Solvis-SmartHome-Server / Fhem-Client

11.01.2020, letzte Änderung: 6.02.2020

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 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.

Anhang A **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 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)

Schnittstelle Server – Client 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.
- 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 XML-Files. Die XML Schema sind mit enthalten, so dass Anpassung mittels XML-Editor (z.B. integriert in Eclipse) stark vereinfacht wird

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

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.

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, ioBroker o.ä. umzusteigen.

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 steht sie 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, aber 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 1.8.0_231, auf der Linux-Seite (Rasbian) mittels OpenJDK 9.

4.2 SolvisSmartHome-Server

Hier wird nur die Installation unter Linux beschrieben. Wenn für Windows Bedarf besteht, ergänze ich das noch. Ich denke, das ist nicht notwendig, da in den meisten Fällen das SmartHome-System auf einem Linux-System laufen wird.

4.2.1 Dateien des Installationspaketes

Das Installationspaket beinhaltet folgende Dateien:

- Makefile — Zur installation
- 73_SolvisClient.pm — FHEM-Modul
- SolvisSmartHomeServer.jar — Das eigentliche Programm
- base.xml — Steuerdatei mit den Basis-Daten
- base.xsd — XML Schema hierzu
- SolvisSmartHomeServer.service — SystemD-Datei für die Service-Einrichtung

4.2.2 Anpassung der Datei *base.xml*

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

Es enthält aktuell (Stand 23.01.2020) 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"
    />
    <tns:Units>
        <tns:Unit
                id="mySolvis"
                type="SolvisMax"
                fwLth2_21_02A="true"
                account="account"
                password="password"
                url="http://aaa.bbb.ccc.ddd"
                defaultAverageCount="12"
                measurementHysteresisFactor="4"
                defaultReadMeasurementsInterval_ms="10000"
                forcedUpdateInterval_ms="3600000"
                bufferedInterval_ms="60000"
                delayAfterSwitchingOnEnable="true">
            <tns:ClockAdjustment
                    enable="true"
                    fineLimitUpper_ms="2000"
                    fineLimitLower_ms="-5000"
                    aproximatlySetAjust_ms="1000"
                    burstLength="0" />  <!-- burstLength = 0: fine tuning disabled -->
        </tns:Unit>
    </tns:Units>
</tns:BaseData>
```

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

- writablePathLinux: Pfad, in dem der Server seine Dateien ablegt
- id: ID der Solvis-Anlage, entspricht in FHEM dem Gerätenamen
- fwLth2_21_02A Die Solvis-RemoteControl-Firmware 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
- account: Account der Solvis-Anlage
- password: Passwort der Solvis-Anlage
- url: Url der Solvis-Anlage

Zu beachten:

Das Passwort steht unverschlüsselt in dieser Datei! Evtl. werde ich das in einer späteren Version ändern. Aktuell sollte man aber ein Passwort einrichten, was man sonst nicht verwendet. Das Makefile setzt die Zugriffsrechte dieser Datei auf 600, so dass nur mit Admin-Rechten oder unter der Kennung fhem (unter dem der Server läuft) auf diese Datei zugreifen kann.

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
defaultAverageCount:	Anzahl der Messwerte, über die der Mittelwert gebildet wird
measurementHysteresisFactor:	Hysteriess-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.
bufferedInterval_ms:	Bei bestimmten Messwerten kann hiermit die minimale Zeit zwischen der Ausgabe von 2 Messwerten bestimmt werden. Zum Disablen ist hier 0 einzutragen.
delayAfterSwitchingOnEnable:	Bestimmte Sensoren benötigen bei meiner Anlage nach dem Einscchlaten 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.

Daneben gibt es noch das ClockAdjustment-Element, über das man die Korrektur der Uhr steuern kann. Es besitzt folgende Attribute:

enable:	true/false: Ein-/Ausschalten der automatischen Korrektur
burstLength:	Bei jedem Stellen der Solvis-Uhr wird kurz die Zeit angehalten (bei meiner Anlage ca. 0,5s). Das kann genutzt werden, um einen Fein-Abgleich durchzuführen. Damit dieser aber die anderen Funktionen nicht verhindert, erfolgt nur alle 10 Minuten ein teilweises Angleichen. Wie viele Setzvorgänge hier erfolgen, gibt die burstLength an. Bei burstLength = 0 ist der Feinabgleich deaktiviert. Sinnvoll ist ein Wert von 4.
fineLimitUpper_ms:	Gibt die obere Grenze an, bei dessen Überschreitung ein Feinabgleich erfolgt. Dieser Wert kann kleiner sein, als der untere Wert, da ein Vorgehen der Uhr schnell korrigiert werden kann. Es sollte aber beachtet werden, dass dieser Wert nicht geringer sein darf, als die Laufzeitschwankungen des LAN/WLAN-Netzwerkes zur Solvis-Anlage hin.
fineLimitLower_ms:	Gibt die untere Grenze an, bei dessen Unterschreitung ein Feinabgleich erfolgt. Dieser Wert sollte großzügiger sein, als der obere Wert, da Nachkorrigieren einer nachgehenden Uhr länger dauert.
aproximatlySetAjust_ms:	Ist ein Schätzwert für die Zeit, über die die Solvis-Uhr beim Neusetzen kurz stehen bleibt. 1s es eher ein großer Wert, der angegebene Wert sollte aber nie kleiner als der wirkliche Wert sein.

4.2.3 Installation des Servers mittels Make

Zur Installation dient das beiliegende Makefile.

Es 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

- learn: Lernmodus des Servers starten
- 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

Standarmäßig wird man in der Regel das Makefile wie folgt starten:

sudo make install

Das gilt sowohl für die Erstinstallation als auch für Updates. Nur wenn der SolvisSmartHome-Server nicht auf dem gleichen System wie Fhem läuft, sind die anderen installationsarten interssant.

4.2.4 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, 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. Dieser 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 aber noch nicht, ganz selten muss man den Lernvorgang wiederholen.

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.

4.2.5 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.

4.2.6 Einrichten des Servers als Service

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

Make installService oder *Make installDebugService*

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.

4.2.7 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.

4.2.8 Deinstallation

Das Progarmmpaket kann mit folgendem Make-Befehl desinstalliert werden:

sudo make uninstall

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

4.2.9 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

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 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 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 aus 3 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 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).

6.3 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.

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.

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 3.01.2020):

```
2020-01-13 17:10:19,477|LEARN|Learning started.
2020-01-13 17:10:19,681|INFO|Solvis state changed to <SOLVIS_CONNECTED>.
2020-01-13 17:10:32,247|LEARN|Screen grafic <Home> learned.
2020-01-13 17:10:36,522|LEARN|Screen grafic <Solar> learned.
2020-01-13 17:10:40,092|INFO|Configuration mask: 1000001
2020-01-13 17:10:48,543|LEARN|Screen grafic <Nachttemperatur> learned.
2020-01-13 17:10:48,795|LEARN|Screen grafic <NachttemperaturNotSelected> learned.
2020-01-13 17:10:52,882|LEARN|Screen grafic <NachttemperaturSelected> learned.
2020-01-13 17:11:00,639|LEARN|Screen grafic <Warmwasser> learned.
2020-01-13 17:11:12,671|LEARN|Screen grafic <TagestemperaturNotSelected> learned.
2020-01-13 17:11:16,575|LEARN|Screen grafic <Tagestemperatur> learned.
2020-01-13 17:11:16,800|LEARN|Screen grafic <TagestemperaturSelected> learned.
2020-01-13 17:11:24,211|LEARN|Screen grafic <Zirkulation> learned.
2020-01-13 17:11:31,762|LEARN|Screen grafic <Sonstiges> learned.
2020-01-13 17:11:31,987|LEARN|Screen grafic <Sonstiges 1> learned.
2020-01-13 17:11:35,800|LEARN|Screen grafic <Heizkreise> learned.
2020-01-13 17:11:39,796|LEARN|Screen grafic <Heizkreis> learned.
2020-01-13 17:11:50,828|LEARN|Screen grafic <Anlagenstatus WW> learned.
2020-01-13 17:11:54,666|LEARN|Screen grafic <Anlagenstatus HK> learned.
2020-01-13 17:12:05,936|LEARN|Screen grafic <Sonstiges 2> learned.
2020-01-13 17:12:09,798|LEARN|Screen grafic <Zaehlfunktion> learned.
2020-01-13 17:12:17,514|LEARN|Screen grafic <Sonstiges 3> learned.
2020-01-13 17:12:21,333|LEARN|Screen grafic <Uhrzeit / Datum> learned.
2020-01-13 17:12:25,190|LEARN|Screen grafic <Zeiteinstellung> learned.
```

Wichtig:

Vor dem Start des Programms muss in der SolvisControl unter „Sonstig./Anlagenstatus“ der Bildschirm mit dem Warmwassser-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 3.01.2020):

```
2020-01-13 17:13:01,580|LEARN|Screen grafic <Zeiteinstellung_YYYY> learned.
2020-01-13 17:13:09,546|LEARN|Screen grafic <Zeiteinstellung_MM> learned.
2020-01-13 17:13:17,283|LEARN|Screen grafic <Zeiteinstellung_DD> learned.
2020-01-13 17:13:24,933|LEARN|Screen grafic <Zeiteinstellung_hh> learned.
2020-01-13 17:13:32,699|LEARN|Screen grafic <Zeiteinstellung_mm> learned.
2020-01-13 17:13:49,895|LEARN|Screen grafic <ModeTag> learned.
2020-01-13 17:13:52,954|LEARN|Screen grafic <ModeNacht> learned.
2020-01-13 17:13:56,041|LEARN|Screen grafic <ModeStandby> learned.
2020-01-13 17:14:00,163|LEARN|Screen grafic <ModeTimer> learned.
2020-01-13 17:14:03,459|INFO|Channel <C06.Anlagenmodus_HK1> is set to Standby>.
2020-01-13 17:14:10,245|LEARN|Screen grafic <WWPumpeAus> learned.
2020-01-13 17:14:13,260|LEARN|Screen grafic <WWPumpeAn> learned.
2020-01-13 17:14:16,333|LEARN|Screen grafic <WWPumpeAuto> learned.
2020-01-13 17:14:16,784|INFO|Channel <C04.WarmwasserPumpe> is set to auto>.
2020-01-13 17:14:16,785|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 SolvisControll (oder über SolvisRemote) erkannt, werden alle Anlagenparameter erneut gelesen, wenn die 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 aber 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.

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.

Anhang B **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)

Anhang C **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 (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 aber Client-spezifische Server-Einstellungen verloren gehen. Es empfielt 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 ist wie folgt aufgebaut:

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

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

A.2.2 CONNECTED

Der CONNECTED-Datensatz ist wie folgt aufgebaut:

```
{"CONNECTED":{"ClientId":427735588,"ServerVersion":"00.01.00","MinClientVersion":"00.01
.00"}}
```

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

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

A.2.3 RECONNECT

Der RECONNECT-Datensatz 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 dann 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 einen 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).

Wird noch ergänzt

Anhang D **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 heir daher verzichtet. Wenn jemand die control.xml verändern will, sollte er einen XML-Editor verwenden, der auch mit der XSD-Dateien umgehen kann (z.B. der XML-Editor des Eclipse-Pakets). Auf diese Weise werden die Fehlermöglichkeiten reduziert. In dem Editor kann mann sich dann in der Regel zu jedem Attribut/Element auch die dazugehörige Dokumention ansehen (in Eclipse durch Tool-Tips).

B.1 **Die XML-Datei log4j2.xml**

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 aber 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 aber noch weitere Infrastruktur, daher ist das erst mal auskommentiert.