendet wird
Meldung bei Verbindung	true	Meldung, welche bei Verbindung gesendet wird
last will topic	*Topic*/*Client*/online	Topic, der bei Verbindungsverlust gelten soll
last will message	false	Meldung, welche bei Verbindungsverlust gilt
~~***Zusätzliche subscriptions***~~	~~*Topic/#*~~	~~Subscription, welche automatisch geholt werden, nur erforderlich, wenn man die Objektliste selber erstellen~~

~~will.~~

Hierbei ist

Topic: der Wert von „topicPrefix“ der in base.xml zutragen ist

Client: Ein vom User festzulegender Name, über den der IoBroker MQTT Client beim SolvisSmartHomeServer identifiziert wird (z.B. IoBroker)

Im Installationspaket befindet sich die Datei mqtt.0.SolvisSmartHomeServer.json. Diese Datei kann man den eigenen Wünschen anpassen und anschließend in die Objekt-Liste des IoBrokers importieren.

Die Beschreibung der MQTT-Topics findet sich im Anhang D.

Die Objekt-Datei mqtt.0.SolvisSmartHomeServer.json basiert auf folgenden Werten:

prefix	SolvisSmartHomeServer
client	IoBroker
unit	mySolvis

Zu jedem Kanal gibt es folgende Objekte:

data	Daten des Kanals vom letzten Auslesen
meta	Die Beschreibung des Kanals
cmnd	Objekt über das man beschreibbare Werte ändern kann

Das Objekt „cmnd“ existiert nur für beschreibbare Känale, wie Solltemperaturen, Pumpensteuerung etc.

Anhang A Kommando-Zeilen-Parameter

--server-terminate	Beendet den laufenden Server-Prozess so, dass noch ein Backup-File der aktuellen Messwerte geschrieben wird
--server-learn	Lernt die Grafiken an (nur wenn notwendig)
--server-restart	Für den Restart des Servers notwendig, damit agentlib-Parameter nach dem Restart möglich sind (Remote-Debug). Dient nur der internen Verwendung, sollte man nicht von der Kommandozeile aus nutzen.
--string-to-crypt	Ermittelt aus einem Wort/Phrase einen AES-256 verschlüsselten Wert. Syntax ist folgende: --string-to-crypt=*Word*
--test-mail	Sendet eine Test-Mail an die im base.xml angegeben Mailadressen
--create-task-xml	Erstellt während der Windows-Installation die notwendige Task.xml. Ist eine Quick-And-Dirty-Lösung, da ich nicht geschafft habe, mittels InnoSetup eine entsprechende Datei zu generieren (UTF-16LE mit Byte Order Mark). Andere Formate, welche die Aufgabenverwaltung annimmt, habe ich nicht gefunden. Warum Windows dort so restriktiv ist, ist mir ein Rätsel.

Anhang B Schnittstelle Server – Client

Die Kommunikation zwischen Server und Client basiert auf JSON.

A.1 **Die Schnittstelle basiert aktuell auf folgende Strukturen, welche im JSON-Format übertragen werden:**

CONNECT	Verbindungsaufbau	Client -> Server
RECONNECT	Wiederaufbau der Verbindung	Client -> Server
CONNECTED	Quittierung für den Aufbau der Verbindung	Server -> Client
DISCONNECT	Verbindungsabbau (von FHEM nicht verwendet)	Client -> Server
CONNECTION_STATE	Status der Verbindung (ALIVE etc.)	Server -> Client
SOLVIS_STATE	Status der Solvis-Anlage (PowerOff etc.)	Server -> Client
DESCRIPTIONS	Meta-Daten der Kanäle/Server Commands	Server -> Client
MEASUREMENTS	Paket mit den Messwerten	Server -> Client
SET	Ändert einen Anlagenparameter	Client -> Server
GET	Stößt das Auslesen eines Anlagenparameters an	Client -> Server
SERVER_COMMAND	Befehl für den Server (z.B. Backup der Messdaten)	Client -> Server

Der erste Verbindungsaufbau erfolgt wie folgt:

Client -> Server	CONNECT mit Namen der Solvis-Anlage
Server -> Client	CONNECTED mit Client-ID
Server -> Client	CHANNEL_DESCRIPTIONS
Server -> Client	MEASUREMENTS
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).

Ein evtl. unterbrochene Verbindung wird wie folgt wieder aufgebaut:

Client -> Server	RECONNECT mit Client-ID

Server -> Client	MEASUREMENTS
Server -> Client	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, bei dem Client-spezifische Server-Einstellungen verloren gehen. Es empfiehlt sich daher die Verbindung wie folgt wieder herzustellen:

Wiederaufbau einer Verbindung:

Client -> Server	RECONNECT mit Client-ID
Server -> Client	MEASUREMENTS
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“.

War der Wieraufbau der Verbindung erfolgreich, werden wieder GET/SET/MEASUREMENTS/SOLVIS_STATE-Pakete ausgetauscht.

A.2 JSON-Elemente

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 der JSON-Daten erkennt, wurde die JSON-Daten in einen Frame verpackt.

Dieser Frame sie wie folgt aus:

Übertragungs-Frame:

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

Folgende JSON-Datensätze sind definiert:

A.2.1 CONNECT

Der Connect-Datensatz wird vom Client versandt und initiiert den Verbindungsaufbau. Er ist wie folgt aufgebaut:

```
{"CONNECT":{"Id":"Name der Anlage"}}
```

Id ist die Id der Unit, welche im *base.xml*-File definiert ist.

A.2.2 CONNECTED

Mit dem CONNECTED-Datensatz teilt der Server dem Client den Erfolg der Verbindung mit (Server existiert und Unit-Id ist im base.xml eingetragen). Er ist wie folgt aufgebaut:

```
{"CONNECTED":{"ClientId":427735588,"ServerVersion":"00.01.00","FormatVersion":"01.02"
}}
```

Dire Client-Id dient der künftigen Identifikation des Clients, falls die Übertragung zwischen Server und Client unterbrochen und wieder neu aufgebaut wird. Neben der Client-Id werden die aktuelle Server-Version sowie die Format-Version der JSON-Übertragung mitgeliefert. Wenn das JSON-Übertragungsformat in künftigen Versionen nicht mehr kompatibel ist, wird diese angepasst. Damit kann erkannt werden, dass der Client veraltet ist.

Nach dem der Client CONNECTED empfangen hat, ist der Server für Befehle (GET, SET, SERVER-Commands) des Clients empfangsbereit.

Im Falle einer Verbindungs-Unterbrechung versucht der Client mittels RECONNECT diese wieder aufzubauen:

A.2.3 RECONNECT

Ist die Verbindung unterbrochen worden, kann der Client die Verbindung zum Server durch senden des RECONNECT-Datensatzes wieder aufbauen. Er ist wie folgt aufgebaut:

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

Die Client-Id ist eine 32-Bit-Integer-Zahl (vorzeichenbehaftet), welche der Server im Datenpaket „CONNECTED“ geliefert hatte. Eine Unterscheidung zwischen CONNECT und RECONNECT ist notwendig, da der Client bestimmte Einstellungen im Server durchführen kann (z.B. SCREEN_RESTORE_INHIBIT). Diese Einstellungen sind Client-bezogen, bei einem CONNECT würden diese verloren gehen. Ist die Verbindung zu lange unterbrochen, werden diese Client-abhängige Einstellungen zurück gesetzt und der Client erhält bei einem Verbindungsversuch über RECONNECT eine Abweisung. Er muss sich dann mittels CONNECT erneut verbinden, die vorherigen Einstellungen sind dann zurück gesetzt. Der Timeout ist im control.xml-File definiert und ist aktuell auf 5 Minuten gesetzt (connectionHoldTime_ms).

A.2.4 DISCONNECT

Derr Client kann die Verbindung durch senden des DISCONNECT-Pakets abbrechen. Es ist wie folgt aufgebaut:

```
{"DISCONNECT": null }
```

Mit dieser Anweisung wird die Verbindung zwischen Server und Client endgültig getrennt. Die Client-spezifischen Einstellungen sowie die Client-Id auf der Server-Seite werden gelöscht.

A.2.5 CONNECTION_STATE

Dieses Paket sendet der Server an den Client. Es ist wie folgt aufgebaut:

```
{"CONNECTION_STATE": {"State": "state"}, {"Message": "message" }
```

state kann folgende Werte annehmen:

- CLIENT_UNKNOW — Reconnect mit unbekannterClient-Id wurde versucht.
- CONNECTION_NOT_POSSIBLE — Verbindung zur Solvis-Anlage nicht möglich.
- COMMAND_ERROR — Vom Client wurde ein unbekannter Befehl gesendet.
- ALIVE — Zur Alive-Überwachung, sendet der Server, wenn er 2 min lang keine Daten gesendet hat.
- USER_ACCESS_DETECTED — Eine Anwender-Bedienung der Solvis-Anlage erkannt
- SERVICE_ACCESS_DETECTED — Eine Wartungs-Bedienung der Solvis-Anlage erkannt
- HUMAN_ACCESS_FINISHED — Manuelle Bedienung der Solvis-Anlage beendet

message ist optional und liefert zusätzliche Information für den Log.

A.2.6 DESCRIPTIONS

Nach dem Aufbau der Verbindung über CONNECT liefert der Server das Descriptions-Paket. Es enthält die Meta-Daten der Kanäle und der möglichen Server-Commands. Dieses Paket besitzt folgenden Aufbau:

```
{"DESCRIPTIONS": {
        "Name 1": {
                "Writeable":   boolean,
                "Type":        "type",
                "Unit":        "unit",
                "Accuracy":    float,
                "IsBoolean":   boolean
```

```
                "Upper":        float
                "Lower":        float
                "Step":         float
                "Modes":        ["mode1","mode2", …, "moden"]
                },
        "Name 2":……
        }
}
```

DESCRIPTIONS enthält für jeden Kanal/Server-Command eine Eigenschaft, wobei der Schlüssel der *Name* des Kanals bzw. der Server-Befehl ist. Der Wert ist dann ein weiteres JSON-Objekt in dem die Eigenschaften des Kanals bzw. Server-Befehls beschrieben sind.

Es existieren folgende Schlüssel:

- Writeable true/false, gibt an, ob der Kanal beschreibbar ist
- Type Typ
- Unit Einheit des Messwertes
- Accuracy Genauigkeit des Messwertes, als float
- IsBoolean Wenn Kanal einen boolscher Wert darstellt

Zur Beschreibung einer Wertemenge von Zahlen dienen folgende Schlüssel:

- Upper Höchster Wert
- Lower Niedrigster Wert
- Step Differenz zwischen den Einzelwerten

Bei einer Beschreibung eines Modes (z.B. WW-Pumpe, Anlagenmodus) gibt es dann noch folgenden Schlüssel:

- Modes Array mit den verschieden Mode-Werten

Die meisten Schlüssel sind optional, nur die Schlüssel „Writeable“ und „Type“ müssen immer vorhanden sein. Nur wenn der Wert beschreibbar ist, existiert die Beschreibung der Wertemenge.

Folgende Typen sind aktuell definiert:

- MEASUREMENT Messwerte werden über die XML-Schnittstelle ausgelesen
- CALCULATION Werte sind berechnet (z.B. Brennerlaufzeit)
- CONTROL Werte werden über das GUI bestimmt, meist veränderbar
- ServerCommand *Name* ist ein Server-Befehl

A.2.7 SOLVIS_STATE

Dieses Paket sendet der Server an den Client und wird bei jeder Status-Änderung der Anlage versendet. Das Paket hat folgenden Aufbau:

```
{"SOLVIS_STATE": {"SolvisState": "state"}, {"Message": "message" }
```

state kann folgende Werte annehmen:

- `POWER_OFF` Die Anlage ist ausgeschaltet (länger als 2 Minuten keine Verbindung möglich)
- `REMOTE_CONNECTED` Nur Verbindung zur Solvis-Remote aufgebaut
- `SOLVIS_CONNECTED` Verbindung zur Anlage über die Solvis-Remote aufgebaut
- `SOLVIS_DISCONNECTED` Keine Verbindung zur Solvis-Remote und Anlage
- `ERROR` Anlage befindet sich im Fehlerzustand (Fehlermeldung auf dem Bildschirm oder „A14.Entstoerung“ true)

DESCRIPTIONS	Meta-Daten der Kanäle/Server Commands	Server -> Client

MEASUREMENTS	Paket mit den Messwerten	Server -> Client
SET	Ändert einen Anlagenparameter	Client -> Server
GET	Stößt das Auslesen eines Anlagenparameters an	Client -> Server
SERVER_COMMAND	Befehl für den Server (z.B. Backup der Messdaten)	Client -> Server

A.2.8 MEASUREMENTS

Die gemessenen Werte sendet der Server bei Veränderungen an den Client in Form des MEASUREMENTS-Pakets. Es besitzt folgenden Aufbau:

```
{"MEASUREMENTS": {
        "Kanalname 1", boolean/float/"String"
        "Kanalname 2", boolean/float/"String"
        ……
        "Kanalname n", boolean/float/"String"
        }
}
```

A.2.9 SET

Zum Verändern der Anlagenwerte (Type CONTROL) sendet der Client an den Server das SET-Package. Es hat folgenden Aufbau:

```
{"SET":{"Kanalname": boolean/float/"String"}}
```

A.2.10 GET

CONTROL-Werte werden nicht periodisch abgefragt, da immer ein Eingriff auf das GUI der Solvis-Anlage notwendig ist. Um die neuen Werte ermitteln zu können, kann der GUI-Auslesevorgang mit dem GET-Paket veranlasst werden. Dazu sendet der Client an den Server dieses Paket. Es hat folgenden Aufbau:

```
{"GET":{"Kanalname":null}}
```

Anhang C XML-Files

Folgende XML-Dateien steuern den Server, zu jeder dieser Dateien existiert eine entsprechendes XML-Shema:

control.xml	Enthält die wesentlichen Informationen. Dort sind die Daten enthalten, welche Bildschirmbereiche zur Identifikation der Bildschirme herangezogen werden müssen, welche Touch-Points unter welchen Umständen gedrückt werden müssen, die Auswertungsinformation für die Daten des Hex-Strings sowie weitere programmbezogene Daten.
graficData.xml	Eine vom Programm generierte Datei mit den zur Identifikation der Bildschirme notwendigen Bildausschnitten. Diese Datei wird beim Learning generiert.
log4j2.xml	Datei zur Steuerung des Loggings
measurements.xml	Datei mit den Messwerten X*.*. Diese Messwerte werden bei einem Beenden des Programms sowie stündlich (veränderbar im control.xml) gespeichert.

Bis auf die Datei „log4j2.xml“ existiert für jede der obigen Datei ein XML-Shema. In dieser Datei sind die einzelnen XML-Elemente dokumentiert. Auf eine weitere Dokumentation wird hier daher verzichtet. Wenn jemand die control.xml verändern will, sollte er einen XML-Editor verwenden, der auch mit den XSD-Dateien umgehen kann (z.B. der XML-Editor des Eclipse-Pakets). Auf diese Weise werden die Fehlermöglichkeiten reduziert. In dem Editor kann man sich dann in der Regel zu jedem Attribut/Element auch die dazugehörige Dokumentation ansehen (in Eclipse durch Tool-Tips).

C.1 Die Properties-Datei „tinylog.properties“

Seit Version V01.00.11 zum Loggen nicht mehr Log4J verwendet sonder TinyLog. Dadurch ist das Programmpaket um 2 MByte kleiner geworden (nun 1,4 MByte). Der Fileinhalt sieht wie folgt aus:

```
writer1             = console
writer1.level       = error
writer1.format      = {date: HH:mm:ss,SSS}|{level}|{message}
writer1.tag         = -

writer2             = rolling file
writer2.level       = info
writer2.file        = solvis-tiny.log.{count}
writer2.format      = [{date: yyyy-MM-dd HH:mm:ss,SSS}] {level} {message}
writer2.append      = true
writer2.policies    = size: 1mb
writer2.backups     = 5

writer3             = console
writer3.level       = info
writer3.format      = {date: HH:mm:ss,SSS}|LEARN|{message}
writer3.tag         = LEARN

writer4             = rolling file
writer4.level       = error
writer4.file        = solvis-error.log.{count}
writer4.format      = [{date: yyyy-MM-dd HH:mm:ss,SSS}] {level} {message}
writer4.append      = true
writer4.policies    = size: 1mb
writer4.backups     = 2

writingthread       = false
autoshutdown        = false
```

Wenn mehr geloggt werden soll, ist der Wert des Properties „writer2.level“ von „info“ auf „debug“ zu stellen. Dies ist zur Fehleranalyse u.U. notwendig.

C.2 Die XML-Datei „log4j2.xml“

Bis zur Version V01.00.10 genutzt.

Der Server nutzt log4j2 von der Apache Software Foundation. Dieses System bietet eine Vielzahl von Möglichkeiten, so ermöglichst es auch das Logging auch in einer Datenbank.

Für den Anwender ist der letzte Abschnitt dieser Datei interessant:

```
<Async name="Async">
        <appender-ref ref="Solvis" level="info" />
        <appender-ref ref="Console" level="LEARN"/>
        <appender-ref ref="Solvis-Error" level="error"/>
        <!-- appender-ref ref="RFC5424" level="info"/ --> <!-- Database logging -->
</Async>
```

Der erste Appender (Solvis) schreibt in die Datei „solvis.log“. In der obigen Einstellungen werden alle Levels ab „info“ dort reingeschrieben. Zur Fehlersuche ist es sinnvoll „info“ durch „debug“ zu ersetzen. Dann liefert der Log noch mehr Informationen.

Der zweite Appender (Console) bestimmt, was auf die Konsole geschrieben wird. Mit der obigen Definition werden nur die Ausgaben des Lern-Vorgangs sowie Fehlermeldungen ausgegeben. Läuft der Server als Service, kann man die letzten Meldungen mittels „sudo systemctl status“ ausgeben lassen.

Der dritte Appender (Solvis-Error) bewirkt, dass Fehlermeldungen in eine Extra-Datei mit dem Namen „solvis-error.log“ geschrieben werden.

Der letzte auskommentierte Appender (RFC5424) wäre für ein Datenbank-Logging notwendig. Der ist aktuell nicht aktiv und die in der log4j2.xml dazu notwendigen Teile habe ich einfach die verwendet, die ich auch in der Arbeit verwende. Das erfordert noch weitere Infrastruktur, daher ist das erst mal auskommentiert.

Anhang D Die MQTT-Schnittstelle

D.1 Allgemeines

Der SolvisSmartHomeServer unterstützt neben seiner proprietären Server-Client-Schnittstelle noch das MQTT-Protokoll. Aktuell jedoch nur die nichtverschlüsselte Version. Sollte bzgl. einer SSL-Verschlüsselung Bedarf bestehen, bitte eine PM an mich: Stefan.Gollmer57@gmail.com

Bei MQTT kennen sich die Clients untereinander nicht. Für den Server (ist ebenfalls ein MQTT-Client, wird aber weiterhin als Server benannt) ist dies jedoch für bestimmte Befehle notwendig. So muss beispielsweise der Server Informationen an den Client liefern können, wenn der Client selber bei Publish einen Befehl fehlerhaft angewendet hat. Daher muss der Client eine im System eindeutige Id besitzen.

Alle Publish-Befehle des Client müssen daher die Id des Clients im Topic besitzen. Der Aufbau der Topics, den die Clients versetzen muss immer folgenden Aufbau besitzen:

prefix/*client*/……

Hierbei ist *prefix* in base.xml unter dem Attribut *topicPrefix* definiert.
client ist die eindeutige ID des Clients

Topics, welche der Server versendet sind immer im folgenden Format:

prefix/……

Hierbei ist *prefix* in base.xml unter dem Attribut *topicPrefix* definiert.

Eine Ausnahme hierzu ist die Rückmeldung des Servers bei einem Fehler auf Grund eines Publish eines Clients. In diesem fall wird die Fehlermeldung in folgendes Topic geschrieben:

prefix/*client*/error

D.2 Present Topics

Alle Topics, welche der Server schreibt (publish) sind vom Typ „present“ mit Ausnahme von „*prefix*/*client* /error“.

D.3 Serverspezifische Topics

Folgende rein Server-spezifischen Topics existieren:

Topic	Richtung (Server-bezogen)	Bedeutung / Werte
***prefix*/server/meta**	publish	JSON-Meta-Beschreibung der Befehle
***prefix*/server/online**	publish	true/false
***prefix*/*client*/server/cmnd**	subscribe	Befehle: backup, restart
***prefix*/*client* /error**	publish	Fehlermeldung für den Client *client*
***prefix*/*client*/online**	subscribe	Status: true/false (Client online). Der LWT (last will) des Client sollte diesen Wert auf false setzen.

Die JSON-Beschreibung der Befehle sieht folgendermaßen aus:

["backup",“restart“]

D.4 Unit- & Serverspezifische Topics

Folgende Topics sind Anlagen-abhängige den Server-Einstellungen/-Status:

Topic	Richtung (Server-bezogen)	Bedeutung / Werte

***prefix*/*unit*/server/meta**	publish	JSON-Meta-Beschreibung der Befehle
***prefix*/*client*/*unit*/server/cmnd**	subscribe	Befehle: SCREEN_RESTORE_INHIBIT, SCREEN_RESTORE_ENABLE, COMMAND_OPTIMIZATION_INHIBIT, COMMAND_OPTIMIZATION_ENABLE, GUI_COMMANDS_ENABLE, GUI_COMMANDS_DISABLE, SERVICE_RESET, UPDATE_CHANNELS

Die JSON-Beschreibung der Befehle sieht folgendermaßen aus:

["SCREEN_RESTORE_INHIBIT“,“SCREEN_RESTORE_ENABLE“,“COMMAND_OPTIMIZATION_INHIBIT“,“ COMMAND_OPTIMIZATION_ENABLE“,“GUI_COMMANDS_ENABLE“, “GUI_COMMANDS_DISABLE“,“SERVICE_RESET“,“UPDATE_CHANNELS“]

D.5 Unit-spezifische Topics

Folgende Unit-spezifische Topics existieren:

Topic	Richtung (Server-bezogen)	Bedeutung / Werte
***prefix*/*unit*/status**	publish	Status der Unit (POWER_OFF, REMOTE_CONNECTED, SOLVIS_CONNECTED, SOLVIS_DISCONNECTED, ERROR)
***prefix*/*unit*/human_access**	publish	none, user, service
***prefix*/*unit*/*channel*/meta**	publish	JSON-Meta-Beschreibung des Kanals
***prefix*/*unit*/*channel*/data**	publish	Aktuelles Datum des Kanals
***prefix*/*client*/*unit*/*channel*/cmnd**	subscribe	Zu setzender Wert des Kanals
***prefix*/*client*/*unit*/*channel*/update**	subscribe	Kanal aktualisieren

Die JSON-Meta-Beschreibung eines Kanalnamens besitzt folgenden Aufbau:

```
{
        "Writeable":    boolean,
        "Type":         "type",
        "Unit":         "unit",
        "Accuracy":     float,
        "IsBoolean":    boolean
        "Upper":        float
        "Lower":        float
        "Step":         float
        "Modes":        ["mode1","mode2", …, "moden"]
}
```

Nähere Beschreibung siehe unter A.2.6

Anhang E Das Makefile

Das makefile wird wie folgt aufgerufen:

sudo make Installationsart

Folgende Installationsarten sind möglich:

- install: Installation des Servers als auch FHEM-Moduls
- installService: Einrichten des Servers als Service
- installDebugService: Einrichten des Servers als Service im Remote-Debug-Mode

- installFHEM: Installation nur des FHEM-Moduls
- installSolvis: Installation des Servers

- uninstall: Deinstallation des Servers als auch des FHEM-Moduls
- uninstallFHEM: Deinstallation des FHEM-Moduls
- uninstallSolvis: Deinstallation des Servers
- uninstallService: Server ist nicht mehr ein Service des Systems
- uninstallDebugService: Server (im Debug-Mode) ist nicht mehr ein Service des Systems

- update bei einer neuen Version, ohne dass ein Learning notwendig ist. Der Service wird dabei gestoppt, die Dateien werden ausgetauscht und der Server wird wieder gestartet

- learn: Lernmodus des Servers starten. Im Anschluss wird der Service wird der Service wieder gestartet
- foreground: Server im Vordergrund (nicht als Service) starten
- debug Der Server wird gestartet, wobei die VM mit einer agentlib-Option (Port 10736) gestartet wird. Damit ist Remote-Debugging des Servers möglich