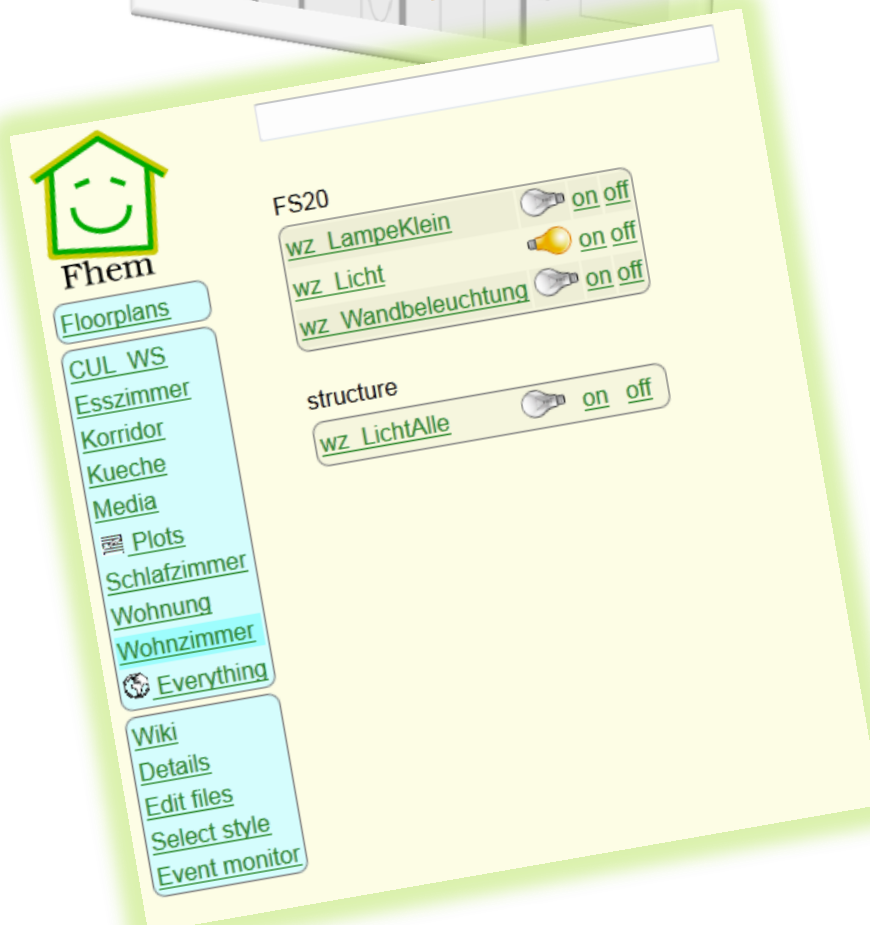


Heimautomatisierung mit fhem

- Für Einsteiger -



Ulrich Maaß

Homematic-Anhang von Martin Pittner

Inhaltsverzeichnis

Vorwort – aller Anfang ist schwer	4
Einleitung und Grundlagen	5
Hardware-Systeme: FS20, HomeMatic und Andere	6
Weitere Hardware: Computer und Computer-Funk-Schnittstelle (CUL)	8
Adressierung, Hauscode und Tastencode	9
Einrichten von Sensoren	11
Anlernen von Aktoren (Pairing)	13
Funktionsgruppen, Lokale Master, Globale Master	13
Planung Ihres Heimautomatisierungs-Systems	14
fhem-Grundlagen	16
fhem installieren	16
fhem Systemüberblick	16
Die fhem Benutzer-Oberfläche	16
Anlernen von Sensoren in fhem - autocreate	18
Schalten von Aktoren - set	19
Aufzeichnung in Logdateien	19
fhem-Konfiguration - Attribute	20
Attribute: Modell und Raum	21
fhem-Konfiguration: Das besondere Gerät „Global“	21
CUL: Definition und Attribute	23
Geräte-Infos: Readings	24
Mehrere Geräte mit einem Klick schalten - structure	24
Timer	25
Dimmer	25
Schalten zu bestimmten Zeitpunkten – at	26
Schalten von Ereignissen abhängig machen - notify	26
Verwendung von notify als Makro	27
Starten eines Makros – trigger	27
Bearbeiten über das Webfrontend	28
Pairing: direkt oder indirekt	28
Anpassen der Darstellung im Webfrontend - FHEMWEB-Attribute	31
Bildschirmgröße und "Skin": stylesheetPrefix	31
Weniger Menüpunkte: hiddenroom	31
Zusätzliche Menüpunkte – menuEntries	32
Nur bestimmte Befehle für ein Gerät - webCmd	32
Befehle umbenennen - eventMap	32
Räume sortieren: sortRooms	32
Gruppen bilden und umbenennen: group	33
Mehrspaltige Darstellung: column	33
Iconaktualisierung ohne browser-refresh – longpoll	33
Welche Icons? stylesheetPrefix und iconPath	33
Eigene Icons	34
Icon vor dem Gerätenamen - icon	34
Icon vor dem Raumnamen - roomIcons	34
Eigene Icons für Geräte-Schaltzustände	35
Weblink	35
Beispiel zur Konfiguration	36
Gruppen frei auf dem Bildschirm platzieren: Dashboard	37
Einrichten eines Grundrisses mit eigenen fhem-Geräten	37

Einfache Programmierung: if-Bedingung	37
Gerätewerte in Bedingungen – Value, ReadingsVal, ReadingsTimestamp	38
Spezielle Variablen und Operatoren	40
Hinterlegen eigener Programme – 99_myUtils.pm.....	41
Nur am Wochenende? \$we	41
In den Ferien? holiday, holiday2we.....	41
Verschachteltes at	42
Heizungssteuerung	43
Messen der Ist-Temperatur.....	43
Berechnen der erforderlichen Heizleistung	43
Thermostate.....	43
Regeln der Heizleistung.....	44
Heizungssteuerung mit fhem	44
Temperaturverläufe als Diagramm – Logs & Plots	45
Jemand zuhause? Anwesenheitserkennung und HomeStatus	46
Anwesenheitssimulation	47
Schaltungen abhängig von Telefonanrufen – Callmonitor	47
Multimedia-Geräte	48
Infrarot-Fernbedienung.....	48
Prüfung und Aufwecken von Servern im Heimnetzwerk: WOL.....	49
Webservices, http-Verbindungen	49
Wetterbericht einbinden.....	49
Google-Kalender, Mails und Messaging	50
Kommunikation über HTTP	50
fhem neu starten – shutdown restart	50
fhem Programmaktualisierungen – update.....	50
Infoquellen.....	51
Kurz betrachtet: Weitere Geräte und Funktionen.....	52
Erweitert: Logs & Plots	52
Dummy	52
Watchdog	52
Grenzwertgesteuertes Schalten – threshold und dewpoint	52
shell-commands	52
Zusammenfassungen – readingsGroup	52
Schalten an bestimmten Wochentagen	53
Kleine Programmierbeispiele	53
Wakeup-Light	53
Untoggle.....	53
Sunset und Sunrise.....	53
Tag und Nacht - isday.....	53
Erstellen eigener fhem-Programme.....	53
Entwicklung eigener fhem-Module.....	53
Sammelsurium	55
Anhang: Ein Einblick in die Konfigurationsdatei fhem.cfg	58
fhem.cfg und Includes	59
Anhang – Das Hardwaresystem HomeMatic	60

Vorwort – aller Anfang ist schwer

Auch ich habe mal mit Heimautomatisierung und fhem angefangen. Ein super Produkt, zu dem man sich jedoch alle Infos schnipselweise zusammensuchen musste: Von wo herunterladen, wo die Hardware beziehen, was braucht man überhaupt, wie initialisiere ich die Hardware.... Fragen über Fragen. Um anderen genau diese Anfangsschwierigkeiten zu ersparen, habe ich dieses Dokument geschrieben: Als Einstieg und Überblick zu den ersten Schritten. Als die erste Version entstand, war ich mit meinen ‚Grundlagenforschungen‘ gerade fertig– zwei Monate nach meinem Start mit fhem. Das war im Mai 2011. Seitdem wurde das Dokument bereits ca. 300.000 mal heruntergeladen – Ansporn für die nun vorliegende vierte Version.

In der zweiten Version wurde der Grundlagenteil überarbeitet und Kapitel zur fhem-Konfiguration durch das Webfrontend, zur Anpassung des Webfrontends sowie zu ersten kleinen fhem-Perl-Programmen sind hinzugekommen.

Zur Version 3 hat Martin Pittner einen Anhang zur HomeMatic-Konfiguration beigesteuert.

Die nun vorliegende vierte Version ist um Kapitel wie Heizungssteuerung, Anwesenheitserkennung, HomeStatus und Multimedia sowie viele kleine weitere Neuerungen erweitert worden.

Über ein kurzes Feedback von Lesern freue ich mich übrigens sehr! Hat's geholfen, ist eine bestimmte Passage nicht so gut verständlich, fehlt etwas? Schickt mir doch einfach mal ne kurze Mail an <mailto:uli.maass@gmail.com>

Mein Dank gilt den Korrekturlesern Annette, Klaus, Andreas und Peter sowie allen Entwicklern von fhem, vor allem Rudolf König.

Viel Spaß beim Lesen und vor allem mit fhem,

Uli

Einleitung und Grundlagen

Wenn Sie das erste Mal einen Blick auf fhem ("Freundliche Hausautomation und Energie-Messung", sprich "feem") werfen, möchten Sie sicher wissen, was Sie damit erreichen können.

Heimautomatisierung bedeutet: Wenn ich auf einen Schalter drücke, soll das Licht angehen; wenn es kalt ist, soll die Heizung eingeschaltet werden; bei Sonnenaufgang soll die Gartenwegbeleuchtung ausgehen; um 7 Uhr morgens soll der Rollladen hochfahren, sofern ich nicht in Urlaub bin. Geräte sollen auch vom PC, Tablet-PC oder Smartphone bedienbar sein. Alles das lässt sich mit fhem bewerkstelligen.

Damit lassen sich auch die Hauptziele der Heimautomatisierung erreichen:

- Energie sparen durch das zielgerichtete Regeln bzw. Ein- und Ausschalten von Geräten
- Mehr Komfort durch Automatisierungen – zeitgesteuert oder abhängig von Messdaten wie z.B. der Temperatur, dem Sonnenstand etc.
- Zugriff auch über (W)LAN und das Internet
- Spaß beim Basteln :-)

In der Welt der Heimautomatisierung gibt es dafür vier Gruppen von Geräten:

1. **Sensoren** (Thermometer, Lichtschalter, Fernbedienungen, Bewegungsmelder, Infrarot-Empfänger etc. - also Geräte, die Werte aus der Umwelt aufnehmen und an das System senden).
2. **Aktoren** (Steckdosenschalter, Unterputzschalter, Rollladen-Schalter etc. – also jedes Gerät, das Befehle vom System entgegennimmt und als Schaltvorgang umsetzt).
3. **Sender&Empfänger**, die als Funk-zu-PC-Schnittstelle fungieren und die (Funk-)kommunikation übernehmen – das kann ein Hardwaresystem-spezifisches PC-Interface sein oder auch ein "CUL" genannter generischer Sender&Empfänger.
4. Außerdem benötigen Sie einen **Computer**, also einen PC oder einfachen Rechner wie z.B. ein NAS, einen RaspberryPi oder eine FritzBox, der ständig eingeschaltet ist, an der der CUL angeschlossen ist und auf dem fhem läuft.

Sensoren und Aktoren gibt es von mehreren Herstellern. Jeder Hersteller hat dabei ein jeweils in sich geschlossenes **Hardwaresystem**, das zunächst auch ohne fhem funktioniert. Beispiele dafür sind die Systeme FS20, FHT, HomeMatic, OneWire oder EnOcean (wenn Sie keine Ahnung haben, was das alles ist – später mehr). Jedes dieser Systeme umfasst eine Vielzahl von Komponenten, bestehend aus Sensoren wie z.B. Bewegungsmelder, Thermostate, Wandschalter (Unterputz und Aufputz) und Fernbedienungen, sowie Aktoren wie z.B. Steckdosenschalter (Funk-Zwischenstecker), funkgesteuerte Steckdosenleisten, Dimmer, Stellantriebe für Heizkörperventile oder Rollladenschalter.

Die meisten dieser Hardwaresysteme funktionieren per Funk in den speziellen Frequenzbändern 433 MHz oder 868 MHz. Bei manchen Systemen erfolgt die Kommunikation aber auch über Kabel wie z.B. in den Hardwaresystemen One-Wire oder KNX/EIB.

Der überwiegende Teil dieser Hardwaresysteme bietet auch ein PC-Interface, um Schaltvorgänge vom PC, ggf. auch über Webfrontend und mit Automatisierungen, steuern zu können. Im FS20-System heißt dieser PC-Adapter z.B. FHZ1300PC, bei HomeMatic heißt er CCU oder LAN-Adapter.

fhem kann sich in diese Hardwaresysteme "einklinken". Dazu wird entweder der systemeigene Adapter genutzt (es wird also die FHZ1300PC oder der LAN-Adapter von fhem aus gesteuert), oder man verwendet statt dieser relativ teuren Hardwaresystem-eigenen PC-Interfaces (=Sender&Empfänger) ein generisches Gerät, den sogenannten 'CUL'. Dieser ist deutlich günstiger und kann z.T. mehrere Hardwaresysteme gleichzeitig bedienen. Der CUL kann also z.B. eine FHZ1300PC ersetzen.

Vermutlich besitzen Sie schon jetzt Geräte, die mit Hilfe von fhem gesteuert werden können: einen modernen SmartTV, eine FritzBox, vielleicht auch ihr AV-Receiver? Diese können Sie ohne Mehrkosten steuern und Überwachen.

Wenn Sie tiefer in die Heimautomatisierung mit fhem einsteigen möchten, sollten Sie sich zunächst für (mindestens) eines der oben genannten Hardwaresysteme entscheiden und die entsprechenden zueinander passenden Komponenten beschaffen.

Einfache Anwendungen, die sich dann gestalten lassen, sind z.B.:

- Ein Sensor soll einen Aktor steuern, z.B. soll mit einem Lichtschalter an der Wand eine Lampe ein- und ausgeschaltet werden.
- Derselbe Schalter soll evtl. auch mehrere Geräte (also Aktoren) bedienen, wenn z.B. mit nur einem Tastendruck mehrere Lampen oder beliebige Verbraucher ein- und ausgeschaltet werden sollen.
- Es sollen evtl. auch mehrere Schalter die gleiche Funktion auslösen, z.B. soll im Schlafzimmer das Licht sowohl vom Lichtschalter an der Wand als auch von der Fernbedienung am Bett bedienbar sein.
- Wenn Sie fhem nutzen, können Sie alle Schaltvorgänge sowohl von einem Webfrontend auf einem Computer als auch von einer App auf einem mobilen Gerät steuern. Außerdem können Sie zeit- und ereignisbasierte Schaltvorgänge auslösen.

Eine solche Konfiguration können Sie ganz ohne Programmierkenntnisse erreichen, es braucht nur einiges technisches Grundverständnis.

Wenn Sie Aktoren ausschließlich vom Webfrontend (also ohne physischen Schalter) bedienen wollen, müssen Sie diese durch wenige einfache Programmzeilen ‚hinzuprogrammieren‘.

Das Zusammenfassen von mehreren Aktoren in Räumen mit gemeinsamer Schaltung vom Webfrontend aus ist ebenfalls mit wenigen Programmzeilen recht einfach oder kann innerhalb des Hardwaresystems eingerichtet werden (also z.B. alle Lampen im Schlafzimmer sollen mit nur einem Tastendruck ausgehen).

Wenn Sie weitergehende Möglichkeiten nutzen möchten, z.B. Zeitsteuerungen wie Laufzeiten, Sonnenauf-/untergang, Ferienkalender, thermostatgeführte Heizungssteuerung etc., müssen Sie zunehmend tief in die fhem- und ggf. Perl-Programmierung einsteigen.

fhem ist sehr umfangreich. Es bietet die Möglichkeit, einfache Konfigurationen ohne Programmierkenntnisse zu erstellen, wie auch eine komplexe Heimautomatisierung, die eben Programmierung benötigt.

Vor dem Einstieg in fhem ist es erforderlich, zunächst die Funktionsweise der Steuerung zu verstehen. Also geht es zunächst um

Hardware-Systeme: FS20, HomeMatic und Andere

Sicher haben Sie einfache Hardware-Systeme schon einmal im Baumarkt gesehen: eine Funk-Fernbedienung, an die einige Zwischensteckdosen gekoppelt werden können. Mit der Fernbedienung kann man dann bequem Lampen oder sonstige Geräte ein- und ausschalten.

Grundsätzlich sind die Hardwaresysteme, die im Kontext zu fhem relevant sind, nichts anderes – sie bieten aber eine größere Anzahl unterschiedlicher Geräte, die miteinander gekoppelt werden können,

auch sind diese Hardwaresysteme erweiterbar. Und ja: auch manche (!) der vorgenannten Baumarkt-Steckdosen lassen sich aus fhem heraus schalten.

Das derzeit am weitesten verbreitete Hardwaresystem heißt FS20. Die Geräte dafür werden im Wesentlichen von den Firmen ELV und Conrad vertrieben. Wenn Sie auf deren Webseiten den Suchbegriff ‚FS20‘ eingeben, bekommen Sie eine bebilderte Übersicht der Komponenten: Sensoren wie Wand-Lichtschalter, Fernbedienungen, Bewegungsmelder, Thermostate oder Helligkeitssensoren, Aktoren wie Steckdosenschalter, Infrarotsender, Steckdosenleisten, Rollladenschalter etc. - schauen Sie doch gleich mal nach.

In FS20 können Sensoren ausschließlich senden und Aktoren ausschließlich empfangen (der Lichtschalter an der Wand kann Funktelegramme nur senden, die Steckdose kann sie nur empfangen). Das ist eigentlich ausreichend – bis ein Aktor z.B. durch eine Funkstörung oder wegen schlechter Laune mal einen Befehl verpasst. Das FS20-System geht einfach davon aus, dass der gesendete Befehl auch ausgeführt wurde. Im vergleichbaren Hardware-System HomeMatic hingegen können Aktoren auch senden, sie senden also eine Bestätigung, dass sie den Befehl auch wirklich ‚gehört‘ haben. Das ist zwar noch keine absolute Garantie, dass der Befehl auch wirklich umgesetzt wurde, aber immerhin gibt es ein Feedback. Bleibt dieses aus, kann z.B. der Befehl erneut gesendet oder eine Warnmeldung ausgegeben werden. HomeMatic-Komponenten sind somit deutlich zuverlässiger, aber auch teurer (sie kosten häufig das Doppelte der vergleichbaren FS20-Komponente). Wenn Sie kritische Geräte steuern möchten, z.B. Ihre Rollläden auch als Einbruchsicherung dienen und daher definitiv geschlossen sein sollen, können Sie HomeMatic in Erwägung ziehen. Wenn es lediglich um das Ein- oder Ausschalten von Lampen u. ä. geht, oder Sie für gewöhnlich Zuhause sind und die Schaltergebnisse selbst sehen, ist FS20 wahrscheinlich ausreichend. Die Erfolgsquote beim Schalten von FS20-Geräten liegt bei mittleren Entfernungen, mittleren Eigenschaften der Wände und gutem Batterie-Ladestand der Sensoren (=Sender) bei gefühlten 95-98%.

Sowohl HomeMatic (HM) als auch FS20-Systeme lassen sich mit fhem steuern. Eine Mischung beider Systeme ist möglich – allerdings benötigen Sie dann einen CUL je System - also einen für FS20, einen für HM.

Bei der Definition eines Geräts in fhem gibt man den Typ des Hardwaresystems an. Wenn Sie in fhem ein Gerät definieren, müssen Sie dabei angeben, ob es sich um FS20, Homematic usw. handelt. Ist diese Gerätedefinition einmal vorgenommen, können dann alle fhem-Funktionen gleichartig genutzt werden, da das Gerät dann nur noch über seinen Namen angesprochen wird. Das Einschalten eines Geräts erfolgt z.B. durch den Befehl `set Lampe on`. Dabei ist es dann gleichgültig, ob das Gerät `Lampe` ein FS20, HomeMatic oder anderes Gerät ist.

Im Weiteren wird FS20 für alle Beispiele verwendet. Eine Erklärung zum Hardwaresystem HomeMatic finden Sie im Anhang "Das Hardwaresystem HomeMatic".

Die größte Besonderheit von fhem ist, dass mehrere Hardwaresysteme eingebunden und ggf. auch kombiniert werden können. Vermutlich werden Sie mit *einem* Hardwaresystem starten, haben aber die Möglichkeit, Ihr System später zu erweitern und sind bei dieser Erweiterung somit nicht auf nur ein Hardwaresystem festgelegt. Eine vollständige (ständig wachsende) Liste finden Sie in der [fhem Befehlsreferenz](#) am Anfang unter „Devices“ sowie [auf dieser Seite](#).

Weitere Hardware: Computer und Computer-Funk-Schnittstelle (CUL)

Sie benötigen den **Sender&Empfänger**. Ein CUL sieht aus wie ein USB-Stick und kann an einen PC, ein NAS oder z.B. eine FritzBox angeschlossen werden. Zu beziehen ist der CUL von busware.de. Für den RaspberryPi gibt es dort das Aufsteckmodul COC, das die gleiche Funktion wie der CUL übernimmt, dabei jedoch keinen USB-Steckplatz belegt. Eine weitere Alternative bildet der CUNO: ebenfalls von busware zu beziehen, wird jedoch über LAN an den Computer angebunden. Von demselben Anbieter gibt es auch die Geräte TuxRadio und TuxRail – beide bieten nicht nur die Sende/Empfangs-Funktionalität, sondern haben auch einen Prozessor und RAM – sie sind ein Mini-Computer incl. Linux und fhem, es wird kein weiteres Gerät wie z.B. ein NAS oder eine FritzBox benötigt. Der CUL ist jedoch das am weitesten verbreitete und damit am besten getestete Gerät und wird daher empfohlen. Im weiteren Text wird nur CUL erwähnt, der dann synonym für „FHZ1300PC, HM LAN-Adapter, CCU, CUL, CUNO, Tuxradio oder Tuxrail sowie alle anderen Schnittstellen zwischen PC und dem Hardwaresystem“ steht.

Der CUL von busware kann leider nur ohne Firmware bestellt werden. Diese Firmware muss also zunächst auf den CUL „ge-flasht“ werden. In der neuesten fhem-Version wird ein CUL automatisch geflasht, der an einem Gerät angeschlossen wird, auf dem fhem läuft. Probieren Sie es also zunächst so. Sollte dies nicht funktionieren, können Sie Details in der fhem-Befehlsreferenz, der [commandref](#), nachlesen.

Alternativ können Sie das Flashen des CUL auch an einem PC ausführen. Die Anleitung dazu finden Sie hier: [Flashen der CUL-Firmware an einem PC](#)

Sie müssen den **Sender&Empfänger betreiben**. Es gibt unterschiedliche Geräte, an denen der Sender betrieben werden kann: Ein PC mit Windows oder Linux, Linux-fähige NAS-Geräte oder auch eine FritzBox. Da es in vielen Haushalten auch um das Sparen von Strom geht, nutzen Sie am besten ein Gerät, das ohnehin ständig eingeschaltet ist.

fhem ist in der Programmiersprache Perl implementiert, d.h. grundsätzlich können Sie jedes Gerät verwenden, das über einen Netzwerkanschluss, einen USB-Anschluss und eine Perl-Installation verfügt. Da ich meine Telefonanlage und Heimnetzwerk mit einer FritzBox 7390 betreibe, ist dies in meinem Haushalt das ideale Gerät: Es ist ständig eingeschaltet, es stellt eine CPU und Speicher zur Verfügung und ist im Heimnetzwerk eingebunden. Außerdem gibt es auf der Hersteller-Webseite einen Download für eine FritzBox-Firmware inklusive fhem. Man kann einfach diese Labor-Firmware (oder noch besser die Alternative von fhem.de) herunterladen und über die FritzBox-Oberfläche wie ein Firmware-Update installieren, den CUL einstecken und das fhem-System funktioniert. Einfach so. Hinweise zur Einrichtung der FritzBox 7390 mit fhem gibt es hier: [FB7390: fhem-Installation](#); zum besseren Grundverständnis dient auch das [Schichtenmodell](#).

Ebenfalls sehr gut zum Betrieb von fhem geeignet ist der Raspberry Pi. Es handelt sich um einen zigaretenschachtel-großen Kleincomputer mit nur 4W Leistungsaufnahme, der für den Betrieb von fhem aber völlig ausreichend ist. Gegenüber der FritzBox bietet er den Vorteil, dass weitere Perl- und Linux-Komponenten nachinstalliert werden können und dass er performanter ist. Eine Einrichtungsanleitung für den Raspberry Pi findet sich z.B. [hier](#), auch gibt es einen [Wiki-Artikel](#).

Sie benötigen außerdem **Sensoren und Aktoren**. Im Weiteren werden als Beispiel eine funkgesteuerte Steckdose (FS20-st) und ein Lichtschalter (FS20-S4A) betrachtet. Andere Geräte funktionieren ähnlich: die Zwischen-Steckdose (FS20-st) arbeitet ebenso wie ein Unterputzschalter (FS20-SU). Ein Wand-Lichtschalter mit vier Knöpfen wirkt genauso wie eine Fernbedienung mit vier Knöpfen (FS20 S4).

Hinweis: Von einigen Hardware-Komponenten finden Sie in den Webshops (leider) unterschiedliche Revisionsstände, z.B. FS20-st-2 oder FS20-st-3. Diese sind in ihrer Funktion identisch und können auch gemischt betrieben werden.

Wenn Sie starten möchten, bestellen Sie den Sender&Empfänger (CUL) zuerst, da er ein paar Tage Lieferzeit hat. Zum Testen können Sie sich außerdem z.B. zunächst 2 Funk-Schalt-Steckdosen (FS20-st) und einen Funk-Lichtschalter (FS20-S4A) besorgen.

Adressierung, Hauscode und Tastencode

Jedes Hardwaresystem verwendet eine ‚Adressierung‘, durch die alle Komponenten des Systems eindeutig identifiziert sind. So verwendet z.B. HomeMatic eine ab Werk im jeweiligen Gerät fest codierte ID als eindeutige Identifikation innerhalb dieses Hardwaresystems. Intertechno arbeitete mit einer Adressierung, die über DIP-Schalter an den Komponenten eingestellt wird. In FS20 wird am Sender der Hauscode eingestellt, der Empfänger wird auf eine Adresse "angelernt". So hat jedes Hardwaresystem seinen eigenen Mechanismus, um Komponenten eindeutig ansprechen zu können.

Das FS20-System arbeitet im 868 Megahertz Funkfrequenzband. Um die Geräte gegebenenfalls innerhalb Ihres Haushalts und vor allem von denen Ihrer Nachbarn zu unterscheiden, wird der sogenannte *Hauscode* verwendet. Der **Hauscode** ist die System-ID Ihres gesamten Hardwaresystems, der auf allen Komponenten, die gemeinsam funktionieren sollen, gleich ist und sie als ‚zusammengehörig‘ kennzeichnet. Normalerweise werden Sie in ihrem gesamten Hardwaresystem denselben Hauscode verwenden – so sind die Hardwaresysteme vom jeweiligen Hersteller ursprünglich konzipiert. Ein Hauscode in FS20 ist achteinstellig, wobei jede Stelle Werte zwischen 1 und 4 annehmen kann. Ein Beispiel ist ‚12341234‘. Die Wahl Ihres eigenen Hauscodes ist beliebig, aber wichtig, falls z.B. auch Ihr Nachbar ein FS20-System betreibt – er soll ja nicht plötzlich Ihre Wohnzimmer-Beleuchtung einschalten können.

Am Sensor – also dem Wandschalter oder der Fernbedienung- wird jeder Taste bzw. jedem Tastenpaar außerdem ein **Tastencode** zugewiesen. Wird eine Taste gedrückt, sendet der Sensor ein Funktelegramm mit der Information Hauscode und Tastencode. Anderen Sensoren, wie z.B. einem Bewegungsmelder, wird ebenfalls ein Tastencode zugeordnet, damit im Funktelegramm eindeutig ist, von welcher Komponente das Funktelegramm stammt. An einigen wenigen Komponenten sind Hauscode und Tastencode nicht frei wählbar, da diese keine Tasten zur Eingabe haben, z.B. ein Thermostat. Diese Geräte werden sich somit nicht in Ihren Hauscode einreihen lassen, fallen dadurch etwas aus der Reihe – funktionieren aber trotzdem :-)

Alternativ können Sie bei Verwendung eines CUL auch z.B. einen Hauscode je Etage Ihres Hauses verwenden, oder aber einen Hauscode je Raum oder sogar je Sensor. Der Vorteil liegt im Einrichten des Sensors, denn ab Werk sind zwar Hauscodes zufällig vergeben, Tastencodes jedoch immer gleich. Wenn Sie jedem Sensor seinen Werks-Hauscode lassen oder je Sensor einen anderen Hauscode vergeben, haben Sie mit geringem Aufwand wiederum die Eindeutigkeit der Funktelegramme erreicht – es gibt zwar mehrere Tasten mit demselben Tastencode, diese sind jedoch durch die unterschiedlichen Hauscodes unterscheidbar.

FS20 bietet die Möglichkeit, **Schaltgruppen** zu bilden, die sogenannten Funktionsgruppen, Lokale Master und Globale Master. Sie sind dazu gedacht, mehrere Aktoren, die sonst auch einzeln schaltbar sein sollen, mit nur einem Tastendruck gleichzeitig zu schalten (z.B. alle Lampen in einem Raum mit nur einem Tastendruck ausschalten).

Wichtig: diese im Hardwaresystem definierten Gruppen funktionieren nur innerhalb desselben

Hauscodes! Die Gruppen werden durch vom Hardwaresystem vorgegebene, spezielle Tastencodes gebildet. Eine detaillierte Beschreibung finden Sie im Handbuch jedes FS20-Sensors. Diese Handbücher sind auch als pdf im Webshop der Anbieter ELV und Conrad verfügbar. Auch ist zu beachten, dass ein Aktor gegebenenfalls mehrfach angelernt werden muss – je einmal für Funktionsgruppe, Local Master und Global Master.

Alternativ lassen sich solche Schaltgruppen ebenso in der fhem-Konfiguration bilden – ebenfalls über unterschiedliche Hauscodes hinweg. Das ist bequemer einzurichten als das Einstellen der Tastencodes an den Schaltern, führt jedoch beim Schalten von Gerätegruppen zu mehr Funkverkehr: es wird dann nicht ein Funktelegramm mit dem Gruppen-Tastencode gesendet, sondern ein Funktelegramm je Kombination Hauscode+Tastencode (=Adresse) – also ein Funktelegramm je Gerät. Da auf Grund funkrechtlicher Bestimmungen ([1%-Regel](#)) nicht mehr als 160 Funktelegramme pro Stunde gesendet werden dürfen, kann dies in großen Systemen zu Problemen führen. Bitte berücksichtigen Sie dies, wenn Sie Ihre Systematik von Haus- und Tastencodes entwerfen.

Als Empfehlung für den Start: Je Hauscode können bis zu 225 Tastencodes definiert werden. Wenn dies für Sie ausreichend ist, nutzen Sie nur einen Hauscode, um ggf. später Schaltgruppen einführen zu können. Reicht Ihnen diese Anzahl nicht, nutzen Sie einen Hauscode je Etage. Damit können Sie Gruppen innerhalb der Etage durch das Hardwaresystem bilden, Gruppen über Etagen hinweg in fhem. Wenn Sie also 3 Etagen haben, und alle Lampen im Haus ausschalten wollen, müssen Sie nur 3 Funktelegramme senden (ein Funktelegramm für die entsprechende Gruppe je Etage).

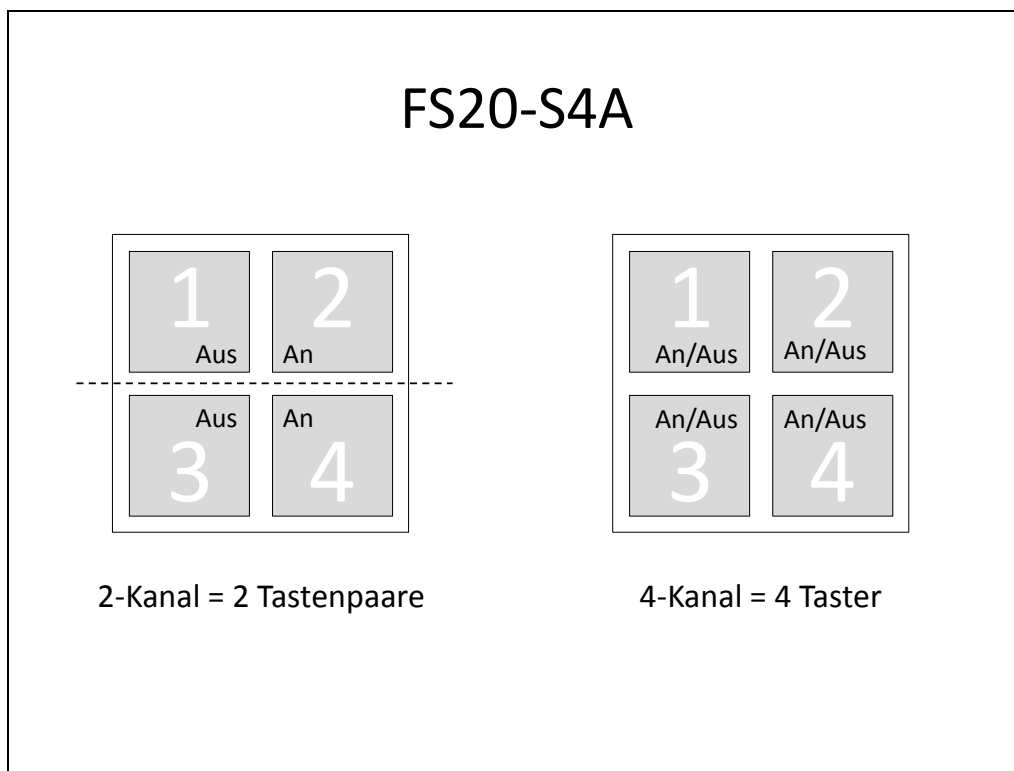
Zu Gruppen und fhem später mehr.

Einrichten von Sensoren

Wenn Sie einen Sensor in Betrieb nehmen, müssen Sie einmalig den Hauscode und den/die Tastencode(s) einstellen, die fortan von dieser Komponente gesendet werden sollen.

Hinweis: Wenn Ihr fhem bereits läuft, wird jede Taste, die Sie an einem Sensor drücken, sofort ‚angelernt‘. Auch wenn’s schwerfällt – bitte erst lesen, dann drücken :-)

Als ersten Sensor betrachten wir einen Lichtschalter FS20-S4A mit vier Knöpfen, genaugenommen vier Tastern. Ab Werk sind diese Lichtschalter mit einem zufälligen Hauscode belegt und als 2-Kanal-Schalter eingestellt, d.h. dass sie wie zwei Tastenpaare bedient werden können – für zwei Geräte jeweils einen An- und Aus-Taster. Wenn Sie diese Einstellung beibehalten, ist die weitere Konfiguration am einfachsten.



FS20 S4A: Unterschiedliche Funktion bei Konfiguration als 2- oder 4-Kanal-Schalter

Der Wandschalter sendet also bei Betätigung einer Taste ein Funktelegramm bestehend aus Hauscode, Tastencodes und den auszuführenden Befehl, z.B. on oder off. Dieses Funktelegramm wird von allen Aktoren in Reichweite „gehört“ – es reagiert jedoch nur der Aktor, der auf die passende Kombination von Hauscode und Tastennummer „angelernt“ oder „ge-paired“ ist – dazu mehr auf den nächsten Seiten.

Wenn mehrere gleichartige Schalter dieselben Aktoren ansprechen sollen, geben Sie ihnen einfach dieselbe Kombination von Hauscode und Tastencodes. Dadurch erzeugt z.B. der 4-knöpfige Lichtschalter an der Wand dieselben Funktelegramme wie die 4-knöpfige Fernbedienung. Da ein Aktor (Steckdose) lediglich Hauscodes und Tastencodes unterscheidet, ist für ihn das Signal vom Lichtschalter dasselbe wie das der Fernbedienung – und ggf. das des CUL, nachdem Sie das Funktelegramm in fhem angelernt haben – auch dazu später mehr.

Sie können mit den vier Tastern aber auch vier unterschiedliche Geräte jeweils ein- und ausschalten, jede einzelne Taste verhält sich dann wie ein Ein-/Aus-Schalter (Hinweis: nennt man auch „toggle“) für ein Gerät.

Details zur Einstellung stehen im jeweiligen Handbuch. Für einen FS20-S4A gilt:

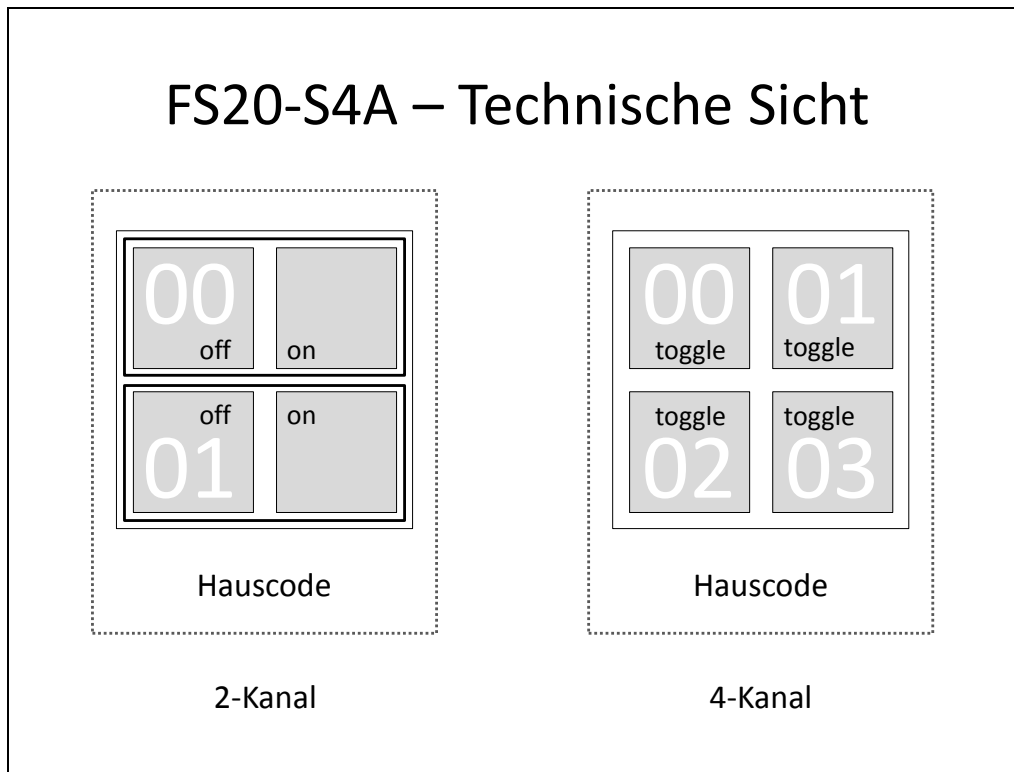
- Hauscode einstellen: Tasten 1+3 für 5 Sekunden halten bis die LED blinkt. Dann den Hauscode als beliebige 8-stellige Zahl festlegen (Aufschreiben!), dabei kann jede der acht Stellen einen Wert zwischen 1 und 4 haben, entsprechend der 4 Tasten. Ein Hauscode wäre also z.B. 12341234.

Hinweis: Setzen Sie bei diesen Schritten den Deckel auf den Schalter – eintippen direkt auf der Platine funktioniert nicht!

- Tasten-Benutzung: Für 2 Tastenpaare mit jeweils „aus“ und „an“ halten Sie die Tasten 2+3 für 5 Sekunden, bis die LED kurz blinkt. Wenn Sie 4 einzelne Taster jeweils zum Ein- und Ausschalten nutzen möchten (Achtung – die fhem-Einrichtung ist später komplizierter!) halten Sie die Tasten 1+4 für 5 Sekunden, bis die LED kurz blinkt.

Der Lichtschalter ist nun entsprechend Ihren Vorgaben konfiguriert.

Bevor der Sensor in fhem eingebunden wird, hier noch kurz die ‚Technische Sicht‘ auf den Schalter.



FS20 S4A: Unterschiedliche Tastencodes bei Konfiguration als 2- oder 4-Kanal-Schalter

Im FS20-System werden beim Betätigen einer Taste immer genau folgende Informationen in einem **Funktelegramm** übermittelt:

- **FS20-Adresse**, bestehend aus Hauscode und Tastencode
- Der Zielzustand (Befehl)

Wenn Sie Ihren Schalter als 2-Kanal konfiguriert haben und Sie drücken den Taster oben rechts, sendet dieser <Hauscode> 00 on. Drücken Sie unten links, sendet er <Hauscode> 01 off.

Hinweis: Sollte Ihnen die Standard-Tastenbelegung ,links = OFF, rechts = ON‘ „unnatürlich“ erscheinen, drehen Sie den Schalter einfach um 90 Grad nach links, dann ist oben ,an‘ und unten ,aus‘ :-)

Haben Sie Ihren Schalter als 4-Kanal konfiguriert, sendet dieser statt on/off lediglich den Befehl ,umschalten‘, englisch ,toggle‘. Drücken Sie die Taste oben rechts, sendet der Schalter demzufolge `<Hauscode> 01 toggle`. Je nachdem, ob das zu schaltende Gerät also zuvor an oder aus war, wird es durch toggle in den entgegengesetzten Zustand versetzt.

Ein unschöner Nebeneffekt ist die Darstellung auf dem fhem Web-Interface: während passend zu den Zuständen on und off eine hübsche ,leuchtende‘ oder ,graue‘ Glühlampe erscheint, wird bei Vier-Kanal-Tastern ein immer gleiches Icon angezeigt. Wie man das ,ausbügeln‘ kann, wird später erklärt – es erfordert Programmierung (siehe [untoggle](#)).

Hinweis: Wenn Sie einen Taster länger als 0,4 Sekunden drücken, werden statt on/off/toggle die analogen Befehle dimup/dimdown/dimupdown zum Dimmen gesendet. Auf diese gehen wir erst später ein.

Anlernen von Aktoren (Pairing)

Um nun einen Aktor (also z.B. eine Steckdose) mit einem Sensor zu koppeln, versetzt man den Aktor durch 5-sekündiges Festhalten des Knopfes (bis der blinkt) in den Lernmodus, und sendet dann einen Befehl durch Drücken des entsprechenden Tasters. Fertig! Der Aktor hat nun die Kombination Hauscode und Tastencode des Sensors/Lichtschalters ,gelernt‘ und reagiert nur noch auf Befehle mit dieser Kombination.

Der Schalter und der Aktor bilden infolgedessen nun ein Paar, das durch das Funktelegramm eindeutig identifiziert ist: Dieselbe Kombination von Hauscode und Tastencode wird vom Sensor gesendet und vom Aktor umgesetzt.

Falls Sie in Ihrem Hardwaresystem auch Gruppen definiert haben, wiederholen Sie den Anlernvorgang ggf. je einmal für eine Funktionsgruppe und/oder einen Lokalen Master und/oder einen Globalen Master durch Drücken der Taste, die mit dem Gruppen-Tastencode eingerichtet ist.

Funktionsgruppen, Lokale Master, Globale Master

Wie bereits erwähnt, bieten viele Hardwaresysteme die Möglichkeit, Gerätegruppen zu bilden. So kann mit nur einem Funktelegramm eine Mehrzahl von Aktoren geschaltet werden.

In FS20 werden Gruppen durch spezielle Tasten- bzw. Gruppencodes eingerichtet. Die betroffenen Aktoren müssen dann mehrfach angelernt (ge-paired) werden: Einmal für ihre individuelle Adresse (siehe vorheriges Kapitel), dann ein weiteres Mal je Gruppe, der sie zugeordnet sein sollen. Ein Aktor kann somit bis zu vier Mal angelernt werden – und reagiert dann auf jegliches Funktelegramm, das den korrekten Hauscode sowie einen der angelernten Tasten- bzw. Gruppencodes entspricht.

- Globaler Master – der Grundgedanke ist, dass alle Aktoren innerhalb des Hauscodes auch auf diese Gruppenadresse angelernt werden. Dadurch können mit nur einem Funktelegramm ALLE Geräte ein- oder ausgeschaltet werden, z.B. als Not-Aus. Je Hauscode gibt es nur einen Tastencode (44 44) für den Globalen Master.
- Lokaler Master – gedacht für z.B. alle Geräte in einem Raum. Je Hauscode gibt es bis zu 15 Lokale Master.
- Funktionsgruppen – gedacht für raum-übergreifende Gruppen, z.B. alle Lichter, alle Audiogeräte etc. Je Hauscode gibt es bis zu 15 Funktionsgruppen.

Die Einrichtung von Gruppen in fhem wird später in diesem Dokument erklärt. Es ist jedoch zu beachten, dass die Autocreate-Funktion für gewöhnlich auf Basis der individuellen Geräteadresse ‚lernt‘. Damit der in fhem angezeigte Schaltzustand des einzelnen Geräts auch dann korrekt angezeigt werden kann, wenn ein Gerät durch einen Gruppenbefehl geschaltet wurde, müssen Globaler Master, Lokaler Master und Funktionsgruppen manuell zur Definition der Geräte hinzugefügt werden. Details dazu finden Sie in der [FS20-commandref](#) bei den Informationen zu define.

Eine einfache Anleitung zur Einrichtung findet sich auch im drittletzten Beitrag dieses [Posts](#).

Planung Ihres Heimautomatisierungs-Systems

Bevor Sie alle Ihre Sensoren und Aktoren bestellen, machen Sie sich eine Skizze, wie Ihre Heimsteuerung aussehen soll. Aspekte, die Sie dabei berücksichtigen sollten, sind:

- Welche Geräte möchten Sie schalten und steuern (z.B. Lampen, Rollläden, Heizkörperventile, Kaffeemaschinen, Radios, WLAN an/aus etc.)?
- Wo befinden sich diese Geräte, in welcher Struktur lassen sie sich zusammenfassen (z.B. Etage und/oder Zimmer)?
- Wo sollen welche Sensoren platziert werden?
- Welche Gruppen von Aktoren sollen gemeinsam geschaltet werden können?

Bauen Sie sich eine Liste auf, in der alle Paare von Sensoren und Aktoren gelistet sind. Geben Sie jedem dieser Paare einen Namen sowie einen Haus- und Tastencode, z.B.

Sensor	Hauscode	Tastencode	fhem-Name	Schaltet Gerät (Aktor)
FS20-S4A Wohnzimmer, Erdgeschoss, Tastenpaar 01	12341234	01	eg_wz_Stehlampe	Stehlampe, Wohnzimmer, Erdgeschoss
FS20-S4A Schlafzimmer, Erdgeschoss, Tastenpaar 00	12341234	00	og_sz_RolladenLinks	Rolladen links, Schlafzimmer, Obergeschoss

Sie können beliebige Namen wählen, die jedoch eindeutig sein müssen. Gerätenamen dürfen neben Punkt und Unterstrich keine Sonderzeichen enthalten. Ein weiteres Beispiel zur Benennung der Geräte finden Sie [hier](#).

Definieren Sie auch, welche Geräte gemeinsam in **Gruppen** geschaltet werden sollen. Auslöser ist auch bei Gruppen ein bestimmter Tastencode. Führen Sie bei den Aktoren auf, welche Ihrer Aktoren bei Auslösen dieser Gruppe geschaltet werden sollen. Bedenken Sie, dass die Bildung von Gruppen nur innerhalb eines Hauscodes möglich ist.

Sensor	Hauscode	Gruppen- und Tastencode	fhem-Name (Gruppe)	Schaltet Gerät (Aktor)
FS20-S4A 2 Wohnzimmer, Erdgeschoss, Tastenpaar 00: Alle Lichter WZ	12341234	44 11*	eg_wz_LichtAlle	Funktionsgruppe fg 11: Stehlampe, Wandleuchte, Deckenlicht, Tischleuchte, Wohnzimmer, Erdgeschoss
FS20-S4A 2 Wohnzimmer, Erdgeschoss, Tastenpaar 01: Media	12341234	44 12*	haus_MediaAlle	Funktionsgruppe fg 12: Verstärker, TV, Videorecorder, Airplay Küche, Airplay Bad

*) Eine detaillierte Erklärung der Adress-Struktur finden Sie im Handbuch für den FS20 S4A ab Seite 12.

Mit den bisherigen Ausführungen sollte die Funktionsweise des FS20-Systems deutlich geworden sein. Weitere Details entnehmen Sie den FS20 Handbüchern, die jedem Sensor beiliegen und auch online einsehbar sind – schauen Sie z.B. auf der ELV-Homepage unter FS20-S4A nach, dort finden Sie das Handbuch als pdf.

fhem-Grundlagen

Wir können uns nun dem eigentlichen Spaß zuwenden: der Einbindung der gekoppelten Geräte. In diesem Abschnitt werden grundlegende FS20- und fhem-Befehle dargestellt. Es liegt im Auge des Betrachters, ob man dies als Scripting oder bereits als Programmierung bezeichnet :-)

Auch werden wir die unterschiedlichen Möglichkeiten kennenlernen, um diese Konfigurationen vorzunehmen.

fhem installieren

Die Installationsschritte sind abhängig vom verwendeten Computer. Grundsätzlich sind folgende Schritte erforderlich:

1. Installieren des Betriebssystems, z.B. Linux oder Windows
2. Installieren von Perl (dies ist Voraussetzung, da fhem in Perl programmiert ist)
3. Installieren von fhem

Im [fhem-Wiki](#) finden Sie [Anleitungen](#) zur Installation für unterschiedliche Betriebssysteme und teilweise auch für bestimmte Hardware, z.B. FritzBox oder QNAP NAS. Bitte schlagen Sie dort nach.

fhem Systemüberblick

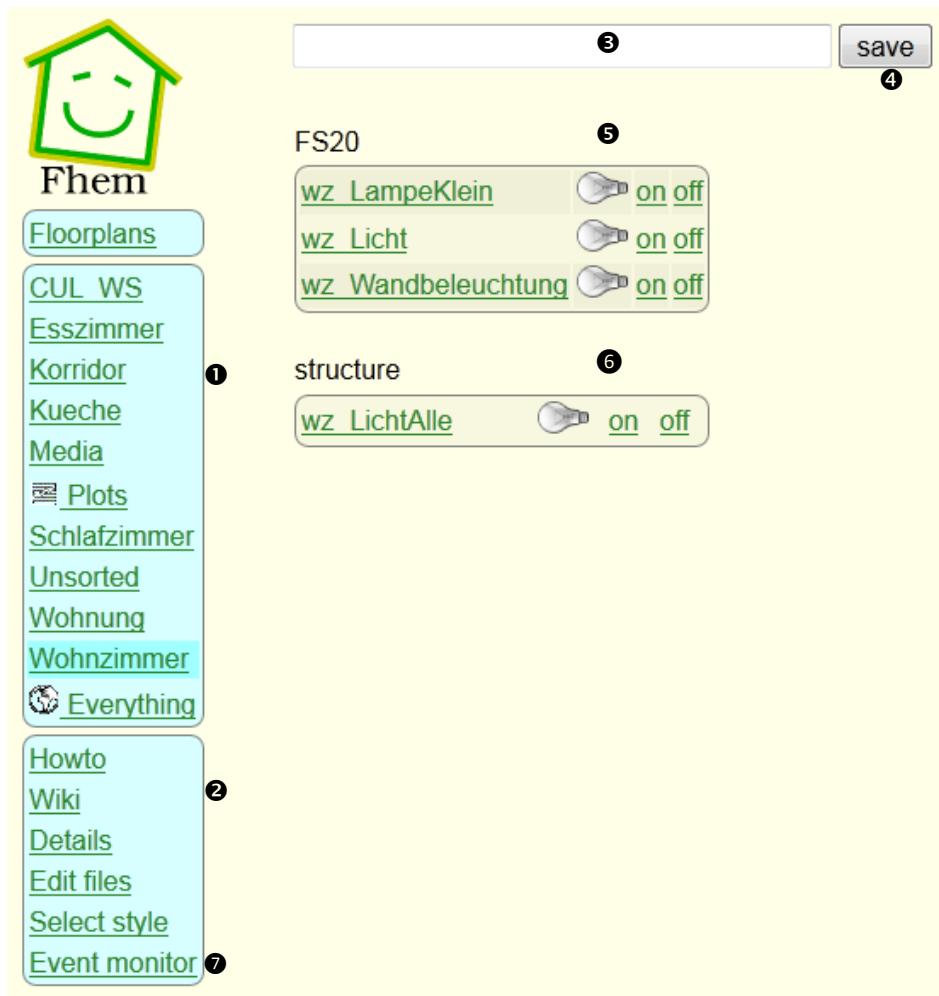
fhem besteht aus dem Kernel (fhem.pl) sowie einer Vielzahl von Modulen. Module können die unterschiedlichsten Aufgaben übernehmen vom Anbinden eines Hardwaresystems über die Bereitstellung eines Frontends bis zur Automatisierung von Aufgaben. Eine Übersicht der grundlegenden Systemarchitektur finden Sie [hier](#).

Die fhem Benutzer-Oberfläche

Bevor die ersten Befehle über die fhem-Oberfläche getätigt werden, hier ein erster kurzer Blick auf das Standard-Frontend von fhem, genannt „pgm2“.

Sie erreichen fhem über Ihren Browser unter der Adresse <http://<ip>:8083/fhem>

Auf der nächsten Seite finden Sie eine kurze Erklärung des Bildschirmaufbaus.



- ❶ Liste der Räume – zu Beginn der Einrichtung erscheinen nur ‚Unsorted‘ und ‚Everything‘. Räume werden den Geräten in der Konfiguration mittels `attr <Gerät> room <Raumname>` zugeordnet.
- ❷ Links auf Hilfetexte, die fhem-Doku und zur Konfigurationsseite ‚Edit files‘
- ❸ Eingabefeld zum direkten Ausführen von fhem-Befehlen
- ❹ Mit Klick auf den `save`-button speichern Sie Ihre Konfigurations-Änderungen in der Datei `fhem.cfg`. **Achtung!** Nach Eingabe eines fhem-Befehls NICHT auf den SAVE-button klicken, sondern lediglich `<ENTER>` betätigen!
In neueren fhem-Versionen gibt es statt des im screenshot gezeigten save-buttons im oberen Bereich des Menüs den Eintrag "save config", dieser hat dieselbe Funktion.
- ❺ Nach der Einrichtung erscheinen hier Ihre Geräte mit der Darstellung des Schaltzustands und einem on/off-Schalter
- ❻ Um Schaltgruppen statt im Hardwaresystem alternativ in fhem aufzubauen, können Geräte auch Strukturen zugewiesen werden. Sie erlauben das gesammelte Schalten aller zugeordneten Geräte - auch über unterschiedliche Hauscodes hinweg.
- ❼ Mit dem Event-Monitor können Sie alle Funkbefehle verfolgen. Sehr hilfreich, um Ablauf und Inhalt der Funkkommunikation zu verstehen.
Hinweis: Der Funkverkehr ist auch über ein telnet-Terminalprogramm an Port 7072 zu sehen, wenn Sie in der telnet-Terminalsitzung `inform timer` eingeben.

Anlernen von Sensoren in fhem - autocreate

Nachdem am Sensor der gewünschte Hauscode und die gewünschten Tastencodes eingestellt sind, werden die Tasten(paare) nacheinander in fhem angelernt.

Hinweis: Sensoren können automatisch an fhem gekoppelt werden. Bevor Sie fortfahren, stellen Sie sicher, dass die dafür zuständige Funktion „autocreate“ aktiviert ist. Tippen Sie dafür im fhem webfrontend den Befehl `list autocreate` in das Kommandofeld ein und bestätigen mit <ENTER>. In der Ausgabe muss u.a. STATE active erscheinen.

```
Internals:
  NAME      autocreate
  NR        6
  STATE     active
  TYPE      autocreate
Attributes:
  autosave  1
  filelog    /var/InterneSpe
```

Erscheint dies nicht, geben Sie in das Kommandofeld ein

```
define autocreate autocreate
```

um das automatische Anlegen von Geräten zu aktivieren.

Um den Sensor mit fhem zu koppeln, drücken Sie die erste zu ‚lernende‘ Taste auf dem Schalter (Sensor). Dadurch sendet dieser ein Funktelegramm, das vom CUL empfangen und an fhem weitergegeben wird. Da die Kombination von Hauscode und Tastencode bisher nicht bekannt ist, wird von fhem automatisch ein zugehöriges fhem-device angelegt. Auf dem fhem webfrontend (mit F5 den Web-Browser refreshen) erscheint dieses nun mit einem kryptischen Namen, sie heißt dort z.B. FS20_602b01. Es wurde von fhem also ein Funktelegramm mit dem vorläufigen Namen FS20_602b01, nämlich dem (hexadezimal dargestellten) Hauscode 602b und dem Tastencode 01 erkannt. Notieren Sie sich auch diese vierstellig hexadezimale Version des Hauscodes – ich habe mir dafür eine Excel-Tabelle angelegt. Auf Ihrem fhem-frontend finden Sie im oberen Bereich das weiße Eingabefeld ③, über das Sie direkt getippte fhem-Befehle absetzen können. Der erste Befehl soll dem Umbenennen des Schalters dienen. Überlegen Sie sich die Bezeichnung Ihres Schalters gut, denn unter diesem Namen wird er fortan überall angezeigt. Ich habe mich dafür entschieden, die ersten zwei Stellen für das Zimmer zu verwenden (also wz=Wohnzimmer, ez=Esszimmer, sz=Schlafzimmer usw.) und dann eine Beschreibung anzuhängen. Meine Sensor-Tasten heißen so, wie die Geräte, die geschaltet werden sollen, also z.B. wz_LampeSofa, sz_LeseLampe, ez_LichtRegal. Man kann sich ein beliebiges Namenssystem überlegen, jedoch müssen die Namen eindeutig sein und dürfen keine Leerstellen enthalten.

Hinweis: In der iPhone/iPad-App fhemobile wird der Teil vor dem Unterstrich nicht angezeigt, aus wz_LampeSofa wird in der Anzeige für das Wohnzimmer also LampeSofa.

Hinweis: Die Darstellung der Werte nach ELV-Quad-Format und hexadezimaler Darstellung ist alternativ – beide Varianten funktionieren. Die Umrechnung erfolgt [so](#).

Angenommen Sie möchten wz_LampeKlein schalten: Tippen Sie im weißen Eingabefeld

```
rename <NameAlt> <NameNeu>, in unserem Beispiel also
```

```
rename FS20_602b01 wz_LampeKlein
```

Das ist nun also der neue fhem-Name des Funktelegramms, also dem Paar aus gedrücktem Taster bzw. Taster-Paars und dem entsprechend angelernten Aktor. Auf dem fhem-frontend erscheint nun wz_LampeKlein mit den klickbaren Links für ON und OFF.

FS20	State	Set to
wz_LampeKlein	 on off	
wz_Licht	 on off	
wz_Wandbeleuchtung	 on off	

Damit ist Ihr erstes fhem-device erstellt. Sie können diese Steckdose nun vom Schalter und vom Webfrontend steuern.

Um Ihre neue Konfiguration mit dem umbenannten FS20-device dauerhaft in der Konfigurationsdatei fhem.cfg zu speichern,

- Klicken Sie links im Menü auf "save config", oder
- geben Sie im Eingabefeld den Befehl **save** ein und bestätigen mit <ENTER>.

Schalten von Aktoren - set

Versuchen Sie im Webfrontend das Klicken auf ON und OFF. Schalten Sie die Steckdose mit dem Schalter ein und aus – wenn sich dabei die Änderung im Web-Frontend nicht automatisch ändert, aktualisieren Sie die Darstellung in Ihrem Browser mit F5 und kümmern Sie sich später um Iconaktualisierung ohne browser-refresh – longpoll. Sie können das Schalten von Geräten auch durch getippte Befehle erreichen: in dem weißen Feld im oberen Bereich der fhem-Seite, dem Kommandofeld, können direkte Befehle eingegeben werden. Probieren Sie `set <Name> on` und `set <Name> off`, also z.B. `set wz_LampeKlein on`

Wenn derselbe Schalter mehrere Aktoren schalten soll, lernen Sie die weiteren Aktoren einfach genauso wie den ersten am selben Sensor (Schalter) an.

Um zu verstehen, welche Funktelegramme mit welchen Inhalten gesendet werden, verwenden Sie den Event Monitor  oder Terminal 7072 `inform timer` (Anleitung siehe *Zugang über Terminalprogramm, Telnet*)

Mit dem set-Befehl können auch mehrere Geräte gemeinsam geschaltet werden. So können Sie z.B. mehrere Geräte aufzählen:

```
set lamp1,lamp2,lamp3 on
```

oder –falls Ihre Geräte passend benannt sind- einen Bereich angeben:

```
set lamp[1-3] on
```

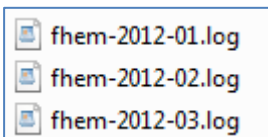
oder wildcards verwenden (eine Perl-Besonderheit: * reicht nicht, es muss .* angegeben werden):

```
set lamp.* on
```

Weitere Beispiele und Erklärungen finden Sie in der commandref unter [Device Specifications](#). Mit den bisher beschriebenen Schritten lässt sich bereits eine recht umfangreiche Konfiguration erreichen.

Aufzeichnung in Logdateien

Alle Funktelegramme, egal ob von einem Sensor oder von fhem/CUL gesendet, werden in Logdateien mitgeschrieben. Besonders wichtig ist die Haupt-Log-Datei von fhem. Diese ist normalerweise in Monatsscheiben aufgeteilt. In meinem System sieht das so aus:



Der Inhalt der Logdatei sind alle Funktelegramme, aber auch Fehlermeldungen von fhem. Insbesondere deshalb ist es wichtig, immer mal wieder, spätestens aber beim Auftreten von Fehlern in die Logdatei zu schauen.

Hier ein Auszug:

```
2012.03.12 08:40:00 2: FS20 set sz Leselampe dim100% 1280
2012.03.12 09:01:01 2: FS20 set sz_Rollo off
2012.03.12 09:20:01 2: FS20 set sz_Leselampe off
2012.03.12 09:20:01 2: FS20 set sz_Stehlampe off
```

Die Funktion autocreate legt außerdem eine Logdatei je Gerät an. Insbesondere für Messgeräte wie Temperatursensoren ist diese wichtig, da der Inhalt die Grundlage für die Darstellung in Diagrammen ist. Dazu später mehr.

Bevor weitere fhem-Befehle erklärt werden, zunächst ein Blick auf die Konfiguration von fhem.

fhem-Konfiguration - Attribute

Die in fhem angelernten bzw. manuell angelegten Geräte können weiter konfiguriert werden, indem ihnen ‚Attribute‘ mit einem Wert zugewiesen werden.

Die Einstellungen z.B. zu `ez_LichtRegal` können über das webfrontend eingesehen werden, indem man nach dem Klick im linken Bereich auf ‚Everything‘ nicht auf ON oder OFF, sondern auf den Namen des Gerätes klickt, z.B. auf `ez_LichtRegal`. Dadurch erreicht man diesen sogenannten device-**Detail-**

Bildschirm:

The screenshot shows the fhem web interface for the device `ez_LichtRegal`. The sidebar on the left contains links for various rooms and general settings. The top bar has links for 'Delete ez_LichtRegal' and 'Modify ez_LichtRegal'. Below this is a table with columns 'State', 'Value', and 'Measured'. The 'State' is 'off', and the 'Value' is 'toggle'. The 'Measured' value is '2011-09-21 19:11:58'. Below this is a table with columns 'Internal' and 'Value'. The 'Internal' table lists various device parameters and their values. The 'Attribute' table lists attributes and their values, with a 'deleteattr' button for each attribute.

Internal	Value
BTN	00
CUL_MSGCNT	1
CUL_RAWMSG	F6969001238
CUL_RSSI	-46
CUL_TIME	2011-09-21 19:11:58
DEF	6969 00
LASTIODev	CUL
MSGCNT	1
NAME	ez_LichtRegal
NR	11
STATE	on
TYPE	FS20
XMIT	6969

Attribute	Value	Action
room	Wohnung	deleteattr
Wohnung	Alle	deleteattr
fm_order	2	deleteattr
model	fs20st	deleteattr
room	Esszimmer	deleteattr

Dieser Block zeigt die „Internals“ eines Geräts. Hier ist der fhem-Name (NAME ❶) des Geräts zu sehen.

(DEF ❷) zeigt die Definition, wie sie zum Zeitpunkt des „define“-Befehls dieses Geräts angegeben wurde. Im screenshot sind dies Hauscode und Tastencode.

Der aktuelle Schaltzustand wird ebenfalls angezeigt (STATE ❸).

Im unteren Bereich werden die Attribute des Geräts angezeigt, die hier auch verändert werden können. Ein Beispiel ist die Angabe des FS20-Modells (model ❹).

Die Veränderung von Geräte-Attributen (Details später) können Sie auf drei unterschiedlichen Wegen erreichen:

1. Klicken Sie im oben gezeigten Detail-screen auf einen Attribut-Namen, um dessen Wert zu bearbeiten – oder wählen Sie aus der Dropdown-Liste das Attribut aus, das Sie hinzufügen möchten. Als Beispiel wählen Sie das Attribut `model` aus, dann in der rechten Liste `fs20-st` und klicken dann die Schaltfläche ‚attr‘. Schließen Sie Ihre Konfiguration ab durch Eingabe des Befehls `save` in das Eingabefeld oder klicken auf "save config" im Menü, durch den Ihre Konfiguration in der Datei `fhem.cfg` gespeichert wird.
2. Geben Sie den Befehl `attr wz_MediaServer model fs20-st` in das Eingabefeld ein und bestätigen Sie mit <ENTER>. Schließen Sie Ihre Konfiguration ab durch Eingabe des Befehls `save` in das Eingabefeld, durch den Ihre Konfiguration in der Datei `fhem.cfg` gespeichert wird.
3. Klicken Sie im linken Bildschirmbereich auf Edit Files, dann auf `fhem.cfg` (oder verwenden Sie einen anderen Editor). Scrollen Sie zu der Definition Ihres devices. Fügen Sie darunter die Zeile `attr wz_MediaServer model fs20-st` ein. Klicken Sie im oberen Bildschirmbereich den button ‚save fhem.cfg‘.

Alle diese Bearbeitungsvarianten erzielen denselben Effekt: Sie ändern Ihre fhem-Konfiguration und speichern Ihre Änderungen in der Datei `fhem.cfg`. Am bequemsten ist der Weg 1, der auch Tippfehler bei der Eingabe weitgehend vermeidet. Zur besseren Lesbarkeit sind im

weiteren Text alle Befehle so dargestellt, wie sie in das Kommandofeld eingetippt werden können – alternativ können Sie alle Befehle aber auch durch Klicks im Detailbildschirm der Geräte erreichen.

Probieren Sie die drei Varianten der fhem-Konfiguration aus: Über über das webfrontend, über Kommandozeile oder direkt in der fhem.cfg. Es wird empfohlen, den Großteil der Konfigurationsschritte mit Variante 1 oder 2 vorzunehmen. Lediglich das Definieren neuer devices (also `define...`) muss über die Varianten 2 oder 3 erfolgen, da es das device im webfrontend ja noch nicht gibt und man daher auch nicht dorthin navigieren kann.

Besondere Beachtung verdient die Information STATE (siehe Screenshot ❸). STATE zeigt den zuletzt gesendeten Zustand des Geräts. Meist ist dieser Zustand `on` oder `off`, kann aber auch `toggle` oder `dim50%` oder (bei einem Temperatursensor) `T: 8.0 C` sein – je nachdem, was zuletzt gesendet wurde und welches Gerät verwendet wird.

Attribute: Modell und Raum

Kennzeichnen Sie auch die bereits in Ihrer Konfiguration vorhandenen Schalter mit dem FS20 Steckdosenmodell (denn diese steuern Sie ja damit):

```
attr wz_Lampe model fs20-st
```

Die Angabe des Modells erlaubt eine passendere Darstellung in den fhem-frontends, z.B. werden für Schalter keine Dimm-Befehle angeboten etc.

Auch können Sie nun jedes device einem Raum in Ihrer Wohnung zuordnen:

```
attr wz_MediaServer room Wohnzimmer
attr wz_Lampe room Wohnzimmer
```

Sie sehen nun im Menü links einen neuen Raum ‚Wohnzimmer‘. Wenn Sie dort klicken, werden alle Geräte angezeigt, die diesem Raum zugeordnet wurden.

Hinweis: Einen existierenden Raum können Sie im Detailscreen durch Auswahl aus der dropdown-Liste zuordnen. Wenn Sie einen neuen Raum anlegen möchten, müssen Sie diesen dem ersten relevanten Gerät durch Eingabe des Befehls in das Kommandofeld zuordnen. Allen weiteren Geräten kann dann auch dieser neue Raum wieder über Auswahl aus der Dropdown-Liste zugeordnet werden.

Wie bereits erklärt, gibt es drei Möglichkeiten zur Konfiguration von fhem:

1. Über die Detail-Sicht der einzelnen Geräte
2. Über die fhem Kommandozeile
3. Über Einträge direkt in der Datei fhem.cfg

Wichtig: Bei Nutzung der Varianten 1 und 2 werden Ihre Änderungen erst dann in die Datei fhem.cfg geschrieben, wenn Sie über die Kommandozeile den Befehl `save` ausführen. Da der Weg 1 empfohlen wird, werden Sie für gewöhnlich nicht direkt in fhem.cfg arbeiten. Dennoch ist es wichtig, die Syntax und den Inhalt zu verstehen.

fhem-Konfiguration: Das besondere Gerät „Global“

Nachdem Sie gesehen haben, wie Attribute gehandhabt werden, können wir uns nun einem ganz besonderen Gerät zuwenden: dem Gerät „Global“. Es wird grundsätzlich ebenso behandelt wie alle anderen Geräte in fhem, sendet jedoch keine Funknachrichten aus, sondern dient als Speicherort für die grundlegende Konfiguration in fhem. Dabei werden alle Einstellungen über Attribute vorgenommen.

Hinweis: Änderungen an Geräten werden erst dann in der Konfiguration gespeichert, wenn man im Kommandofeld den Befehl save eingibt und dann <ENTER> drückt. Verwendet man diesen Befehl nicht, sind alle Eingaben beim nächsten Neustart verloren.

Hinweis: Für Details lesen Sie den [Anhang: Ein Einblick in die Konfigurationsdatei fhem.cfg](#).

Ein direktes Bearbeiten dieser Datei ist für gewöhnlich nicht erforderlich – ebenso wenig wie das direkte Bearbeiten der Registry eines Windows-Systems.

The screenshot shows the fhem web interface. On the left is a sidebar with a 'Fhem' logo and a list of floorplans and rooms. The main area displays the configuration for a device named 'global'. In the 'Internals' section, 'NAME' is set to 'global' and 'currentlogfile' is set to '/var/InternerSpeicher/Intenso-MicroLine-01/fhem/log/fhem-2014-01.log'. Below this, the 'Attributes' section lists various configuration parameters such as 'autoload_undefined_devices', 'backup_before_update', 'configfile', 'holiday2we', 'logfile', 'modpath', 'mold', 'mseclog', 'room', 'sendStatistics', 'statefile', 'uniqueID', 'userattr', 'verbose', and 'version'. Each attribute has a 'deleteattr' link next to it.

Das Gerät „global“ zeigt u.a. den Namen des aktuellen logfiles an

Die Attribute des Geräts „global“ steuern das Systemverhalten.

Die wichtigsten Attribute im Überblick:

Attribut	Beschreibung
modpath	Gibt an, unter welchem Pfad des Betriebssystems sich der Ordner fhem befindet. Der Pfad hängt von dem Betriebssystem und häufig auch der Hardware ab, auf der fhem installiert wurde. attr global modpath /var/InternerSpeicher/fhem (hier muss natürlich der zu Ihrem Gerät passende Pfad stehen!) Wenn dieses Attribut in Ihrer fhem.cfg –Datei bereits eingestellt ist und fhem funktioniert, lassen Sie diese Angabe am besten unverändert :)
logfile	Ebenso wichtig ist die Pfadangabe zur fhem Logdatei: attr global logfile /var/InternerSpeicher/fhem/log/fhem-%Y-%m.log (Hier muss natürlich der zu Ihrem Gerät passende Pfad stehen!). Durch die Angabe %Y-%m wird eine separate Datei pro Monat pro Jahr angelegt – wie weiter oben im Internal-Wert „currentlogfile“ zu sehen ist. Das erlaubt eine zielgerichtete Archivierung, da die Logdateien je nach Anzahl und Art der verwendeten Geräte recht schnell recht groß werden können. Im Beispiel ist zu sehen, dass die Logdateien auf einen USB-Stick ausgelagert wurden.

statefile	Neben der fhem.cfg spielt auch die Datei fhem.save eine zentrale Rolle. In dieser Datei werden die Schaltzustände aller Geräte gespeichert, wenn Sie <code>save</code> ausführen– so werden alle Geräte nach einem Neustart von fhem wieder mit ihrem letzten Schaltzustand angezeigt. Der Speicherort dieser Datei ist festgelegt durch <code>attr global statefile /var/InternerSpeicher/fhem/log/fhem.save</code>
verbose	Legt fest, wie viele Meldungen im Logfile protokolliert werden sollen: <code>verbose 1</code> logt nur Katastrophen, <code>verbose 5</code> jeden kleinsten Schritt. <code>verbose 3</code> ist der default, mit dem ein „gesundes Mittelmaß“ mitgeloggt wird.

Eine vollständige (lange) Liste aller global-Attribute finden Sie in der commandref [hier](#).

CUL: Definition und Attribute

Hier ein kurzer Blick auf die Definition des CUL in fhem. Neben dem Systempfad zum CUL-device und der Baudrate zur Kommunikation wird in der **DEF**inition auch die FHT-ID angegeben.

Internals

CMD5	BCFiAGMRTVWXefmltux
CUL_MSGCNT	3849
CUL_TIME	2014-01-15 17:37:01
Clients	:FS20:FHT.*:KS300:USF1000:BS:HMS: :CUL_EM:CUL_WS:CUL_FHTTK:CUL_RFR:CUL_HOERMAN :ESA2000:CUL_IR:CUL_TX:Revolt:IT:
DEF	/dev/ttyACM0@38400 2332
DeviceName	/dev/ttyACM0@38400
FD	5
FHTID	2332
NAME	CUL
NR	3
NR_CMD_LAST_H	5
PARTIAL	
RAWMSG	T515100B66F34
RSSI	-48
STATE	Initialized
TYPE	CUL
VERSION	V 1.42 CUL868
initString	X21

Readings

ccconf	freq:868.300MHz bWidth:325KHz rAmpl:42dB sens:4dB
cmds	B C F i A G M R T V W X e f m l t u x
raw	N/A
state	Initialized
version	V 1.42 CUL868

attr CUL room System

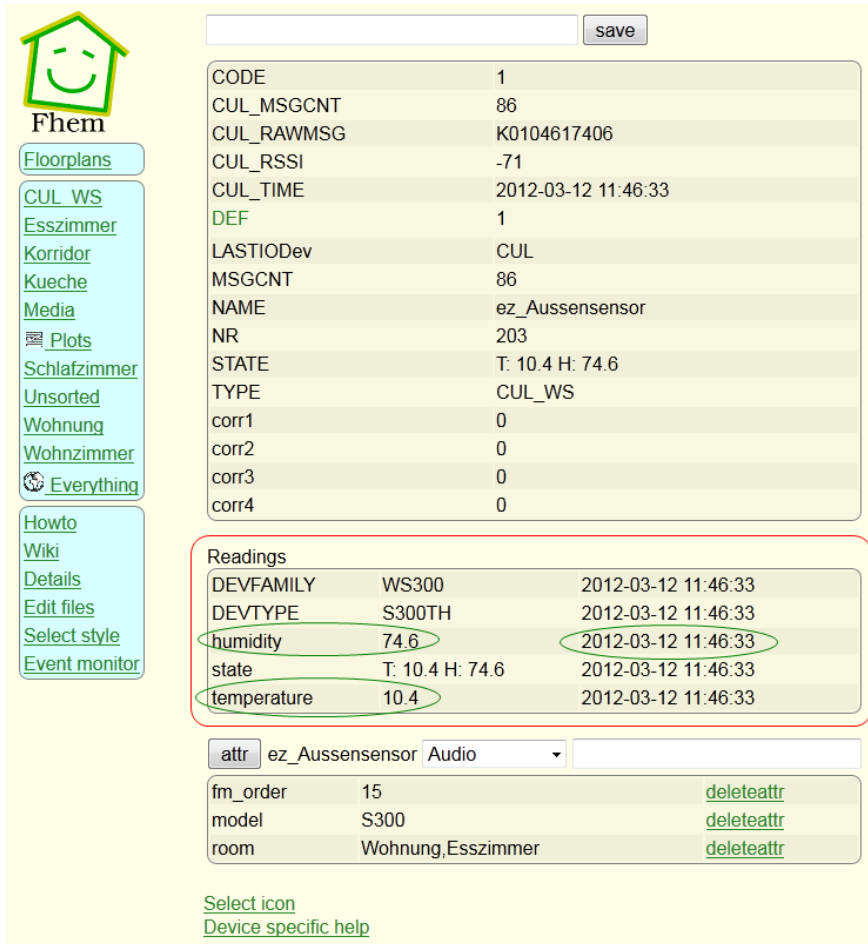
Die FHT-ID (im Beispiel 2332) ist quasi der Hauscode des Hardwaresystems FHT, das parallel zu FS20 betrieben werden kann und zur Heizungssteuerung dient. Solange Sie keine FHT-Geräte betreiben (z.B. FHT80b, FHT8V, ...) setzen Sie die FHT-ID auf 0000, da sonst unnötige Funktelegramme gesendet werden.

Zum Betreiben von FS20-Komponenten wird der Hauscode ausschließlich in der Definition der FS20-Geräte verwendet, es ist nicht erforderlich, den FS20-Hauscode dem CUL zuzuweisen.

Geräte-Infos: Readings

Einige Geräte stellen zusätzliche Informationen zur Verfügung, sogenannte Readings. So wird z.B. bei einem Bewegungsmelder der Zeitpunkt des letzten Auslösens angezeigt, bei einem Temperatur- und Luftfeuchte-Sensor (S300TH) stehen eben die zusätzlichen Readings Temperatur, Luftfeuchtigkeit und Zeitpunkt der letzten Übermittlung zur Verfügung. Auch diese Readings werden im Detail-Bildschirm eines Geräts angezeigt.

Hier als Beispiel ein Temperatur- und Luftfeuchte-Sensor:



The screenshot shows the fhem web interface. On the left is a sidebar with a house icon and the 'fhem' logo, followed by a list of links: Floorplans, CUL_WS, Esszimmer, Korridor, Kueche, Media, Plots, Schlafzimmer, Unsorted, Wohnung, Wohnzimmer, Everything, Howto, Wiki, Details, Edit files, Select style, and Event monitor. The main content area has a search bar and a 'save' button. Below this is a table of device attributes:

CODE	1
CUL_MSGCNT	86
CUL_RAWMSG	K0104617406
CUL_RSSI	-71
CUL_TIME	2012-03-12 11:46:33
DEF	1
LASTIODev	CUL
MSGCNT	86
NAME	ez_Aussensensor
NR	203
STATE	T: 10.4 H: 74.6
TYPE	CUL_WS
corr1	0
corr2	0
corr3	0
corr4	0

Below the attributes table is a 'Readings' section, highlighted with a red border. It contains a table with three columns: attribute, value, and timestamp.

DEVFAMILY	WS300	2012-03-12 11:46:33
DEVTYPE	S300TH	2012-03-12 11:46:33
humidity	74.6	2012-03-12 11:46:33
state	T: 10.4 H: 74.6	2012-03-12 11:46:33
temperature	10.4	2012-03-12 11:46:33

Below the readings table is a dropdown menu with 'attr ez_Aussensensor' selected and 'Audio' as the current view. Below that is another table with device metadata:

fm_order	15	deleteattr
model	S300	deleteattr
room	Wohnung,Esszimmer	deleteattr

At the bottom of the main content area are two links: [Select icon](#) and [Device specific help](#).

Mehrere Geräte mit einem Klick schalten - structure

Zum Schalten von Gerätegruppen gibt es grundsätzlich drei Möglichkeiten:

- Die Einrichtung von Gruppen innerhalb des Hardwaresystems. Diese Gruppen sind auch von fhem aus bedienbar. Die Vorgehensweise ist in den jeweiligen Handbüchern beschrieben, siehe auch [Funktionsgruppen](#), [Lokale Master](#), [Globale Master](#)
- Die Zuordnung der zu gruppierenden Geräte zu einem Makro (siehe `notify` später in diesem Dokument)
- Die Einrichtung von Gruppen innerhalb fhem. Dieser Weg ist in diesem Kapitel beschrieben.

Wie bereits erwähnt liegt der wesentliche Unterschied im erzeugten Funkverkehr: Während für das Schalten einer im Hardwaresystem gebildeten Gruppe das Senden nur eines Funktelegramms erfordert, wird beim Schalten einer in fhem gebildeten Gruppe ein Funktelegramm je Gerät gesendet.

Um alle Geräte einer Gruppe (also z.B. alle Geräte im Wohnzimmer) mit nur einem Klick gemeinsam schalten können zu können, legen Sie in der fhem.cfg die Gruppe fest:

```
define wz_LichtAlle structure room wz_Lampe wz_MediaServer
```

Nach dem Speichern wird nun auch die Struktur wz_LichtAlle mit ihren ON und OFF-Schaltern angezeigt und ist bedienbar. Wenn Sie auch diese Struktur dem Raum Wohnzimmer zuweisen, wird die Struktur auch in diesem Raum angezeigt:

```
attr wz_LichtAlle room Wohnzimmer
```

Dabei wird diese Festlegung nun auf alle der Struktur zugehörigen Objekte vererbt (wz_Lampe und wz_MediaServer werden ebenfalls dem Raum Wohnzimmer zugeordnet).

Details zur Einrichtung finden Sie [hier](#).

Timer

Viele Aktoren beherrschen auch eine timer-Funktion, so z.B. die Funksteckdose FS20-st oder der Dimmer FS20-di.

In fhem stehen hierfür mehrere Befehle zur Verfügung. Ein Beispiel ist

```
set lamp1 on-for-timer 10
```

(lamp1 für 10 Sekunden einschalten, dann wieder ausschalten). Analog gibt es den Befehl off-for-timer.

Die Zeitangaben für timer-Befehle akzeptieren jeglichen ganzzahligen Wert in Sekunden. Der timer verfügt leider nur über [112 Werte](#), mit denen sich in wachsender Abstufung Zeiträume von 0,25 Sekunden bis 4,25 Stunden abbilden lassen. (Die Auflösung beträgt 0,25s bei 0 bis 4s, 0,5s von 4 bis 8s, 1s von 8 bis 16s usw. Für bessere Genauigkeit bei großen Werten verwenden Sie besser `at`) Ggf. wird Ihr Wert automatisch umgesetzt auf den nächsten Passenden Wert – in diesem Fall erscheint eine Meldung im Log.

Da der Timer innerhalb des Aktors abgearbeitet wird, und der letzte an ihn gesendete Befehl ein ‚on-for-timer‘ ist, würde nun das Glühbirnen-Symbol im Webfrontend dauerhaft auf ON stehenbleiben. Um dies zu verhindern gibt es das Attribut `follow-on-for-timer`. Damit aktivieren sie diesen Modus für Ihr Gerät. Setzen Sie also z.B.

```
attr lamp1 follow-on-for-timer 1
```

(die 1 steht in diesem Fall nicht für eine Sekunde, sondern für „aktiv“)

Details und weitere Beispiele sind in der [fhem-Referenz](#) und [im Wiki](#) beschrieben.

Dimmer

Als spezielle Aktoren gibt es auch Dimmer, z.B. FS20-di. Diese lassen sich wie Schalter mit on/off/toggle bedienen, beherrschen aber einige zusätzliche Befehle: `dimup`, `dimdown`, `dimupdown` sowie die Einstellung auf vorgegebene Dimmstufen, z.B. `dim50%`. Die vollständige Liste der 16 Dimmstufen finden Sie in der [fhem-Referenz](#).

Wenn Sie an einem Sensor (Schalter) den Taster länger als 0,4 Sekunden gedrückt halten, sendet dieser statt on/off/toggle den Befehl `dimup/dimdown/dimupdown`. So lassen sich wiederum ohne Programmierung auch Dimm-Vorgänge erreichen.

Ein Dimmen aus dem fhem-Webfrontend ist über eingetippte Befehle möglich, Sie können also z.B. `set lamp1 dimup` oder `set lamp1 dim50%` eingeben.

Eine besondere Variante via fhem ist die Einstellung eines gleichmäßigen Dimm-Vorgangs über einen Zeitraum. So kann z.B. die Nachtschlampe zum wakeup-light werden, wenn sie über ca. 20 Minuten (1280 Sekunden) langsam heller wird:

```
set lampe1 dim100% 1280
```

Die Funktionen des Wakeuplight lassen sich beliebig erweitern. So kann man den o.g. Befehl z.B. durch einen at-Befehl (siehe nächstes Kapitel) auslösen lassen und über eines der Multimedia-Module (siehe ebenfalls weiter unten in diesem Dokument) mit der Wiedergabe von Musik und dem Hochfahren der Rolläden kombinieren.

Schalten zu bestimmten Zeitpunkten – at

Ein häufiger Anwendungsfall ist das Schalten von Geräten zu festgelegten, sich ggf. wiederholenden Zeitpunkten. Dazu steht in fhem der Befehl `define...at` zur Verfügung:

```
define <name> at <timespec> <command> also z. B.
```

```
define LampeAnUm1700 at 17:00:00 set lamp on
```

(Um eine solche „Zeitschaltuhr“ anzulegen, tippen Sie den o.g. Befehl einfach im fhem webfrontend in das Kommandozeilen-Feld ein.)

LampeAnUm1700 ist hierbei lediglich ein Name/Platzhalter, unter dem diese „Zeitschaltuhr“ in fhem gespeichert wird und später wiedergefunden werden kann. Sobald Sie den oben genannten Befehl eingeben, erscheint dieses geplante Ereignis unter ‚Everything‘ im Abschnitt ‚at‘. Ist die angegebene Uhrzeit erreicht, wird die Anweisung genau einmal ausgeführt. Damit ist das Ereignis erfolgreich abgearbeitet und wird gelöscht.

Auch Wiederholungen sind möglich, z.B. täglich um 17:00. Dazu wird der Uhrzeit ein * vorangestellt:

```
define LampeTaeglichAn at *17:00:00 set lamp on
```

Der **modify**-Befehl dient dazu, den Zeitpunkt zu ändern, ohne den Befehls-Teil erneut angeben zu müssen:

```
modify LampeTaeglichAn *17:30:00
```

Über die absolute Angabe von Schaltzeiten (um 17:00) hinaus erlaubt der at-Befehl auch relative Angaben. Dies wird durch ein vorangestelltes Plus-Zeichen erreicht. Also „in 10 Minuten“:

```
define a5 at +00:10:00 set lamp on
```

Die Kombination von + und * bewirkt dann z.B. „alle 10 Minuten“, hier für 30 Sekunden:

```
define a6 at +*00:10:00 set lamp on-for-timer 30
```

Für solche Durchläufe kann auch die gewünschte Anzahl der Wiederholungen in geschweiften Klammern angegeben werden:

```
define a7 at +*{3}00:00:02 set lamp on-for-timer 1 # drei mal blinken
```

(drei mal alle 2 Sekunden für 1 Sekunde einschalten).

In der fhem-Referenz sind hierzu viele Beispiele genannt, die auch Abhängigkeiten von Wochenenden, Sonnenauf- und -untergängen aufzeigen – dies erfordert einfache Programmierung und wird z.T. später in diesem Dokument aufgezeigt. Diese Infos finden Sie [hier](#).

Schalten von Ereignissen abhängig machen - notify

Vor allem bei indirekten Schaltungen möchten Sie erreichen, dass ein Ereignis (z.B. Drücken einer Taste) das Schalten eines Aktors nach sich zieht. Die Kopplung erfolgt so:

```
define <name> notify <pattern> <command>
```

also z.B.

```
define Schalter1Notify notify Schalter1 set wz_Media on
```

Der bei `define...notify` angegebene Name bezeichnet den ‚Event-Handler‘. Die besondere Eigenschaft dieser `notify`-Anweisungen ist, dass sie nicht zum Zeitpunkt der Eingabe oder zu einer festgelegten Uhrzeit ausgeführt werden, sondern im Hintergrund ‚mitlauschen‘. Sobald der nach `notify` angegebene Sensor (Schalter1) einen Funkbefehl übermittelt, wird die im `notify` angegebene Anweisung ausgeführt. Die oben dargestellte Zeile schaltet also `wz_Media` an, sobald der Taster Schalter1 betätigt wird. Häufig möchten Sie mit nur einem Klick mehrere Geräte schalten. In diesem Fall können Sie als `<command>` auch eine Liste von Geräten angeben (siehe `devspec` beim Befehl `set`) oder eine Struktur schalten (siehe `structure`).

Allerdings ist das o.g. Beispiel -so wie es da steht- ein Anwendungsfall vornehmlich für die Einstellung als 4-Kanal-Schalter, also Tastern: das Gerät soll beim Betätigen des Schalter1 immer eingeschaltet werden. Für das Ausschalten würde man also analog Schalter 2 mit dem `off`-Befehl koppeln:

```
define Schalter2Notify notify Schalter2 set wz_Media off
```

Ist Ihr Schalter als 2-Kanal definiert, können Sie herausfiltern, ob der `on`- oder der `off`-button gedrückt wurde:

```
define Schalter1NotifyOn notify Schalter1:on set wz_Media on
```

oder Sie schalten das Licht aus, wenn Media eingeschaltet wird – und umgekehrt:

```
define Schalter1NotifyAn  notify Schalter1:on  set wz_Media on;;set wz_Licht off
define Schalter1NotifyAus notify Schalter1:off set wz_Media off;;set wz_Licht on
```

Hinweis: Bei Aufzählungen keine Leerstellen vor und nach den Semikola!

Als kürzere Alternative zu

```
define Schalter1NotifyOn  notify Schalter1:on set wz_Media on
define Schalter1NotifyOff notify Schalter1:off set wz_Media off
```

verwenden Sie die Variable `$EVENT`:

```
define Schalter1Notify notify Schalter1 set wz_Media $EVENT
define Schalter1Notify notify Schalter1 set wz_Media %           (alte Schreibweise)
```

Dabei hat `$EVENT` genau den Wert, der vom abgefragten Sensor (also Schalter1) gesendet wurde. Sendet also Schalter1 den Wert `on`, so wird der Befehl `set wz_Media on` ausgeführt. Sendet der Schalter1 den Befehl `off`, so wird `set wz_Media off` ausgeführt. Analog funktioniert das für jeden Befehl des Sensors, also auch `toggle`, `dimup`, `dimdown` etc.

Eine Beschreibung mit vielen Beispielen gibt's im [fhemWiki](#).

Verwendung von `notify` als Makro

Als Makro bezeichnet man das Ausführen mehrerer Befehle auf Grund nur eines auslösenden Ereignisses (also z.B. dem Drücken einer Taste, dem Klick auf einen Weblink o.ä.). In diesem Sinne lässt sich `notify` als Makro verstehen, wenn Sie wie oben als `<command>` mehrere Befehle – durch Semikola getrennt- angeben:

```
define Schalter1NotifyAn  notify Schalter1:on  set wz_Media on;;set wz_Licht off
```

Das Abarbeiten eines `notify` (Makros) wird gestartet, sobald das `<pattern>` eintritt.

Starten eines Makros – trigger

Ein solches Ereignis muss nicht durch ein Funktelegramm ausgelöst werden, sondern kann auch durch den Befehl `trigger` gestartet werden.

Wenn z.B. abends in jedem Raum eine Lampe eingeschaltet werden soll:

```
define Abends notify Abends set wz_LampeKlein on;;set sz_Stehlampe on;;set
ku_Downlight on
```

können Sie den Vorgang starten durch

```
trigger Abends
```

Diese Möglichkeit ist besonders zum Testen neuer notify-Makros hilfreich. Eine umfangreiche Darstellung der Möglichkeiten von notify finden Sie in [diesem Wiki-Artikel](#).

Bearbeiten über das Webfrontend

Wenn Ihre notify-Definition bzw. Ihr Makro einmal erstellt ist, können Sie es im Webfrontend bearbeiten. Klicken Sie dazu im Menü auf „Everything“ und scrollen dann zum Abschnitt notify. Durch Klicken auf den Namen des notify gelangen Sie in den Detail-Bildschirm, den Sie ja von Geräten bereits kennen:

The screenshot shows the fhem web frontend interface. On the left is a sidebar with a 'Fhem' logo and a menu including 'Floorplans', 'CUL_WS', 'Esszimmer', 'Korridor', 'Kueche', 'Media', 'Plots', 'Schlafzimmer', 'Unsorted', 'Wohnung', 'Wohnzimmer', 'Everything' (selected), 'Howto', 'Wiki', 'Details', 'Edit files', 'Select style', and 'Event monitor'. The main content area has a 'CFGFN' tab and a 'DEF' section (circled in red) containing the macro definition: 'Abends set wz_LampeKlein on;set sz_Stehlampe on;set ku_Downlight on'. Below the definition is a 'modify Abends' button. A table displays the macro's configuration: NAME (Abends), NR (161751), NTFY_ORDER (50-Abends), REGEXP (Abends), STATE (active), and TYPE (notify). At the bottom, there is an 'attr' field with 'Abends', an 'Audio' dropdown, and links for 'Select icon' and 'Device specific help'.

Wenn Sie auf DEF klicken, können Sie die Definition Ihres notify in einem Textfeld bearbeiten – z.B. wenn Sie ein weiteres Gerät zu Ihrem Makro hinzufügen möchten.

Hinweis: Im DEF-Textfeld erscheinen nur <pattern> und <command>. Der Name des notify steht unterhalb des Textfensters.

Beenden Sie die Bearbeitung mit Klick auf die Schaltfläche modify <name> .

Hinweis: Ihre Änderung wird erst dauerhaft in die Konfigurationsdatei fhem.cfg übernommen, wenn Sie nach dem Klick auf modify <name> über das Kommandofeld den Befehl save ausführen.

Pairing: direkt oder indirekt

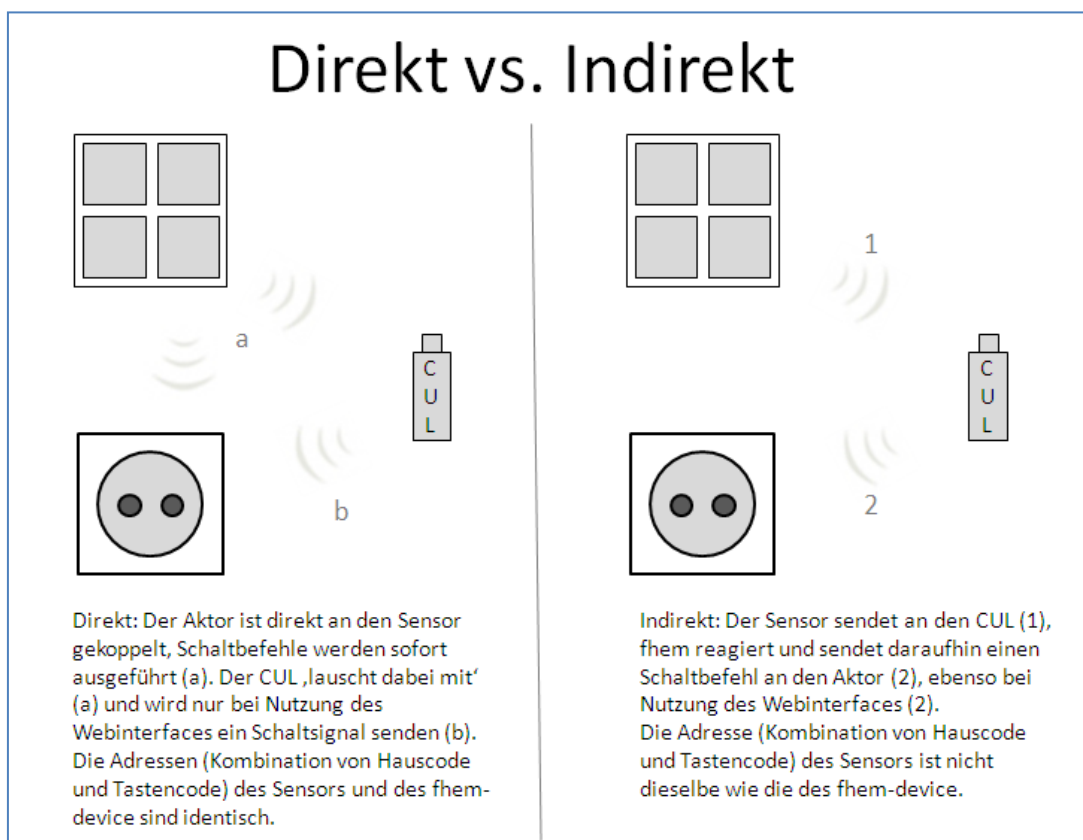
In der bisher beschriebenen Konfiguration ist jeder Aktor direkt mit einem Sensor gekoppelt.

Nachdem Sie nun notify kennengelernt haben, können Sie auch eine „indirekte“ Kopplung vornehmen.

Eine direkte Kopplung (Pairing) hat den Vorteil, dass die Steuerung der Aktoren direkt vom Sensor vorgenommen wird und fhem zumeist nur ‚mitlauscht‘. Sobald also der Sensor – z.B. nach einem Tastendruck auf den Wandschalter oder dem Auslösen eines Bewegungssensors - ein Funktelegramm sendet, wird der Aktor direkt durch das vom Sensor gesendete Signal angesteuert und reagiert sofort. Eine Zeitverzögerung z.B. wegen Auslastung oder gar Nicht-Verfügbarkeit des fhem-Systems tritt nicht auf . Das ist durchaus wichtig, denn schon eine Zeitverzögerung von einer halben Sekunde zwischen Drücken des Lichtschalters und Einschalten des Lichts kann sehr irritierend wirken, da man gewohnt ist,

dass das Licht nach Betätigen des Lichtschalters eben sofort angeht. Geschieht dies nicht, vermuten viele einen Fehler und drücken den Schalter gleich noch mal...

Im Rahmen der Hausautomatisierung gibt es aber häufig den Fall, dass nach dem Drücken eines Schalters erst noch geprüft werden soll, welche Aktion nun auszuführen ist – ist es vielleicht Nacht und es soll statt des grellen Korridorlichts nur eine kleinere Lampe eingeschaltet werden? Daher gibt es auch die Alternative, den Aktor separat vom Sensor zu definieren – also nicht durch das Anlernen auf Basis eines Sensor-Funktelegramms. Sensor und Aktor verwenden in diesem Fall NICHT DIE SELBE Kombination von Hauscode und Tastencode; vielmehr wird in fhem ein device angelegt mit einer ANDEREN Adresse (Kombination Hauscode+Tastencode) als der des Sensors. Auf diese separate fhem-Adresse wird der Aktor angelernt. Ein Funktelegramm eines Sensors (Lichtschalter) wird dann also von keinem Aktor umgesetzt, da kein Aktor auf dieses Funktelegramm angelernt wurde. CUL+fhem ‚hören‘ das Funktelegramm (und lernen –falls noch nicht geschehen– den Sensor über autocreate an). In fhem kann man nun mittels `notify` eine beliebige Reaktion auf dieses Funktelegramm, also auf das eingetretene Ereignis (das Drücken einer Taste oder das Auslösen eines Bewegungsmelders) definieren. Je nach im `notify` definierter resultierender Aktion sendet fhem ggf. ein separates Funktelegramm mit einer anderen Adresse, auf die ein Aktor angelernt ist und seinerseits reagiert. Ähnliches gilt, wenn ein Aktor gar nicht von einem physischen Schalter, sondern ausschließlich aus dem Web-Frontend getriggert (engl. „ausgelöst“) werden soll: Eine bestimmte Kombination von Hauscode und Tastencode wird an keinem Sensor eingestellt, sondern nur in einem fhem-device definiert, um dann einen Aktor auf das fhem-Funktelegramm anzulernen, das nach Klick auf ‚on‘ oder ‚off‘ im webfrontend vom CUL gesendet wird.



Für indirektes Pairing muss ein Aktor ‚manuell‘ in fhem eingerichtet werden. Hierfür wird der Befehl `define...FS20` verwendet:

```
define <devicename> FS20 <Hauscode> <Tastencode> [fg][lm][gm] , also z.B.  
define lampe1 FS20 12341234 01
```

Als Hauscode verwenden Sie hier

- Denselben Hauscode wie für Ihr Hardwaresystem gewählt, wenn Sie Schaltgruppen (Funktionsgruppe (fg), Lokaler Master (lm) oder Globaler Master (gm)) über die Adressierung des Hardwaresystems nutzen möchten
- Einen separaten Hauscode für alle ‚nur-fhem-devices‘, wenn eine Einbindung des devices in Gruppenschaltungen des Hardwaresystems nicht vorgesehen ist. Gruppen können in fhem mittels `structure` oder `notify` über unterschiedliche Hauscodes hinweg definiert werden, das Schalten von Gruppen erzeugt dann aber mehr ‚Funklast‘.

Außerdem ist beim indirekten Pairing eine fhem-‚Kopplung‘ zwischen Sensor und Aktor erforderlich, da fhem zunächst nicht wissen kann, welcher Sensor welchen Aktor schalten soll. Für diese ‚Kopplung‘ wird der Befehl `define...notify` verwendet. Wenn also z.B. durch das Drücken von Schalter1 der Aktor `lampe1` eingeschaltet werden soll, so wird dies erreicht durch

```
define <name> notify <pattern> <command> , also z.B.  
define S1notify notify Schalter1 set lampe1 on
```

In diesem Beispiel waren Sensor und Aktor aus demselben Hardwaresystem FS20. Der Aktor `lampe1` kann aber ebenso gut aus einem anderen Hardwaresystem stammen – man kann also z.B. einen FS20-Wandschalter als Sensor verwenden, und dadurch über `notify` einen HomeMatic- oder 1-wire-Aktor schalten. Ebenso funktioniert es natürlich auch umgekehrt – ein EnOcean-Sensor kann eine FS20- oder Intertechno-Steckdose schalten.

Anpassen der Darstellung im Webfrontend - FHEMWEB-Attribute

Das fhem Webfrontend ist flexibel und funktional. Es gestattet, die Darstellung den eigenen Bedürfnissen anzupassen. Das Webfrontend ist in fhem über das (Pseudo-)Gerät FHEMWEB realisiert. Wie in der Liste der Geräte (Raum „Everything“) zu sehen ist, können Sie mehrere Geräte vom Typ FHEMWEB definieren und diesen Attribute zuordnen. Eine vollständige Liste aller Attribute steht in der [comandref hier](#).

Ein Konfigurationsbeispiel mit screenshot und zugeordneten Attributen gibt's auf Seite 36.

Bildschirmgröße und "Skin": stylesheetPrefix

Der **Port 8083** wird dem Gerät WEB zugeordnet. Dieser ist erreichbar unter <http://<ip>:8083/fhem>

```
define WEB FHEMWEB 8083 global
```

erzeugt eine für PC-Bildschirme angepasste Darstellung. Den gewünschten "Skin" können Sie unter dem Menüpunkt "Select Style" auswählen. Jeder Style ist die Kombination des Farbschemas (default, dark, iOS7) und der Optimierung auf die jeweilige Bildschirmgröße (default, smallscreen, touchpad).

Um eine Darstellung mit dunklem Hintergrund zu erzeugen, wählen Sie unter "Select style" aus der Liste "dark" aus – dadurch wird im Hintergrund gesetzt

```
attr WEB stylesheetPrefix dark
```

und dadurch zur Formatierung `darkstyle.css` verwendet.



Port 8084 (WEBphone) ist das Attribut `stylesheetPrefix smallscreen` zugeordnet. Dadurch wird die Darstellung für Geräte mit kleinem Bildschirm angepasst. Die Liste der Räume erscheint dafür nicht mehr am linken Bildschirmrand, sondern als Drop-Down-Liste oben.

```
define WEBphone FHEMWEB 8084 global
```

```
attr WEBphone stylesheetPrefix smallscreen
```

Im Hintergrund wird hier zur Formatierung die Datei

`smallscreenstyle.css` verwendet. Analog existieren auch `darksmallscreen.css` und `iOS7smallscreen.css`.

Port 8085 (WEBtablet) trägt den prefix `touchpad`. Hierdurch werden Schriftgröße und Darstellungsart für Tablet-PCs optimiert.

```
define WEBtablet FHEMWEB 8085 global
```

```
attr WEBtablet stylesheetPrefix touchpad
```

Besonderheit: für iPhone und iPad ist die Darstellung in `smallscreen` und `touchpad` so gestaltet, dass das Webfrontend wie eine Fullscreen-„App“ genutzt werden kann. Dazu öffnen Sie fhem in Safari mit dem entsprechenden port (also z.B. <http://<ip>:8085/fhem>) . Klicken Sie auf den Optionen-button  und wählen Sie „Zum Homebildschirm“. Dadurch erhalten Sie ein neues Icon, mit dem Sie diese fhem-Seite mit nur einem Klick starten können. Die Leiste für URL und Websuche werden dann nicht angezeigt, so dass sich fhem verhält wie eine vollwertige App.

Weniger Menüpunkte: hiddenroom

Es besteht außerdem die Möglichkeit, aus dem Standard Webfrontend einige Menüpunkte aus dem unteren Menüblock auszublenden.

```
attr WEB hiddenroom save,Howto,FAQ,Examples,Unsorted,"Select style"
```

Ebenso kann man den Zugang zum Detailview der Geräte blockieren (`detail`), das Kommandofeld unterdrücken (`input`) und den save-Button ausblenden (`save`). Dies kann mit dem Attribut `hiddenroom` erreicht werden.

Hinweis: Verwenden Sie eine ältere fhem-Version, ist insbesondere das Ausblenden des "save"-Buttons für Anfänger empfohlen, da er durch seine Position direkt neben dem Kommando-Eingabefeld immer wieder zu Verwirrung führt. Mit diesem Button wird nicht die Eingabe eines Befehls bestätigt (dazu reicht das drücken der <ENTER>-Taste), sondern es wird die aktuelle Konfiguration gespeichert, damit sie nach dem nächsten Neustart von fhem wieder verfügbar ist. Nach dem Ausblenden dieses Buttons kann das Speichern dadurch erfolgen, dass man den Befehl `save` in das Kommandofeld eingibt und <ENTER> drückt (oder bei neueren Versionen im fhem-Menü auf "save config" klickt).

Zusätzliche Menüpunkte – menuEntries



Auch ist es möglich, dem fhem-Menü eigene Menüpunkte hinzuzufügen durch Verwendung des Attributs `menuEntries`. Hinterlegt wird eine komma-getrennte Liste von Paaren; zunächst die Bezeichnung wie sie auf dem Bildschirm erscheinen soll (z.B. `restart`) gefolgt von dem auszuführenden Befehl beginnend mit `cmd=` und dann dem fhem-Kommando. Leerstellen im Kommando werden dabei mit `+` aufgefüllt (z.B. `cmd=shutdown+restart`) :

```
attr WEB menuEntries
restart,cmd=shutdown+restart,update,cmd=update,updatecheck,cmd=update+check,reloadMyUtils,cmd=reload+99_myUtils.pm
```

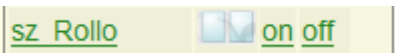
Nur bestimmte Befehle für ein Gerät - webCmd

Das Attribut `webCmd` beeinflusst die Anzahl der im webfrontend angezeigten Befehle. Ein FS20-Gerät hat z.B. ca. 25 mögliche Befehle (`on`, `off`, `dimup`, `dimX%`, `toggle`, `on-for-timer`, ...). Mit `webCmd` wählen Sie aus, welche davon im webfrontend angezeigt werden (wenn Sie `webCmd` nicht angeben, ist der default `on,off`).

```
attr Lampe webCmd on:off:on-for-timer 600
```

Befehle umbenennen - eventMap

Mit `eventmap` können Sie den für das webfrontend verwendeten Text eines Kommandos bzw. Schaltzustandes anpassen (dem Gerät `sz_Rollo` wurden eigene Icons zugeordnet, deshalb sehen Sie hier nicht die gewohnte Glühlampe, sondern ein offenes Fenster – siehe nächster Abschnitt)

Vorher: 

```
attr sz_Rollo eventMap on:Down off:Up
attr sz_Rollo webCmd Down:Up
```

Nachher: 

Räume sortieren: sortRooms

Standardmäßig werden die Räume in alphabetischer Sortierung angezeigt. Dies kann mit dem Attribut `sortRooms` geändert werden:

```
attr WEB sortRooms Wohnzimmer Schlafzimmer Media Wetter Heizung System
```

Gruppen bilden und umbenennen: group

Standardmäßig werden in fhem Geräte nach ihrem TYPE sortiert, so gibt es Gruppen wie FS20, FHT usw. Möchten Sie eigene Gruppen bilden, funktioniert das mit dem Attribut [group](#), das sich ähnlich wie room verwenden lässt. Eine Gruppe muss also nicht separat definiert werden, sondern entsteht dadurch, dass sie dem ersten Gerät zugeordnet wird.

```
attr ez_FHT group Heizung
attr ez_Aussensensor group Heizung
```

erzeugt also eine neue Gruppe "Heizung", die dann als separater Block in einem Raum dargestellt wird.

Mehrspaltige Darstellung: column

Standardmäßig erfolgt die Darstellung im Webfrontend einspaltig, es werden also alle Gruppen untereinander dargestellt. Um eine mehrspaltige Darstellung zu erreichen, legen Sie je Raum fest, welche Gruppen untereinander (getrennt durch Komma) bzw. nebeneinander (getrennt durch |) angeordnet werden. Das Attribut wird der Webinstanz zugeordnet:

```
attr WEB column Übersicht:Wohnung,Heizung,iTunes|weather
```

Durch Setzen dieses Attributs werden also im Raum "Übersicht" die Gruppen Wohnung, Heizung und iTunes untereinander dargestellt, in einer Spalte rechts daneben erscheint die Gruppe weather. Die Liste kann natürlich für weitere Räume erweitert werden.

Iconaktualisierung ohne browser-refresh – longpoll

Während man einen fhem-Raum im Browser anzeigt, kann ja eine Änderung des Schaltzustands eintreten, wenn z.B. ein Sensor auslöst oder ein Lichtschalter betätigt wird. In fhem wird dieser Schaltzustand in der Darstellung aktualisiert, ohne dass im Browser der „refresh“-button betätigt werden muss.

Möchte man dies nicht, kann man diesen sogenannten „longpoll“-Mechanismus für die Webinstanz deaktivieren:

```
attr WEB longpoll 0
```

Zur Anpassung der Darstellung im Webfrontend stehen weitere Attribute zur Verfügung. Eine vollständige Liste der Möglichkeiten findet sich in der [commandref](#).

Welche Icons? stylesheetPrefix und iconPath

Zusammen mit fhem werden mehrere Icon-Bibliotheken ausgeliefert. Sie sind abgelegt unterhalb des Ordners fhem/www/images. Der Ordner default enthält die fhem Standard-Icons. In den Ordnern dark, smallscreen und tablet sind Icons abgelegt, die speziell zu der jeweiligen Darstellungsweise passen. Die Suchreihenfolge hängt dabei vom verwendeten stylesheetPrefix (siehe oben) ab: ist z.B. stylesheetPrefix dark gesetzt, werden Icons zuerst im Ordner dark gesucht. Werden sie dort nicht gefunden, wird die Suche im Ordner default fortgesetzt.

Der Ordner openautomation enthält eine Icon-Bibliothek des gleichnamigen Projekts. Diese Icons sind in SVG erstellt und lassen dadurch Größenskalierung und das Setzen von Farben zu. Die Liste ist sehr umfangreich, jedoch unterscheidet sich das Layout dieser Icons stark von dem des fhem Standard.

Welche Icons verwendet werden sollen, kann mittels [iconPath](#) eingestellt werden:

```
attr WEB iconPath default:openautomation
```

Ist die default-Einstellung, solange dieses Attribut nicht gesetzt ist. Mit

```
attr WEB iconPath openautomation:default
```

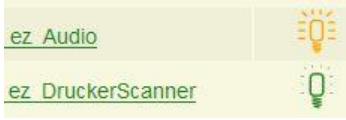
wird die Suchreihenfolge umgekehrt: wo vorhanden, werden die SVG-icons aus dem Ordner openautomation verwendet.

Vorher:



```
attr WEB iconPath openautomation:default
```

Nachher:



Probieren Sie es doch einfach mal aus.

Hinweis: iOS 7 gestattet das Aktualisieren von jpg/png-icons mittels longpoll leider nicht. Möchten Sie also longpoll verwenden, stellen Sie auf die Darstellung mit openautomation-icons um.

Eigene Icons

fhem sieht außerdem die Möglichkeit vor, den Geräten andere bzw. eigene Icons zuzuweisen. Dabei kann grundsätzlich jedes Bild verwendet werden, es empfiehlt sich jedoch, sich an der Größe der bereits vorhandenen Icons zu orientieren. Legen Sie eigene Icons in den geeigneten Ordnern ab, siehe vorheriger Abschnitt.

Icon vor dem Gerätenamen - icon

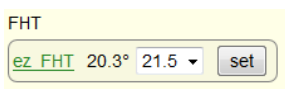
Klicken Sie in der Detail-Ansicht eines Geräts auf den Link ‚Select icon‘. Sobald Sie ein Icon auswählen, wird dem Gerät ein Attribut ‚icon‘ mit dem entsprechenden Dateinamen als Wert zugeordnet. Alternativ können Sie dieses Attribut natürlich auch manuell eingeben – oder es löschen, wenn kein Icon mehr verwendet werden soll.

Für die Liste der verfügbaren Icons werden von fhem alle Dateien mit dem Prefix `ico` angezeigt – benennen Sie Ihre eigenen Icon-Datei also `ico*.jpg` oder `ico*.png`.

Icon-Bilddateien liegen im Icon-directory `fhem/www/images/default`.

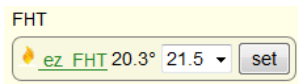
Hier ein Beispiel mit und ohne Icon vor dem Gerätenamen:

Vorher:



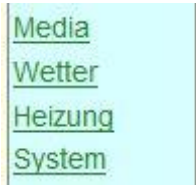
```
attr ez_FHT icon icoHEIZUNG.png
```

Nachher:



Icon vor dem Raumnamen - roomIcons

Auch vor einem Raumnamen im fhem-Menü kann ein icon angezeigt werden. Die Zuordnung erfolgt über das FHEMWEB-Attribut `roomIcons`: Angegeben werden durch `:` verbundene Paare von Raumname und icon, die Paare sind durch Leerstellen getrennt:



```
attr WEB roomIcons Media:audio_eq
Wetter:weather_cloudy Heizung:sani_heating_temp
System:edit_settings
```

Hinweis: Das ist eigentlich eine Zeile, zur korrekten Darstellung müsste die Schrift aber so klein eingestellt werden, dass man sie nicht mehr lesen könnte ;-)



Eigene Icons für Geräte-Schaltzustände

Auch ist es möglich, an Stelle der grauen bzw. leuchtenden Glühlampe andere Icons anzuzeigen. Eine einfache Vorgehensweise ist, ein bestehendes Icon (z.B. FS20.ON.PNG) zu kopieren und die Kopie dann zu bearbeiten. Bitte beachten Sie, dass die Dateinamen 100%ig zu den Gerätenamen und Schaltzuständen passen müssen – auch Groß- und Kleinschreibung wird unterschieden.

Auch die hierfür verwendeten Dateien werden in den Ordnern unterhalb fhem/www/images abgelegt.

Die Namenskonvention ist:

<devicename>.<state>.[jpg|png|gif]

Beispiel: sz_Rollo.on.jpg , sz_Rollo.off.jpg,

Werden diese Dateien nicht gefunden, verwendet fhem die Standard-Icons

<type>.<state>.[jpg|png|gif] , also z.B. FS20.on.png , FS20.off.png

Die Suchreihenfolge ist dabei wie folgt:

- Suche nach einem Icon, das zum Gerätenamen passt
- Wird keines gefunden, wird ein Icon für den Geräte-TYPE gesucht (FS20, FHT, ...)

Jeder dieser Suchläufe erfolgt dabei auf Basis der Ordnerstruktur unterhalb des Dateipfads fhem/www/images .

- Eine Besonderheit der Suchreihenfolge ist die Berücksichtigung von stylesheetPrefix: ist hier z.B. "dark" eingestellt, wird zunächst im Ordner dark gesucht, ob hier spezielle Icons zur Darstellung vor dunklem Hintergrund vorhanden sind. Wird dort kein Icon gefunden, wird die Suche gemäß iconPath fortgesetzt. Die Voreinstellung hierfür ist eine Suche zunächst im Ordner default, danach im Ordner openautomation.
- Wird kein icon gefunden, wird der aktuelle Schaltzustand nicht als Icon, sondern als Wort dargestellt.

Weblink

In das fhem-webfrontend können Hyperlinks und Web-Grafiken eingebunden werden. Hierzu ist das (Pseudo-)device `weblink` verfügbar. Details und Beispiele finden Sie in der commandref [hier](#) .

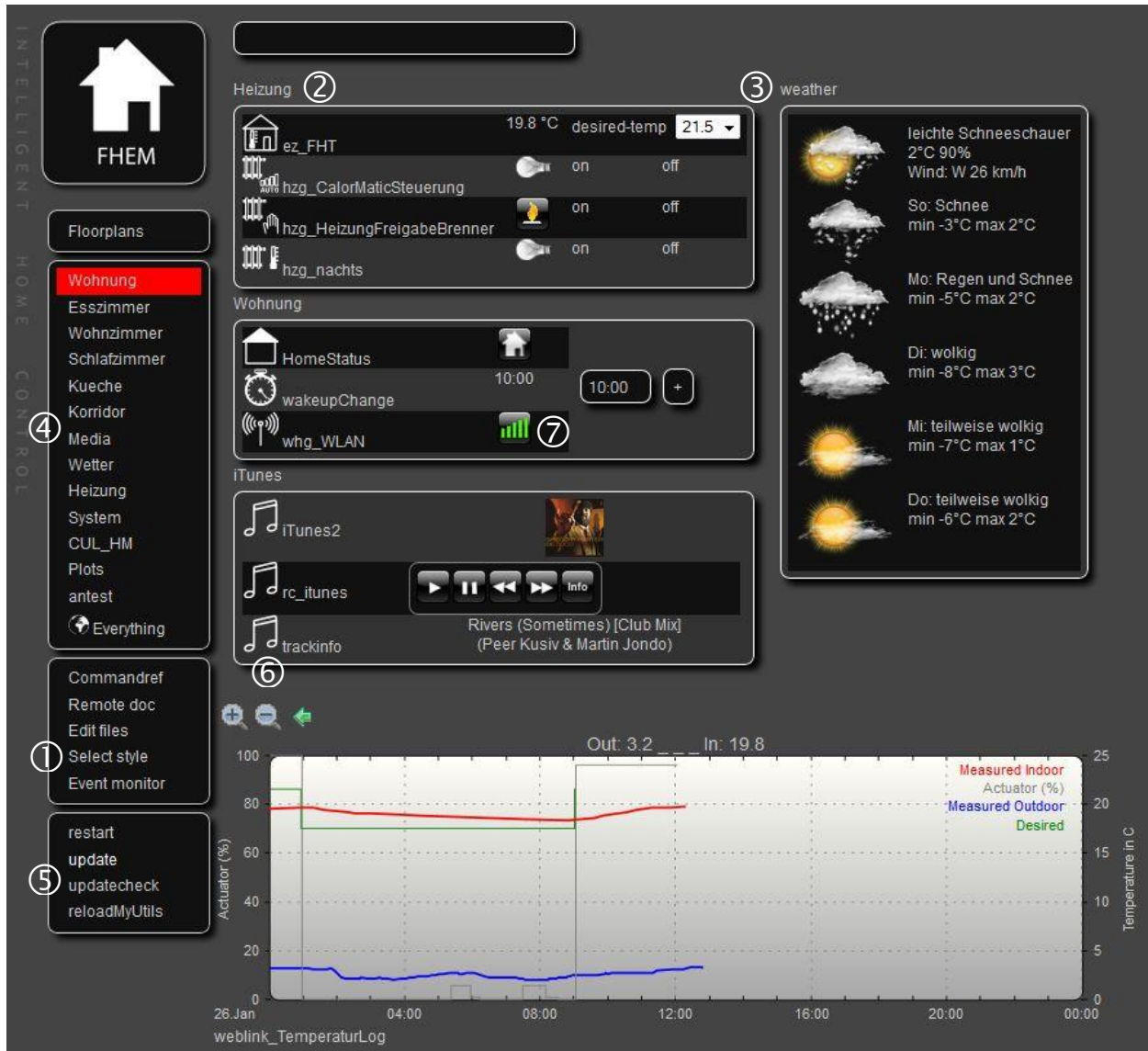
Ein Beispiel ist das Einblenden eines Wetterberichts (Achtung: EINE Zeile!):

```
define Wetter weblink iframe http://www.wetteronline.de/cgi-bin/hpweather?PLZ=12345
```

Zur Anzeige und Verwendung von Wetterdaten gibt es außerdem die fhem-Module [weather](#) und [GDS](#), siehe unter Wetterbericht einbinden .



Beispiel zur Konfiguration



①	Dunkler Hintergrund – Menüpunkt "Select style" -> dark (FHEMWEB-Attribut stylesheetPrefix)
②	Bildung eigener Gerätegruppen – Geräte-Attribut group
③	Mehrspaltige Anordnung – FHEMWEB-Attribut column
④	Sortierung der Raumlise – FHEMWEB-Attribut sortRooms Menüpunkte ausgeblendet (z.B. Unsorted) – FHEMWEB-Attribut hiddenroom
⑤	Eigene Befehle – FHEMWEB-Attribut menuEntries
⑥	Geräte-Icons – Geräte-Attribut icon
⑦	Zustands-Icons – durch Erzeugen von icon-Dateien passend zum Gerätenamen und -Schaltzustand, hier z.B. whg_WLAN.on.png

Gruppen frei auf dem Bildschirm platzieren: Dashboard

Das Modul [Dashboard](#) gestattet eine noch flexiblere Darstellung von Gruppen als das Attribut column. So können mittels Dashboard Gruppen in Tabs angeordnet und auf- und zugeklappt werden.

Einrichten eines Grundrisses mit eigenen fhem-Geräten

Einen Beispiel-Screenshot finden Sie auf dem Deckblatt dieses Dokuments. In fhem heißt so ein Grundriss FLOORPLAN, siehe dazu auch den [entsprechenden commandref-Eintrag](#). In diesem commandref-Eintrag finden Sie auch den Link auf einen detaillierten [Einrichtungs-Leitfaden](#).

Einfache Programmierung: if-Bedingung

Wir haben ja bereits gesehen, dass Schaltungen von bestimmten Kriterien abhängen können, also z.B. ob der on- oder off-Befehl gesendet wurde:

```
define Schalter1Notify notify Schalter1:off set wz_Media off
```

Auch die Variable \$EVENT haben wir bereits verwendet. Sie repräsentiert den aktuellen Schaltzustand:

```
define Schalter1Notify notify Schalter1 set wz_Media $EVENT
```

Da fhem in Perl programmiert ist, gibt es auch die Möglichkeit, Perl-code aus fhem heraus auszuführen. Oder andersherum: zusätzlich zu den fhem-eigenen Befehlen lassen sich auch Perl-Routinen zur Steuerung verwenden. Um unterscheiden zu können, ob fhem-Befehle oder Perl-Befehle verwendet werden, werden Perl-Kommandos immer in geschweifte Klammern gesetzt.

Auch aus Perl-Code heraus lassen sich fhem-Befehle absetzen. Daher bewirken folgende zwei Zeilen dasselbe:

```
set lamp off
{fhem("set lamp off")}
```

In der ersten Zeile ist der bereits bekannte fhem set-Befehl verwendet.

In der zweiten Zeile ist Perl-Code dargestellt – er ist in geschweifte Klammern eingebunden. In Perl gibt es den Befehl fhem, mit dem man aus Perl heraus fhem-Befehle ausführen kann. Als Parameter des Befehls fhem ist der auszuführende Befehls-String angegeben – also das, was Sie sonst in das fhem Kommandofeld eingeben würden.

Die Perl-Version des o.g. Codes können Sie nun zusammen mit einem notify oder at verwenden, oder Sie können das Perl-Kommando (incl. geschweiften Klammern!) ebenso in das fhem Kommandofeld eingeben. Durch die geschweiften Klammern weiß fhem, dass es sich um Perl-Code handelt.

```
define LampeAusUm9 at 09:00 {fhem("set lamp off")}
```

Der vermutlich am häufigsten verwendete Befehl ist der if-Befehl. Seine Syntax lautet:

```
{ if (Bedingung) {Befehl1} else {Befehl2} }
```

Das eingangs gezeigte Beispiel

```
define Schalter1Notify notify Schalter1:off set wz_Media off
```

kann man also ebenfalls schreiben als

```
define Schalter1Notify notify Schalter1 { if ( "$EVENT" eq "off" ) {fhem("set wz_Media off")} }
```

Wenn also die Variable \$EVENT den Wert off hat, wird in fhem der Befehl set wz_Media off ausgeführt.

Hinweis: In Codebeispielen im Wiki oder alten Forumsbeiträgen finden Sie eventuell noch die Variable % statt \$EVENT.

Achtung: Wenn Sie im Code Ihres notify anstatt der Variable % wirklich das Zeichen % verwenden möchten, schreiben Sie dies als %% bzw. analog \\ oder @@. Das sieht im Programmcode seltsam aus, zum Ausführungszeitpunkt wird aber in *einfache* Zeichen umgewandelt. Details und weitere Sonderfälle finden Sie in der commandref [hier](#) im Abschnitt „Notes“.

Tipp: Um ein **notify** oder **at** mit **Perl-Code** zu **erstellen**, gehen Sie in zwei Schritten vor:

1. Definieren Sie das notify/at in der Kommandozeile mit einem leeren Perl-code-Block, also z.B. `define Schalter1Notify notify Schalter1 {}`
2. Wechseln Sie nun über das Menü ‚Everything‘ in die Detail-Ansicht Ihres at/notify. Klicken Sie dort auf ‚DEF‘ um die Details Ihres at/notify zu bearbeiten. Fügen Sie erst dort den Perl-Code ein. Klicken Sie schließlich die Schaltfläche ‚modify <name>‘ zum Speichern Ihres Codes. Auf diese Weise ist die Bearbeitung am einfachsten und entspricht der, die bereits bei notify gezeigt wurde (siehe dort [Bearbeiten über das Webfrontend](#))

Hinweis: Die **Schreibweise** von Perl-code ist in der Detail-Ansicht ‚ganz normal‘, d.h. Sie können ganz normalen Perl-code verwenden. Nachdem Sie Ihre Änderungen gesichert haben und außerdem den Befehl `save` ausgeführt haben, schauen Sie sich das Ergebnis in `fhem.cfg` an. Sie werden feststellen, dass dort a) alle Semikola, Prozentzeichen und @ verdoppelt wurden und b) jeder Zeilenumbruch mit einem backslash (\) ‚geschützt‘ wurde. Ersparen Sie sich diese Schreibweise, indem Sie nicht direkt Ihre `fhem.cfg` bearbeiten, sondern die Detail-Ansicht Ihres at/notify zur Bearbeitung von Perl-code verwenden.

Tipp: Verwenden Sie einen Editor, bei dem Schlüsselwörter und vor allem zusammengehörende Klammerpaare ersichtlich sind. Der häufigste (Anfänger-)Fehler sind vergessene oder in falscher Reihenfolge gesetzte schließende Klammern! Verwenden Sie z.B. das kostenlose Notepad++ . Erst nachdem Sie in Notepad++ alle Klammern richtig gesetzt haben, kopieren Sie Ihr Miniprogramm nach `fhem` und hängen es z.B. wie oben beschrieben an ein at oder notify.

Tipp: Für Notepad++ gibt es die Möglichkeit, `fhem`-Befehle erkennen zu lassen. Damit lassen sich Tippfehler leichter finden. Installieren Sie dafür die zusätzliche Programmiersprache `fhem` und wählen Sie diese beim Bearbeiten der `fhem.cfg` oder Ihrer include-Datei aus.

Tipp: Um Ihren Perl-code zu **testen**, verwenden Sie das trigger-Kommando. Da dieses Kommando nur auf notify, jedoch nicht auf at wirkt, entwickeln und testen Sie Ihren Perl-code immer mit einem notify, auch wenn Sie ihn letztendlich einem at zuordnen möchten – das hinüberkopieren ist ja einfach. Wenn Sie also z.B. ein `define Testcode notify Schalter1:on {Perl-code}` testen möchten, können Sie Ihren Code mit ‚trigger Schalter1 on‘ testen. trigger simuliert das Eintreten eines Ereignisses (also das Eintreffen eines Funktelegramms von Schalter1 mit dem Befehl on).

Gerätewerte in Bedingungen – Value, ReadingsVal, ReadingsTimestamp

Häufig sollen Schaltvorgänge abhängig gemacht werden. Für das Gerät, das ein notify auslöst, steht wie oben dargestellt die Variable `$EVENT` zur Verfügung.

Sollen jedoch Werte anderer Geräte mit einbezogen werden, kommen die Funktionen `Value()` und `ReadingsVal` zum Einsatz.

Das o.g. Beispiel kann man auch so schreiben:

```
define Schalter1Notify notify Schalter1 { if ( Value("Schalter1") eq "off") {fhem("set
wz_Media off")}} }
```

Die Variable \$EVENT ist hier ersetzt durch die Funktion Value(), es wird der Status des Geräts Schalter1 ausgelesen. Man hat nun die Möglichkeit, den Status eines beliebigen Geräts als Schaltbedingung einzubauen, es muss nicht dasselbe sein, das den notify auslöst. Also z.B.

```
define Schalter1Notify notify Schalter1 { if ( Value("Schalter1") eq "off" &&
Value("Schalter2") eq "off") {fhem("set wz_Media off")}} }
```

Hier wird wz_Media nur dann ausgeschaltet, wenn sowohl Schalter1 als auch Schalter2 den Zustand off haben.

Wie im Abschnitt [fhem-Konfiguration: Das besondere Gerät „Global“](#) gezeigt, verfügen viele Geräte nicht nur über den STATE im oberen Bereich (der bei den meisten Geräten on oder off ist, z.B. bei Thermometern aber die aktuellen Messwerte zeigt), sondern auch über Detailwerte, die sogenannten Readings im unteren Bildschirmbereich der Detailansicht.

The screenshot displays the fhem web interface for a device named 'ez_Aussensensor'. The left sidebar contains navigation links such as 'Floorplans', 'CUL_WS', 'Esszimmer', 'Korridor', 'Kueche', 'Media', 'Plots', 'Schlafzimmer', 'Unsorted', 'Wohnung', 'Wohnzimmer', 'Everything', 'Howto', 'Wiki', 'Details', 'Edit files', 'Select style', and 'Event monitor'. The main content area shows device details. At the top, there is a 'save' button. Below it, a table lists various attributes like CODE, CUL_MSGCNT, CUL_RAWMSG, CUL_RSSI, CUL_TIME, DEF, LASTIODev, MSGCNT, NAME, NR, STATE, TYPE, corr1, corr2, corr3, and corr4. The 'STATE' row is highlighted in blue and shows 'T: 10.4 H: 74.6'. Below this, the 'Readings' section is highlighted in red, showing a table with columns for the reading name, value, and timestamp. The 'humidity' and 'temperature' rows are circled in green. At the bottom, there is a section for 'attr' (attributes) with a dropdown menu set to 'ez_Aussensensor' and 'Audio', and a table listing 'fm_order', 'model', and 'room' with 'deleteattr' links. At the very bottom, there are links for 'Select icon' and 'Device specific help'.

Die Funktion **Value()** liest STATE des Geräts aus, oben blau markiert.

Um ein Reading (oben rot markiert) auszuwerten, wird die Funktion **ReadingsVal()** verwendet. Hier muss man zusätzlich zum Devicenamen auch den Namen des gewünschten Readings hinterlegen. Außerdem wird ein Default-Wert angegeben, der verwendet wird, falls das Auslesen des Readings fehlschlagen sollte. Die Syntax ist

```
ReadingsVal(<device>, <reading>, <default-Wert>)
```

Möchte man also einen Schaltvorgang z.B. von der gemessenen Temperatur des im screenshot dargestellten device abhängig machen, könnte das so aussehen:

```
define Heizungssteuerung at +*01:00:00 { if
(ReadingsVal("ez_Aussensensor","temperature",99) < 20) { fhem("set heizung on") } else
{ fhem("set heizung off") } }
```

Es wird also jede Stunde geprüft, ob die Außentemperatur unter 20 Grad Celsius liegt und abhängig davon die Heizung ein- oder ausgeschaltet. (o.k., man kann den Temperatursensor ja auch nach innen hängen ;-)

Wenn statt des Reading-Werts der Zeitstempel ausgelesen werden soll, verwenden Sie die Funktion `ReadingsTimestamp(<device>, <reading>, <default-Wert>)`

Spezielle Variablen und Operatoren

Wie bereits dargestellt gibt es in fhem ein paar spezielle Variablen (siehe auch [hier](#)):

fhem-Schreibweise	Erklärung
<code>\$EVENT</code>	Enthält den Wert des zuletzt eingetretenen Ereignisses, also z.B. <code>on</code> oder <code>off</code> oder <code>toggle</code> oder <code>20°C</code> . Die „alte“ Schreibweise dafür ist das Prozentzeichen, die neue und empfohlene Schreibweise ist <code>\$EVENT</code> .
<code>\$EVTPART0</code> , <code>\$EVTPART1</code> usw.	Wenn der vom Gerät übermittelte event aus mehreren Worten besteht (getrennt durch Leerzeichen), so enthält <code>\$EVTPARTx</code> die einzelnen "Worte". Bei einem event-Wert <code>"actuator: 86 %"</code> enthalten die Variablen also folgende Werte: <code>\$EVENT: actuator: 86 %</code> <code>\$EVTPART0: actuator:</code> <code>\$EVTPART1: 86</code> <code>\$EVTPART2: %</code>
<code>\$NAME</code>	Enthält den Namen des Geräts, von dem das Ereignis ausgelöst wurde. Die „alte“ Schreibweise dafür ist das <code>at</code> -Zeichen <code>@</code> , die neue und empfohlene Schreibweise ist <code>\$NAME</code> .
<code>\$we</code>	Wochenende – siehe nächste Seite

Man kann also schreiben:

```
define test notify *.* { if ($NAME eq "Taster" && $EVENT eq "on") {fhem("set lampe $EVENT")} }
```

Die Programmiersprache perl hat die Besonderheit, dass Variablen nicht mit einem Typ definiert werden. Deklariert man also z.B. `MeineVariable (my $MeineVariable;)`, so wird dabei nicht angegeben, ob sie vom Typ `Character`, `Integer`, `Date` o.ä. ist. Das vereinfacht die Deklaration und führt zu weniger Fehlermeldungen bei Wertzuweisungen im weiteren Programmverlauf. Allerdings muss bei späteren Auswertungen klargestellt werden, ob der Wert der Variable als `Character (String)` oder als `Zahl` betrachtet werden soll. Ein Beispiel:

```
my $var1 = '10'; my $var2 = '2';
# Ergibt true, da die Zahl 10 größer ist als die Zahl 2:
if ($var1 > $var2)...
# Ergibt false, da die Zeichenfolge 10 kleiner ist als die Zeichenfolge 2:
if ($var1 gt $var2)...
```

Die Unterscheidung wird also nicht zum Zeitpunkt der Variablendeklaration durchgeführt, sondern durch die Auswahl des Operators (hier `>` oder `gt`). Daher merke:

```
<, >, <=, >=, ==, != sind numerische Operatoren
lt, gt, le, ge, eq, ne sind die entsprechenden Stringvergleichsoperatoren
```

Auch sei darauf hingewiesen, dass der Operator `eq` auf identischen Inhalt prüft, zB

```
my $val="on"; if ($val eq "on") ... #ist true
```

Eine Prüfung auf ein Muster funktioniert NICHT:

```
my $val="fhem"; if ($val eq "*he*") ... #ist false
```

Hier kann der perl Mustervergleich `=~ m` verwendet werden:

```
my $val="fhem"; if ($val =~ m"he") ... #ist true
```

Dieser Mustervergleich (matching) ist äußerst mächtig und firmiert auch unter dem Namen "Regular Expression" oder kurz "regexp". Mehr Lesestoff findet sich z.B. [hier](#).

Hinterlegen eigener Programme – 99_myUtils.pm

Kleine eigene Automatisierungen können wie bereits beschrieben mittels notify als "Makro" hinterlegt werden. Werden diese eigenen Automatisierungen jedoch umfangreicher, lohnt sich das Anlegen einer separaten Programmdatei, in der diese eigenen Progrämmchen hinterlegt werden. Wie das geht und was dabei zu beachten ist, steht in [diesem Wiki-Artikel](#).

Auch werden bereits einige Helfer-Routinen bereits mit ausgeliefert. Sie sind gesammelt in 99_Utils.pm (**Achtung** – 99_Utils.pm wird ggf. per update überschrieben - ist nicht dasselbe wie 99_myUtils.pm – in 99_Utils.pm daher keine eigenen Routinen hinterlegen!). Eine Liste der in 99_Utils.pm enthaltenen Funktionen findet sich [hier](#).

Nur am Wochenende? \$we

Wenn Ihre Nachttischlampe als Wecker dienen und jeden Tag um 08:00 eingeschaltet werden soll, erreichen Sie das mit

```
define WeckenMitLicht at *08:00 set sz_Lampe on
```

Wenn Sie am Wochenende länger schlafen möchten und daher auf diesen Weckdienst verzichten wollen, können Sie das folgendermaßen erreichen:

```
define WeckenMitLicht at *08:00 { if (!$we) { fhem("set sz_Lampe on") } }
```

Die Variable `$we` wird von fhem automatisch gefüllt: Sie ist `true` an Samstagen und Sonntagen, und `false` an Wochentagen.

Mit einem `!` können Sie außerdem die Auswertung umkehren: `!$we` steht für „nicht am Wochenende“ und ist gleichbedeutend mit „nur an Wochentagen“.

Wenn Sie also am Wochenende zu einer späteren Zeit geweckt werden möchten:

```
define WeckenMitLichtWE at *09:30 { if ($we) { fhem("set sz_Lampe on") } }
```

In den Ferien? holiday, holiday2we

Das vorgenannte Beispiel macht Schaltvorgänge von Wochenenden abhängig. Möglicherweise möchten Sie ja an Ferientagen ebenso wie an Wochenenden erst später geweckt werden. Hierfür gibt es in fhem zum einen die Möglichkeit, Kalender zu definieren, zum anderen die Möglichkeit, die Ergebnisse des Kalenders ebenfalls für die Variable `$we` zu berücksichtigen.

Holiday

In einer separaten Datei werden alle Feiertage eines Jahres gelistet. Hier ein Beispiel des Datei-Inhalts von **Bayern.holiday** (mit einer kleinen Erweiterung am Ende) :

```
# Format für einzelne Tage: 1 MM-DD <Text>
1 01-01 Neujahr
1 01-06 Heilige Drei Koenige
1 05-01 Tag der Arbeit
1 08-15 Mariae Himmelfahrt
1 10-03 Tag der deutschen Einheit
1 11-01 Allerheiligen
1 12-25 1. Weihnachtstag
1 12-26 2. Weihnachtstag

# Osterbezogene Feiertage
# Format: 2 <relative Tage von Ostern> <Text>
2 -2 Karfreitag
2 1 Ostermontag
2 39 Christi Himmelfahrt
2 50 Pfingsten
2 60 Fronleichnam

5 -1 Wed 11 23 Buss und Betttag

# Urlaube
# Format: 4 MM-DD MM-DD <Text>
4 12-30 12-31 Urlaub1
```

Details zu den einzelnen Termintypen finden Sie in der commandref [hier](#).

Der Inhalt der .holiday-Datei kann im Webfrontend bearbeitet werden. Wählen Sie dazu den Menüpunkt ‚Edit Files‘ und klicken Sie auf den Namen der gewünschten Holiday-Datei.

Diese Datei wird mit

```
define <holidayname> holiday          also z.B.
define Bayern holiday
in fhem eingebunden.
```

Über die Kommandozeile können Sie dann prüfen, ob an einem bestimmten Datum ein Feiertag ist:

```
get <holidayname> mm-dd
```

```
get Bayern 12-26
```

Das Ergebnis ist „none“ oder der in der Liste hinterlegte Name des Feiertags.

holiday2we

Wenn eine Ihrer Feiertagsdateien zur Ermittlung von Schaltvorgängen einbezogen werden soll, wählen Sie diese zur Überleitung in die Variable \$we aus:

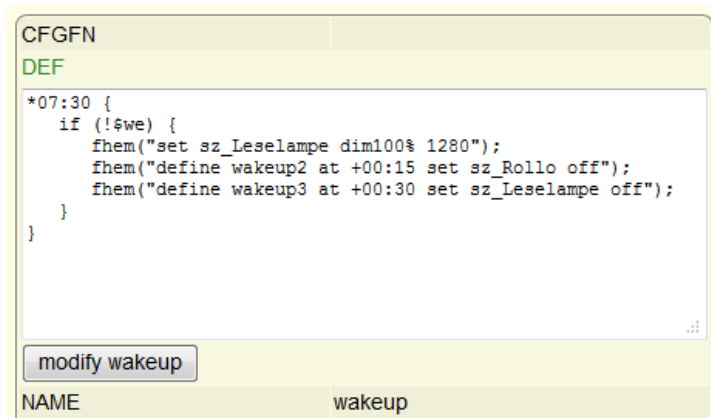
```
attr global holiday2we Bayern
```

Es wird nun täglich um Mitternacht geprüft, ob der anbrechende Tag gemäß der angegebenen holiday-Datei ein Feiertag ist und die Variable \$we ggf. auf true gesetzt.

Verschachteltes at

Ein häufiger Anwendungsfall ist das Schalten zu einer bestimmten Uhrzeit mit von der Startzeit abhängigen weiteren Schaltvorgängen. Beispielsweise soll Ihr morgendlicher Weckvorgang um 7:30 starten mit dem langsamen Hochdimmen der Nachtschlampe. Außerdem soll 15 Minuten später das Rollo hochgefahren und schließlich 30 Minuten nach wakeup-Zeit die Nachtschlampe wieder ausgeschaltet werden. Wichtig: Den Startzeitpunkt können Sie leicht mit `modify wakeup *08:00` ändern – die Angabe z.B. für das Rollo bezieht sich auf den Start dieser Prozedur, wird also durch das `modify` ebenfalls verschoben.

Als Code sieht das so aus:



Beachten Sie die Verwendung der * bei den Zeitangaben: Der wakeup-Zyklus soll täglich ausgeführt werden, also Angabe mit *. Nach diesem täglichen Start sollen aber die weiteren Steuerungen nur jeweils einmalig ausgeführt werden, daher ist hier bei den Zeitangaben **kein** * gesetzt.

Ein detailliert erklärtes Beispiel dazu finden Sie im [fhem-wiki](#).

Heizungssteuerung

Die Steuerung von Heizungen kann zunächst in drei Bereiche gegliedert werden:

1. Messen der Ist-Temperatur
2. Berechnen der erforderlichen Heizleistung, um von der gemessenen Ist-Temperatur zur gewünschten Zieltemperatur zu gelangen. Dabei soll die angeforderte Heizleistung umso größer sein, je größer der Unterschied von Ist- und Solltemperatur ist, damit die Solltemperatur möglichst schnell erreicht wird.
3. Regeln der Heizleistung

Messen der Ist-Temperatur

Die Messung erfolgt durch ein Thermometer.

Berechnen der erforderlichen Heizleistung

Die Berechnung wird von „PID-Reglern“ (Proportional-Integral-Differential) genannten Algorithmen übernommen. PID ist also eine Software, die aus den Eingabeparametern Ist- und Solltemperatur als Output die erforderliche Heizleistung berechnet. Diese wird in % angegeben, sie kann also z.B. 20% oder 90% der maximal verfügbaren Heizleistung betragen.

Thermostate

Alle zur Heizungssteuerung relevanten Hardwaresysteme bieten als separate Komponente ein „Thermostat“, das die Einstellung einer Solltemperatur ermöglicht, über ein eingebautes Thermometer verfügt und auch einen PID-Rechner enthält, der die erforderliche Heizleistung berechnet. Meist kann dabei die Solltemperatur direkt manuell eingestellt werden oder auch als Tages- und Wochenprogramm hinterlegt werden, so dass z.B. nachts die Solltemperatur niedriger eingestellt wird als tagsüber. In festen Zeitzyklen (z.B. alle fünf Minuten) wird dann die gemäß Solltemperatur benötigte Heizleistung berechnet.

Viele Thermostate bieten optional auch die Möglichkeit, einen Fensterkontakt anzukoppeln. Meldet der Fensterkontakt „Fenster offen“, wird auf eine einstellbare niedrigere Solltemperatur (z.B. 17°C) gewechselt und die Heizleistung entsprechend reduziert. Wird das Fenster geschlossen, kehrt der

Thermostat zur zuletzt eingestellten Solltemperatur zurück und passt darauf basierend die Heizleistung wieder an. So wird ein „für draußen heizen“ vermieden.

Regeln der Heizleistung

Das Regeln der Heizleistung erfolgt meist über sogenannte „Ventilstelltriebe“. Diese ersetzen die bisherigen (Thermostat-) Ventile an den Heizkörpern und sind per Funk regelbar. Der Austausch der Ventile ist dabei unproblematisch, da das eigentliche Ventil im Heizkörper verbleibt – es sind also keine Installateurs-Arbeiten erforderlich.

Auch gibt es Ventilstelltriebe, die den Thermostat selbst beinhalten. Der Unterschied zum (raum-)zentralen Thermostat ist lediglich der Standort der Messung der Ist-Temperatur.

Falls Sie selbst der Betreiber der Heizungsanlage sind (Etagenheizung oder eigene Heizanlage im Haus) besteht außerdem die Möglichkeit, die Heizungsanlage selbst zu steuern. Dazu bieten viele Heizanlagen die Möglichkeit, in die Brennersteuerung einzugreifen – insbesondere den Brenner „abzuwürgen“, auch wenn er laut heizsystem-eigener Steuerung weiter heizen würde,

Heizungssteuerung mit fhem

Wie beschrieben sind die verfügbaren Hardwaresysteme zur Heizungssteuerung in sich autarke Regelkreisläufe. Mit fhem kann man zur weiteren Steigerung der Energie-Effizienz und zur Erhöhung des Komforts in diese Regelungen eingreifen.

Häufigster Anwendungsfall ist dabei das Anpassen der Solltemperatur. Hierbei kann man das Wochenprogramm, das meist in den Thermostaten verfügbar ist, intelligent erweitern oder ersetzen.

So gibt es z.B. das spezielle fhem-Modul [Heating_Control](#) zum Ersetzen bzw. sinnvollen Erweitern der thermostat-internen Wochenprogramme.

Hier einige Anwendungsbeispiele:

- Setzen der Solltemperatur abhängig davon, ob wirklich jemand zuhause ist
- Setzen der Tag/Nachttemperatur nicht nach festen Uhrzeiten, sondern Absenkung wenn die Bewohner wirklich schlafen gehen und Rückkehr zur Wohlfühltemperatur 20 Minuten vor der in fhem eingestellten Weckzeit
- Zeitpunkt der Umschaltung von Nachtabsenkung auf Tagestemperatur von der Nacht-Tiefsttemperatur abhängig machen – je kälter die Tiefsttemperatur laut yahoo-weather, desto früher soll morgens die Heizung wieder anspringen
- Steuern der Solltemperatur abhängig von Einträgen in einem Google-Kalender
- Manuelles Steuern der Solltemperatur via VPN-Verbindung von unterwegs

Auch kann man über die reine Beeinflussung der Solltemperatur hinausgehen. So kann man z.B. den Heizungsbrenner abschalten, wenn alle Thermostate als erforderliche Heizleistung 0% melden.

Hinweis: Zur direkten Beeinflussung des Heizungsbrenners ist es in der Regel erforderlich, einen Funkschalter an die Heizungsanlage anzuschließen. Dies nicht von einem Elektriker oder Heizungsfachmann durchführen zu lassen, gefährdet nicht nur die Heizanlage, sondern auch Sie – es wäre „Sparen am falschen Ende“ !

In meinem eigenen Haushalt betreibe ich einen Funk-Raumthermostat in nur einem Raum, dessen Solltemperatur ich abhängig vom HomeStatus setze. Auf Ventilstelltriebe verzichte ich vollständig. Meldet der Thermostat eine erforderliche Heizleistung von <50%, wird der Heizbrenner meiner Etagenheizung durch fhem abgeschaltet. Meldet der Thermostat eine erforderliche Heizleistung von

>50%, wird der Brenner freigegeben und die Heizungsanlage arbeitet auf die dort fest eingestellte Solltemperatur (=Wohlfühltemperatur 21°C) hin.

Da fhem zwei PID-Module bietet ([PID](#) und die erweiterte Version [PID20](#)), ist es außerdem denkbar, reguläre Funkthermometer zu verwenden, die erforderliche Heizleistung durch den PID-Regler in fhem zu berechnen und gekoppelte Funk-Ventilstelltriebe direkt zu steuern und/oder den Heizbrenner direkt zu steuern. Dabei können die Komponenten durchaus aus unterschiedlichen Hardwaresystemen stammen, also z.B. Temperaturmessung mit 1-wire und Heizkörperreglung mit FHT, HomeMatic oder MAX.

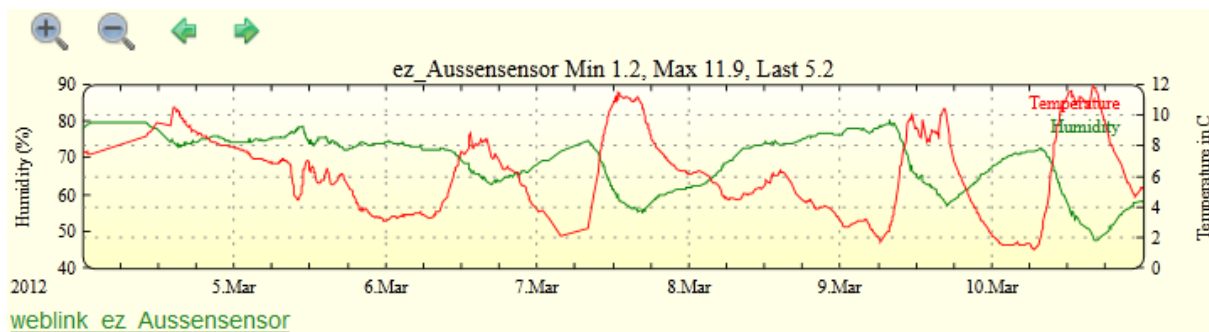
Für welchen Weg Sie sich entscheiden, hängt von den Gegebenheiten ihres Heims und Ihrer Heizungsanlage ab.

Die an fhem anbindbaren Hardwaresysteme sind:

Hardwaresystem	Raumthermostat	Ventilstelltrieb	Thermostat + Stelltrieb in einem
FHT	FHT80b	FHT8v	-/-
HomeMatic	HM-CC-TC (alt)	HM-CC-VD (alt)	HM-CC-RT-DN (neu)
MAX!	-/-	-/-	Ja
1-wire und andere	Nur Thermometer	-/-	-/-

Temperaturverläufe als Diagramm – Logs & Plots

Die autocreate-Funktion erzeugt beim Erstellen bestimmter Messgeräte wie z.B. FHT-80b (Heizungssteuerung) oder S300TH (Thermostat und Hygrometer) automatisch auch ein Verlaufsdigramm. Hier als Beispiel die Wochendarstellung für ein S300TH, das als Außenthermometer dient. Die Temperatur wird mit einer roten, die Luftfeuchtigkeit mit einer grünen Linie dargestellt:



Mit den Schaltflächen am oberen Rand kann die Zoomstufe (Tag, Woche, Monat) verändert werden, auch kann man mit den grünen Pfeiltasten durch die Historie ‚wandern‘.

Die dargestellten Werte werden aus der zugeordneten Logdatei gelesen. In der commandref finden Sie die relevanten Infos unter [FileLog](#) und [weblink fileplot](#) .

Details dazu folgen in der erweiterten Version dieses Dokuments.

Jemand zuhause? Anwesenheitserkennung und HomeStatus

Ein zentraler Parameter der Haussteuerung ist unterschiedliches Verhalten abhängig davon, ob derzeit jemand zuhause ist oder nicht (Solltemperatur der Heizung, Scharfschalten der Alarmanlage, Musik an/aus etc.) bzw. beim Übergang von einem Zustand zum Anderen.

Dazu besteht die Möglichkeit, einen „Status“ des Hauses zu definieren (ich nenne ihn hier einmal „HomeStatus“) der die Werte *Zuhause* oder *Nicht zuhause* annehmen kann oder – als differenziertere Variante – *Zuhause*, *Schlafen*, *Kurz weg* und *Lange weg*.

Beim Wechsel in einen neuen Status können dabei unterschiedliche Aktionen ausgelöst werden:

Wechsel zu Status	Beleuchtung	Heizung	Musik
Zuhause	Korridorbeleuchtung an für 5 Minuten	Solltemperatur 21°C	Starten
Schlafen	Alle Lichter aus außer im Schlafzimmer	Solltemperatur 18°C	Stoppen
Kurz weg	Alle aus	Solltemperatur 19°C	Stoppen
Lange weg	Alle aus	Solltemperatur 17°C	Server herunterfahren

Auch kann der HomeStatus zur Steuerung weiterer Geräte verwendet werden. So soll z.B. morgens zur Weckzeit das Wakeuplight nur dann aktiviert werden, wenn auch jemand zuhause ist (HomeStatus *Zuhause* oder *Schlafen*) oder nachts durch den Bewegungsmelder nicht das grelle Korridorlicht, sondern nur eine schwache Beleuchtung eingeschaltet werden (HomeStatus *Schlafen*), während die Alarmanlage nur dann aktiv ist, wenn niemand zuhause ist (HomeStatus *Kurz weg* oder *Lange weg*).

In fhem sieht diese Tabelle z.B. so aus:

Status	desired-temp	ez_FHT	ez_LichtMaster	ko_LichtKorridor	sz_LichtMaster	sz_Main	whg_Audio	whg_WLAN	wz_LichtMaster	wz_MacMini
away	19.0	[off]	[off]	[off]	[off]	[off]	[off]	[alarm]	[off]	[off]
home	21.0	[off]	[on]	[off]	[off]	[on]	[on]	[music]	[off]	[on]
sleep	19.0	[off]	[off]	[off]	[off]	[off]	[off]	[alarm]	[off]	[off]
travel	17.0	[off]	[off]	[off]	[off]	[off]	[off]	[alarm]	[off]	[off]

(In der ersten Zeile sieht man den aktuelle Schaltzustand der Geräte, im unteren Block den Zielzustand je Status, hier home, sleep, away, travel)

Ein Beispiel mit dem zugehörigen code findet sich im [fhem-wiki](#).

Der Komfort kann weiter erhöht werden, indem das Setzen des Status automatisch erfolgt. Zur Anwesenheitserkennung gibt es unterschiedliche Mechanismen:

- Prüfung, ob bestimmte Geräte/Handys im WLAN eingeloggt sind (fhem-Modul [PRESENCE](#)) – problematisch ist hierbei, dass häufig ein zeitlicher Verzug zwischen realer Ankunft/Abreise und dem Melden der An/Abmeldung des Geräts auftritt. Auch melden sich viele mobile Geräte bei Nicht-Nutzung aus dem WLAN ab (Stromsparmmodus).

- Verwendung von Bewegungsmeldern – funktioniert gut für den Übergang von „niemand da“ -> „bewohnt“, jedoch ist schwierig herauszubekommen, ob alle Bewohner das Haus verlassen haben oder nur gebannt vor dem Fernseher sitzen.
- Verwendung von Mikrotastern, um zu erkennen, ob die Haus-/Wohnungstür verschlossen ist – nur nutzbar falls nachts nicht abgeschlossen wird, erkennt auch nicht, ob niemand zuhause ist oder alle schlafen.
- Erkennung über Bluetooth – dies verbraucht bei mobilen Geräten jedoch noch zu viel Strom, um dauerhaft genutzt werden zu können.
- Erkennung des Nutzer-Aufenthaltsorts, z.B. über Geofancy (zu dem auch ein gleichnamiges fhem-Modul existiert) – auch hierzu erforderliche GPS-Empfänger verbrauchen in mobilen Geräten zu viel Strom, um dauerhaft eingeschaltet bleiben zu können.
- Manuelles Setzen des HomeStatus über einen Wandschalter neben der Eingangstür – funktioniert zuverlässig, sofern nicht vergessen wird, den Schalter beim Verlassen oder Betreten der Wohnung zu betätigen :)

Der „Königsweg“ für dieses Thema ist also leider noch nicht gefunden. Welcher Weg für Ihre Umgebung am besten passt, müssen Sie selbst entscheiden.

In meinem eigenen Haushalt habe ich mich für das manuelle Setzen des HomeStatus durch einen Schalter neben der Eingangstür und eine Fernbedienung neben dem Bett entschieden.

Anwesenheitssimulation

Um während des eigenen Urlaubs potentiellen Einbrechern vorzugaukeln, dass jemand zuhause sei, lassen sich Geräte in definierbaren Zeiträumen zufällig ein- und ausschalten. Beispiele und die Einrichtung sind beschrieben im Modul [RandomTimer](#).

Schaltungen abhängig von Telefonanrufen – Callmonitor

Wenn Sie eine FritzBox in Ihrem Heimnetz betreiben, können Sie eingehende Anrufe erkennen und daraufhin z.B. die Lautstärke Ihrer Audiogeräte reduzieren sowie nach dem Beenden des Gesprächs die Lautstärke wieder auf den vorherigen Stand zurücksetzen. Auch ist es möglich, eine bestimmte Rufnummer zur Heizungssteuerung zu reservieren und z.B. die Solltemperatur der Heizung zu erhöhen, wenn auf dieser Rufnummer ein Anruf von definierten Herkunfts-Telefonnummern erfolgt.

Dabei muss fhem nicht auf der FritzBox laufen, auch das Koppeln einer entfernten FritzBox von Ihrem fhem-Server aus ist möglich. Details siehe [FB Callmonitor](#).

Multimedia-Geräte

Eine zunehmende Zahl von Multimedia-Geräten kann über LAN gesteuert werden. Ein einfaches Anwendungsbeispiel ist das Regeln der Lautstärke bei eingehendem Anruf. Die unten angegebenen Geräte können aus fhem heraus gesteuert werden. Die Anzahl verfügbarer Module wächst ständig, hier eine kleine Auswahl (Stand 01/2014):



Gerätetyp	Hersteller / Gerät	fhem-Modul
Audio	Logitech Squeezebox	SB_PLAYER
	MPD	MPD
	Mplayer (zur mp3- oder Audiostream-Wiedergabe)	StreamRadio
	Sonos	SONOS, SONOSPLAYER
	Yamaha Blu-ray Player	YAMAHA_BD
TV / Video	Sony TV	Viera
	Samsung TV	STV
	LG TV	LGTV
	Philips TV	PhilipsTV
	AppleTV	iTunes
AVR (Audio/Video-Receiver)	Yamaha	YAMAHA_AVR
	Onkyo	ONKYO_AVR
	Denon	DENON_AVR
Satellitenreceiver / Set-top-boxen	Dreambox, VUplus	ENIGMA2
Mediacenter	XBMC	XBMC
	MediaPortal	MediaPortal
	iTunes	iTunes
Radiowecker	Geräte mit Firmware von www.listenlive.nl	Listenlive

Details zu den genannten fhem-Modulen finden sich in der [commandref](#).

Außerdem gibt es das Modul [remotecontrol](#), das eine grafische Fernbedienung (wie oben links) anzeigt, die an jedes der o.g. Module angepasst und gekoppelt werden kann. So können Sie Ihr Tablet als Fernbedienung für Ihre an fhem gekoppelten Geräte verwenden.

Infrarot-Fernbedienung

Einige Geräte sind nur über Infrarot ansprechbar. Zur Einbindung in fhem gibt es zwei FS20 Infrarot-Geräte: FS20-iru (IR-Empfänger) und FS20-IRF (IR-Sender) sowie einen [iTach](#).

Prüfung und Aufwecken von Servern im Heimnetzwerk: WOL

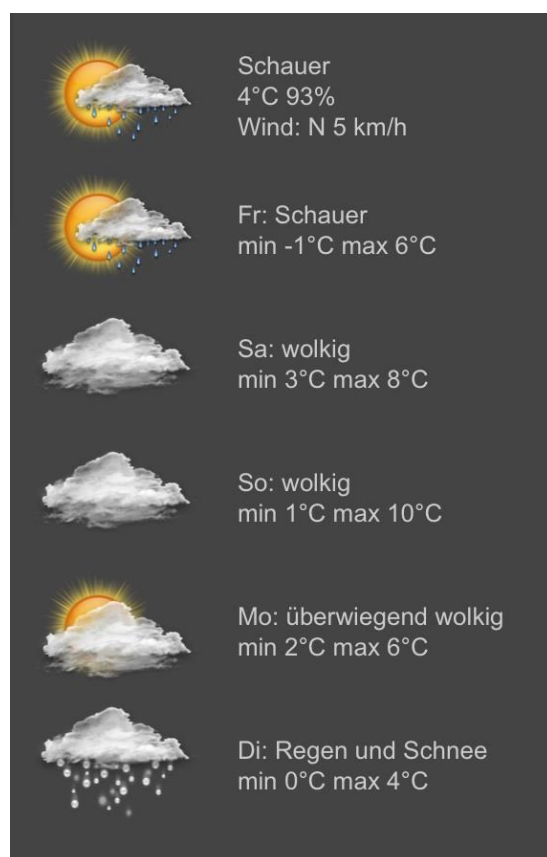
Zur Statusprüfung und dem Senden von WOL-packages dient das Modul [WOL](#).

Webservices, http-Verbindungen

Wetterbericht einbinden

Hierfür gibt es die fhem-Module [weather](#) basierend auf yahoo-Weather, sowie das Modul [GDS](#) zur Einbindung von Daten des deutschen Wetterdienstes. Beide aktualisieren die Informationen in einstellbaren Intervallen (z.B. alle 15 Minuten) und stellen die gelesenen Daten in fhem als Readings zur Verfügung. In das Webfrontend können so Wetterberichtsdaten eingeblendet werden. Die Vorhersagedaten können beispielsweise zur vorausschauenden Heizungssteuerung verwendet werden, um z.B. morgens die Heizung früher anspringen zu lassen, wenn es nachts besonders kalt war, oder um die Markise einzufahren und Rollläden zu schließen wenn starke Winde oder Hagel vorhergesagt sind.

condition	Schauer
current_date_time	17 Jan 2014 3:20 pm CET
day_of_week	Fr
fc1_code	11
fc1_condition	Schauer
fc1_day_of_week	Fr
fc1_high_c	6
fc1_icon	chance_of_rain
fc1_low_c	-1
fc2_code	26
fc2_condition	wolkig
fc2_day_of_week	Sa
fc2_high_c	8
fc2_icon	cloudy
fc2_low_c	3
fc3_code	26
fc3_condition	wolkig
fc3_day_of_week	So
fc3_high_c	10
fc3_icon	cloudy
fc3_low_c	1
fc4_code	11
fc4_condition	Schauer
fc4_day_of_week	Mo
fc4_high_c	6
fc4_icon	chance_of_rain
fc4_low_c	2
fc5_code	5
fc5_condition	Regen und Schnee
fc5_day_of_week	Di
fc5_high_c	4
fc5_icon	rainsnow
fc5_low_c	0
humidity	93
icon	chance_of_rain
pressure	982
pressure_trend	2
pressure_trend_sym	-
pressure_trend_txt	fallend
state	T: 4 H: 93 W: 3
temp_c	4
temp_f	39
temperature	4
visibility	8
wind	3
wind_chill	4
wind_condition	Wind: S 3 km/h
wind_direction	180
wind_speed	3



Zur Einbindung in das fhem-Webfrontend oder einen floorplan stehen besonders schöne Icons zur Verfügung.

Das Weather-Modul stellt viele Readings zur Verfügung, die für weitere Steuerungen verwendet werden können.

Google-Kalender, Mails und Messaging

Es können Einträge aus einem Google-Kalender verwendet werden, um Schaltvorgänge in fhem auszulösen. Details und umfangreiche Beispiele stehen [hier](#) im commandref-Eintrag sowie [hier](#) im Wiki.

Auch eingehende Mails können als Schalt-Trigger in fhem eingebunden werden. Das zugehörige fhem-Modul heißt [Mailcheck](#).

Ebenso können Mails mit Statusinformationen versendet werden, z.B. beim Auftreten eines Alarms oder bei Eintreffen eines Anrufs. Das zu verwendende Modul ist abhängig vom Betriebssystem des Rechners, auf dem fhem betrieben wird. Ein generisches Modul ist [MSGMail](#), speziell für die FritzBox gibt es die Routine `FB_mail($$$)`, siehe auch [hier](#). Weitere Infos finden sich im [fhem-Wiki](#) und im [fhem-Forum](#).

Außerdem ist das Versenden von Push-Nachrichten an Smartphones möglich, siehe [Pushover](#).

Kommunikation über HTTP

Immer mehr Geräte bieten eigene Web-Seiten, über die der Status ausgelesen werden kann bzw. eine Steuerung der Geräte möglich ist.

Um den Status eines Geräts auszulesen, muss häufig eine entsprechende URL aufgerufen werden, die dann eine html-Seite anzeigt, in der die gewünschten Statusinformationen enthalten sind.

Verwenden Sie das Modul HTTPMOD, das Sie durch Angabe der URL konfigurieren und dabei auch angeben, welche Information der zurückgegebenen Webseite in welches Reading extrahiert werden soll. Änderungen der Readingswerte lösen dabei fhem-Ereignisse aus, die dann mittels notify zur Steuerung beliebiger Aktionen verwendet werden können.

Um auf Basis beliebiger Ereignisse ein Gerät über dessen Webseite (also Angabe der URL erweitert um die entsprechenden CGI-Parameter) zu steuern, verwenden Sie die mit fhem ausgelieferte Funktion `GetFileFromURL()` (oder eine andere Funktion aus dem Funktionspool `HttpUtils.pm`) oder wiederum das o.g. Modul HTTPMOD.

Ist das parsing des Moduls HTTPMOD nicht hinreichend, schreiben Sie eine eigene Routine in `myUtils`, in der Sie mittels `GetFileFromURL()` die Geräte-Webpage „abholen“ und anschließend mittels `regexprs` die gewünschten Werte extrahieren.

fhem neu starten – shutdown restart

Um –vor allem nach `update`- das Programm neu zu starten, führen Sie den Befehl

```
shutdown restart aus.
```

Möchten Sie lediglich die Konfigurationsdatei `fhem.cfg` erneut einlesen, ohne fhem neu zu starten, nutzen Sie den Befehl `rereadcfg`.

fhem Programmaktualisierungen – update

Damit Sie Ihre fhem-Installation immer auf dem neuesten Stand halten, sichern Sie Ihren Stand regelmäßig in Backups und führen anschließend den Befehl `updatecheck` aus, so können Sie sich die neuesten bugfixes und Weiterentwicklungen anzeigen lassen. Werden diese Weiterentwicklungen von Ihnen benötigt, führen Sie nach einer Datensicherung den Befehl `update` aus zum herunterladen der neuesten Programmversionen auf Ihr Gerät.

Infoquellen

Mit den bisher dargestellten Möglichkeiten sollten Sie alle Grundlagen haben, um Ihre eigene Anwendung in fhem abzubilden.

Weitere hilfreiche Informationsquellen sind

- Die fhem-Doku unter <http://www.fhem.de>
- Die Befehlsreferenz unter <http://fhem.de/commandref.html>
- Das fhem-wiki unter <http://www.fhemwiki.de/wiki/FHEM>
- Das sehr aktive fhem-Forum unter <http://forum.fhem.de/>

Im folgenden Abschnitt erhalten Sie eine Kurzübersicht über weitere Möglichkeiten, die Sie mit fhem haben.

Im „Sammelsurium“ finden Sie kleine Hinweisschnipsel, z.B. zum Flashen Ihres CUL.

Bei der weiteren Arbeit mit fhem wünsche ich viel Spaß und Erfolg!

München, Februar 2014

Uli Maaß

Kurz betrachtet: Weitere Geräte und Funktionen

Die folgenden Kurzbeschreibungen sollen einen Ausblick auf Inhalte einer erweiterten Version von „Heimautomatisierung mit fhem“ bieten. Für ein Selbststudium sind z.T. bereits Links auf commandref und Wiki angegeben.

Erweitert: Logs & Plots

Erstellen Sie eigene Logdateien, die Daten unterschiedlicher Geräte zusammenfassen und damit die Ausgabe kombinierter Diagramme ermöglichen. Dieses Themengebiet ist sehr flexibel – und daher leider auch etwas kompliziert. In der commandref finden Sie die relevanten Infos unter [FileLog](#) und [weblink fileplot](#) sowie eine detaillierte Beschreibung unter [Creating plots](#) .

Dummy

In fhem können dummy-devices angelegt werden, die wie eine Variable genutzt werden können. So wird z.B. zur Abbildung des HomeStatus ein dummy benutzt: Der Wert des HomeStatus soll gespeichert werden, eine Funkkommunikation wie z.B. bei einem FS20-Gerät gibt es aber nicht. Diesem dummy können dann Attribute zugeordnet werden, die z.B. nur das Setzen bestimmter Werte zulassen, siehe Nur bestimmte Befehle für ein Gerät - webCmd.

Infos in der commandref finden Sie [hier](#).

Watchdog

Mit dieser Funktion können Vorfälle überprüft und ggf. Aktionen eingeleitet werden, wenn z.B. ein Fenster länger als einen bestimmten Zeitraum offen steht oder sich ein Gerät seit über einen definierten Zeitraum nicht meldet. Watchdog ist also immer dann relevant, wenn nach Eintreten eines bestimmten Ereignisses nach einer bestimmten Zeitspanne geprüft werden soll, ob dasselbe oder ein anderes event (nicht) erneut eingetreten ist. Hier finden Sie die [commandref-Infos](#)

Grenzwertgesteuertes Schalten – threshold und dewpoint

Als Auslöser zum Schalten können auch besondere Berechnungen vorgenommen oder Grenzwerte vorgegeben werden. [Dewpoint](#) berechnet den Taupunkt und ermöglicht damit z.B. das zielgerichtete Schalten zur Entfeuchtung von Badezimmern oder Kellern.

Das Modul [THRESHOLD](#) gestattet die Definition von Schwellwerten, bei deren Über- oder Unterschreiten eine Aktion ausgelöst werden kann.

shell-commands

Aus fhem heraus können nicht nur Perl-Programme, sondern auch Betriebssystem-Programme gestartet werden. Siehe hierzu den Aufruf über system() oder `backticks` - mehr Details stehen in der commandref [hier](#).

Zusammenfassungen – readingsGroup

Mit dem Modul readingsGroup lässt sich ein Pseudo-Device anlegen, das dazu dient, readings unterschiedlicher Geräte in einer gemeinsamen liste anzuzeigen. Diese Liste kann dann in einem beliebigen Raum oder in einem floorplan angezeigt werden.

Beispiele sind die Anzeige des readings battery von allen devices in Ihrer Konfiguration, oder das Anzeigen aller Temperature-Messwerte. Es kann auch eine beliebige Liste unterschiedlicher readings von unterschiedlichen Geräten zusammengestellt werden. Details stehen in der [Modulbeschreibung](#), viele Beispiele finden Sie im entsprechenden [Wiki-Artikel](#).

Schalten an bestimmten Wochentagen

Das Schalten von Geräten nach einem Zeitplan, der nicht jeden Tag gleich ist, lässt sich mit dem Modul [WeekdayTimer](#) leicht einrichten.

Kleine Programmierbeispiele

Wakeup-Light

Ihre Nachtschlampe soll als Wakeuplight funktionieren, also morgens durch langsames ‚hochdimmen‘ den Sonnenaufgang simulieren, um das Aufwachen erträglicher zu machen.

Details in einem [Wiki-Artikel](#).

Untoggle

Beim Verwenden von Schaltern als Taster (4-Kanal-Modus) soll das gesendete ‚toggle‘ in ‚on‘, bzw. ‚off‘ umgesetzt werden, um die Darstellung im Webfrontend zu verschönern.

Die Lösung steht in einem [Wiki-Artikel](#).

Sunset und Sunrise

Steuerzeitpunkte können vom Sonnenauf- und -untergang abhängig gemacht werden. Infos stehen in der [commandref](#). Auch eine dämmerungsabhängige Steuerung ist möglich auf Basis von [Twilight](#).

Tag und Nacht - isday

Wenn Sie einen Bewegungsmelder haben, der aber für ein Nachtlicht nur nachts auslösen soll, verwenden Sie die Funktion `isday()`:

```
define Nachtlicht notify Bewegungsmelder {if (!isday() ) {fhem("set lampe on")}} }
```

Da `isday()` von `SUNRISE_EL` gesetzt wird, muss dieses Modul aktiv sein.

Erstellen eigener fhem-Programme

Wenn Ihre eigenen Routinen an `notify` oder `at` zu lang und unübersichtlich werden, lohnt sich das Auslagern in eigene Programmdateien. Erste Schritte dazu finden Sie [hier](#).

Beachten Sie die Übernahme von Code-Änderungen mit `reload`.

Beachten Sie die fhem Perl-Specials aus der [commandref](#) (siehe [hier](#)).

Entwicklung eigener fhem-Module

Der fhem-core ist bewusst flexibel gestaltet, um Entwicklern das Anbinden eigener Module zu erleichtern. Einen Einstieg zur Entwicklung eigener Module mit einer Erklärung der Interface-Datenstrukturen findet sich im wiki [hier](#).

Neue Module können auch in das [fhem-repository in sourceforge](#) aufgenommen werden. Dabei gibt es zwei Varianten:

- Einchecken in den Zweig „contrib“. Hier eingeecheckte Module werden nicht in die fhem-Distribution aufgenommen, werden nicht über „update“ verteilt und erscheinen auch nicht in der [commandref](#). Möchte ein User dieses Modul verwenden, muss es manuell aus dem SVN heruntergeladen werden.
- Einchecken in den Zweig fhem/FHEM. Module, die hier eingeecheckt werden, werden automatisch zum Bestandteil der nächsten fhem-Version. Folgende Kriterien sind zu erfüllen:
 - Das Modul muss eine Dokumentation enthalten, die dann automatisch in die fhem [commandref](#) eingebunden wird. Als Beispiel kann ein beliebiges anderes fhem-Modul dienen.

- Der Modulator muss bereit sein, das Modul über das Forum mindestens drei Monate lang zu betreuen, also auf user-Fragen zu reagieren und ggf. bugfixes zu implementieren.

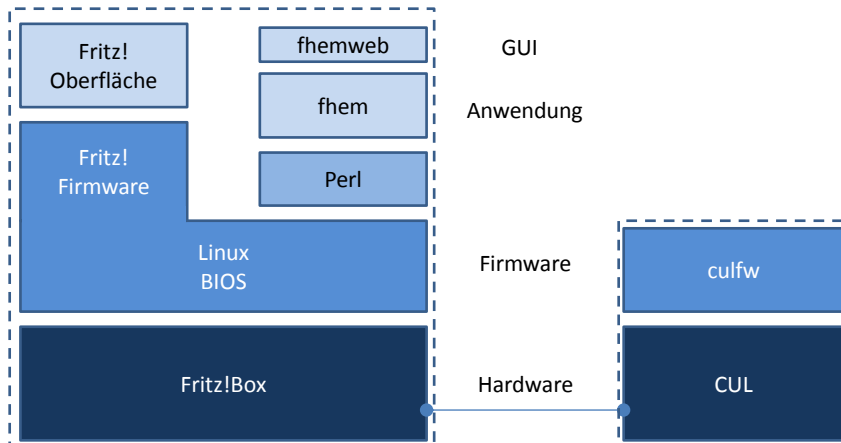
Für beide Wege muss sich der Modulator einen logon zu Sourceforge besorgen und dann eine Mail an Rudolf König senden, damit der user Schreibrechte für den fhem-Branch erhält.

Sammelsurium

FritzBox specials

Schichtenmodell & Zugänge

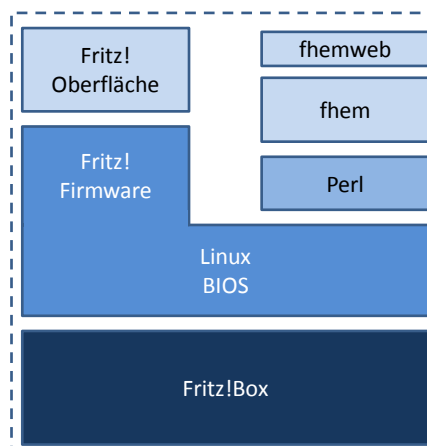
Schichtenmodell FB7390



Zugänge

<http://fritz.box:80>
<https://<WAN-IP>:443>

telnet://fritz.box:23



<http://fritz.box:8083/fhem>
<http://fritz.box:8084/fhem>

telnet://fritz.box:7072

FB7390: fhem-Installation

Es gibt es zwei Varianten. Beide führen zu einem lauffähigen fhem, das an der Oberfläche identisch aussieht – auf der Linux-Ebene laufen diese Systeme aber unterschiedlich:

1. Fritz!Labor-Image von AVM, das fhem auf Ihre FritzBox installiert. Diese Distribution läuft in einer sogenannten "chroot"-Umgebung. Diese ist besser gegen Zugriff von außen geschützt, erlaubt jedoch keinen Zugriff auf die FB-eigenen Funktionen (z.B. WLAN von fhem aus ein- und ausschalten, Mailversand, Zugriff auf Telefonie-Befehle wie den callmonitor usw.)
2. fhem-Distribution von fhem.de. Läuft eben NICHT in einer chroot-Umgebung, hat kürzere Pfade (praktisch!) und erlaubt den Zugriff auf die FB-eigenen Funktionen. Den download gibt's [hier](#)

Egal ob Sie das Labor-Image oder das von fhem.de verwenden – es kann wie ein Firmwareupdate über die FritzBox-Oberfläche installiert werden. Damit ist die Installation von fhem abgeschlossen. Sie können nun den CUL in die FritzBox stecken – der CUL blinkt, im FritzBox-Systemprotokoll steht: Unsupported USB-Device – klingt nicht gut, ist aber ok. Im fhem-Startbildschirm erscheint ‚CUL initialized‘ – das entspricht ‚running‘.

fhem kann nun aufgerufen werden mit <http://fritz.box:8083/fhem>

Bei Problemen:

- in der FB unter Heimnetz\USB-Geräte den USB-Fernanschluss aktivieren (dort Häkchen bei ‚USB-Geräte‘ setzen), dann wieder deaktivieren -> bei manchen hat's geholfen und es erschien wie von Zauberhand unter /dev auch ttyACM0, was dann auch von fhem erkannt wird
- Oder FritzBox neu starten
- Oder CUL erneut flashen

Siehe auch den Post im [Wiki](#).

FB7390: Neustart von fhem (Restart)

Manuelles Stoppen des fhem-Servers mit `shutdown`; manuelles Starten mit:

```
cd /var/InternerSpeicher/fhem
sh startfhem
```

FB7390: Ein/Ausschalten des WLAN aus fhem

```
define n_wlan notify FS20button {\
  my $wstate = ("%" eq "on" ? "start" : "stop");;\
  system("/etc/init.d/rc.net wlan$wstate &");;\
}
```

Quelle: [Forums-Beitrag](#)

Zugang über Terminalprogramm, Telnet

Telnet-Client herunterladen und installieren, zB PUTTY for windows (install als Admin starten!)

PuTTY verwenden:

Hinweis: Für eine FritzBox kann am DECT-Telefon der Telnet-Zugang der FritzBox ein- oder ausgeschaltet werden:

```
#96*7* Telnet an
#96*8* Telnet aus
```

Siehe auch Schichtenmodell – erreichen von:

- fhem: <ip> port 7072, Protocol RAW (nicht telnet), flag "Terminal:Implicit CR in every LF"
Sobald das Terminal-Fenster erscheint, einmal <enter> tippen, dann den Befehl `inform timer` mit <enter> eingeben. Ab sofort werden alle Funktelegramme angezeigt. Dieses Fenster kann

man im Hintergrund länger geöffnet lassen, um den Funkverkehr über einen längeren Zeitraum zu beobachten. Alle Funktelegramme werden auch im Log mitprotokolliert.

- Linux: <ip> port 21, Protocol telnet
-

fhem-Routinen speziell für die FritzBox

Eine Sammlung von Hilfsroutinen speziell für die FritzBox findet sich in FritzBoxUtils.pm im Ordner fhem/FHEM. Wollen Sie diese Routinen verwenden, müssen Sie das Laden dieser Datei auslösen durch einen Eintrag in Ihrer 99_myUtils.pm. Tragen Sie dort zu Anfang ein:

```
use FritzBoxUtils;
```

Einige der in dieser Programmdatei enthaltenen FritzBox-Hilfsprogramme sind im [fhemwiki](#) beschrieben und erklärt.

Flashen der CUL-Firmware an einem PC

Das Flashen der CUL-Firmware wird seit fhem 5.3 automatisch durchgeführt. Sollte dies einmal nicht funktionieren, nutzen Sie diese Schritte:

1. [FLIP runterladen](#) (Programm zum flashen des ROMs auf dem CUL)
2. CUL in den Rechner stecken (beim Einstecken Mikroschalter gedrückt halten!), dann zum Installieren des USB-Treibers: Im Windows Gerätemanager den device ATMEL xxxx anklicken, Eigenschaften, Gerätetreiber aktualisieren, von Festplatte, Verzeichnis c:\Programme\FliP\<flipversion>\USB auswählen und Treiber installieren lassen.

<http://code.google.com/p/micropendous/wiki/ProgramAndTestWindows>

3. CUL-Firmware (Hex) für fhem [runterladen](#) (Version passend zur CUL-Hardware-Version auswählen, bei mir CUL-Hardware-Version V3.2, also hex V3)
Siehe auch <http://culfw.de/culfw.html>
 4. Flip starten, unter ‚select device‘ exakt den Chip auswählen, der als Name des CUL im Geräte-Manager angezeigt wird (je nach Modell, zB ATU822U4), danach in der Flip Symbolleiste auf den 2. Button von links klicken (auch wenn der grau sein sollte – funktioniert trotzdem) und ‚USB‘ auswählen. Sollte hier eine Fehlermeldung auftauchen, hat Schritt 2 nicht geklappt oder es wurde der falsche Chip ausgewählt.
Dann mittels Menü Datei das bei Schritt 3 geladene hex-file nach Flip einlesen und mit button unten links auf den CUL flashen
 5. Wenn der CUL nach Abziehen und erneut in den Rechner stecken langsam grün blinkt, hat’s wohl geklappt.
-

Fehlermeldungen

LOVF

If too many messages for FHT devices are queued in CUL, the fht buffer subsystem of CUL overflows. You get EOB (end of buffer) messages and likely **LOVF** (limit overflow) messages, too. define `attr CUL fhtsoftbuffer 1` to activate a quasi unlimited software buffer in fhem itself to avoid this behavior. Also check the [Wiki](#).

Toggeln

Um bei 4-Kanal mittels Taster ‚togglen‘ zu können, aber dennoch im Webfrontend ein Glühbirnensymbol statt dem Wort ‚toggle‘ zu erhalten, diese [Anleitung](#) befolgen.

Basics: FHZ1x000pc versus CUL+CUNO Systemvergleich

Befehle an fhem

Um von einem anderen Rechner im Heimnetzwerk Befehle an fhem zu senden, unter Linux z.B.:

```
/volume1/@appstore/fhem/fhem/fhem.pl 7072 "set LillifeeDI dim100%"
```

Quelle: [Forums-Beitrag](#)

Anleitung zum Einrichten von Funktionsgruppen, Local Master, Global Master

Eine gute Anleitung zum Einrichten von Local Master, Global Master und Anlernen der Aktoren gibt's [hier](#).

Anhang: Ein Einblick in die Konfigurationsdatei fhem.cfg

Hier eine kurze Erläuterung ‚am lebenden Objekt‘:

<pre>attr global autoload_undefined_devices 1 attr global holiday2we Bayern attr global logfile /var/InternerSpeicher/fhem/log/fhem-%Y-%m.log attr global modpath /var/InternerSpeicher/fhem attr global port 7072 global attr global statefile /var/InternerSpeicher/fhem/log/fhem.save attr global verbose 3 define CUL CUL /dev/ttyACM0@38400 2332 define WEB FHEMWEB 8083 global define WEBS FHEMWEB 8084 global attr WEBS smallscreen 1 define WEBP FHEMWEB 8085 global attr WEBP touchpad 1 define autcreate autcreate attr autcreate autosave 1 attr autcreate filelog /var/InternerSpeicher/fhem/log/%NAME-%Y.log attr autcreate weblink 1 attr autcreate weblink_room Plots define Bayern holiday define HomeStatus dummy attr HomeStatus room Wohnung define ez_Audio FS20 2332 51 attr ez_Audio model fs20st attr ez_Audio room Esszimmer define FileLog_ez_Audio FileLog /var/InternerSpeicher/fhem/log/ez_Audio-%Y.log ez_Audio attr FileLog_ez_Audio logtype text define ez_LichtRegal FS20 6969 00 attr ez_LichtRegal model fs20st attr ez_LichtRegal room Esszimmer define FileLog_ez_LichtRegal FileLog /var/InternerSpeicher/fhem/log/ez_LichtRegal-%Y.log ez_LichtRegal attr FileLog_ez_LichtRegal logtype text</pre>	Feiertage Bayern Pfad und Name der Logdatei Pfad zum FHEM-Verzeichnis fhem soll über port 7072 per telnet zugänglich sein Pfad zu fhem.save – Speicherort der Schaltzustände Verbose Loglevel (5=<u>alles</u> loggen, 1=nur Katastrophen) Die Definition des CUL mit dessen FHT-ID Port 8083 – das ‚normale‘ fhem-frontend Port 8084 – für iPhone und andere Geräte mit kleinem Bildschirm Port 8085 – für das iPad und andere Tablet-PCs autcreate - Festlegung, dass für Funktelegramme bisher unbekannter Geräte automatisch ein Gerät in fhem erzeugt wird, incl. Logfile und weblink Feiertage Bayern – siehe holiday2we Definition einer globalen dummy-Variable, die dem Raum ‚Wohnung‘ zugeordnet wird Ein per autcreate erzeugter und dann umbenannter Aktor mit Namen ez_Audio, Typ FS20, Hauscode 2332, Tastencode 51 und Attributen zu Modell und Raum Ebenfalls von autcreate erzeugt – die Logdatei zum Gerät ez_Audio Das nächste Gerät, selbe Abfolge wie zuvor
---	--

Zu Anfang der Datei finden Sie alle Attribute des (Pseudo-)Geräts `global`, siehe auch fhem-Konfiguration: Das besondere Gerät „Global“.

Der Speicherort dieser Datei und damit der Betriebssystem-Pfad zum Ordner „fhem“ ist als Attribut `modpath` angegeben:

```
attr global modpath /var/InternerSpeicher/fhem
```

(hier muss natürlich der zu Ihrem Gerät passende Pfad stehen! Wenn dieses Attribut in Ihrer `fhem.cfg` – Datei bereits eingestellt ist und `fhem` funktioniert, lassen Sie diese Angabe am besten unverändert).

Ebenso wichtig ist die Pfadangabe zur `fhem` Logdatei:

```
attr global logfile /var/InternerSpeicher/fhem/log/fhem-%Y-%m.log
```

(Hier muss natürlich der zu Ihrem Gerät passende Pfad stehen!). Durch diese Angabe wird eine separate Datei pro Monat pro Jahr angelegt. Das erlaubt eine zielgerichtete Archivierung, da die Logdateien je nach Anzahl und Art der verwendeten Geräte recht schnell recht groß werden können.

Neben der `fhem.cfg` spielt auch die Datei **`fhem.save`** eine zentrale Rolle. In dieser Datei werden die Schaltzustände aller Geräte gespeichert, wenn Sie `save` ausführen– so werden alle Geräte nach einem Neustart von `fhem` wieder mit ihrem letzten Schaltzustand angezeigt.

Der Speicherort dieser Datei ist festgelegt durch

```
attr global statefile /var/InternerSpeicher/fhem/log/fhem.save
```

In der Definition des CUL finden Sie die „FHT-ID“ des CUL, z.B. 2332. Die FHT-ID ist quasi der Hauscode des Hardwaresystems FHT, das parallel zu FS20 betrieben werden kann und zur Heizungssteuerung dient. Solange Sie keine FHT-Geräte betreiben (z.B. FHT80b, FHT8V, ...) setzen Sie die FHT-ID auf 0000, da sonst unnötige Funktelegramme gesendet werden.

Zum Betreiben von FS20-Komponenten wird der Hauscode ausschließlich in der Definition der FS20-Geräte verwendet, es ist nicht erforderlich, den FS20-Hauscode dem CUL zuzuweisen.

fhem.cfg und Includes

Da jede Geräte-Definition sowie alle Attribute, `notify`, `at`, `weblinks` usw. in der Datei `fhem.cfg` abgelegt werden, wird diese schnell lang und unübersichtlich. Mit `include` können Sie einzelne Konfig-Abschnitte in separaten Dateien speichern und beim Start von `fhem` zusammenführen.

Infos finden Sie in der [commandref](#) und einem [Wiki-Artikel](#).

Anhang – Das Hardwaresystem HomeMatic

Autor: Martin Pittner

Einleitung.....	61
Lesen des Dokuments	61
Philosophie.....	61
Voraussetzungen.....	62
Nomenklatur und Konventionen im Dokument.....	62
HM-Geräte – Allgemein.....	63
Hintergrund zu HM-Geräten – allgemein	63
Hintergrund zu HM-Geräten - Aufbau	64
Hintergrund zu HM-Devices und Kommunikation und Protokoll.....	67
HM-Devices /Empfangspegel Rssi.....	68
IO Devices – HMLAN, HMUSB, CUL, CUNO	69
<u>HMLAN, HMUSB</u>	69
<u>CUL, CUNO</u>	70
<u>Allgemein</u>	70
HM-Devices Pairen	71
HM-Kanäle anlegen.....	71
<i>Kanäle und Logfiles</i>	72
HM-Kanäle peeren	72
<u>Praxis:</u> HM-Kanäle peeren, peerChan	74
Sonderfälle des peerens.....	74
Konfiguration von HM-Entities mit FHEM (Register setzen).....	75
Register Lesen:	76
Register Schreiben	77
Register Sichern	78
Sammlung wichtiger Befehle für HM und Parameter in Kurzform	79
Attribute.....	79
Beispiele	79
StateMachines	80
Condition Table.....	81
Schalter	82
Das Gerät	82
Jalousie.....	82
Kanal Parameter (List 1)	83
Peer Parameter.....	83
State-Übergänge.....	84
Dimmer	86
Virtuelle Dimmer-Kanäle:.....	88
Rauchmelder – SD Sensor	90
Teammaster:.....	90
Raumklima Heizungssteuerung über – TC, VD und RHS Sensoren	91
Register und Einstellungen VD	92
Register und Einstellungen RHS.....	92
Register und Einstellungen TC	92
Register-Definitionen	93
Register Definition des Schalt Aktor – HM-LC-SW2	93
Register Definition des Blind Aktor – HM-LC-Blx	94
Register Definition des Dimmer Aktor – HM-LC-Dim1T.....	96

Register Definition der Fernbedienung RC12	98
Register Definition der Fernbedienung RC19	99
Hints, FAQs, Ideas.....	100
Schalter entprellen.....	100
Peer im Aktor löschen wenn der Sensor nicht mehr existiert	100
Peer Verhalten kopieren	100
Tips SystemÜbersicht, Hminfo.....	101
protoEvents.....	101
RSSI.....	102
peerXref	102
peerCheck	102
checkConfig.....	103
autoReadReg.....	103
param.....	103
HMinfo web Ansicht.....	103

Einleitung

Diese Anleitung ist eine Sammlung von Informationen, die ich in verschiedenen Foren und Webseiten gefunden habe sowie aus eigenen Tests.

Vielen Dank an alle, die bereits sehr hilfreich Infos bereitgestellt haben und mir die Inbetriebnahme erst ermöglichten.

Eingerichtet werden Dimmer, Jalousie-Steuerung, Lichtschalter und Fernbedienungen. Die Anleitungen, die ich bisher gefunden habe, waren sehr hilfreich, haben jedoch den Umfang meiner Installation, den ich mir vorstelle, nicht komplett abgedeckt. Einige der benötigten Funktionen könnten somit nicht realisiert werden.

Diese Anleitung erhebt in keinsten Weise Anspruch auf Vollständigkeit. Sie wurde nach bestem Wissen erstellt, kann jedoch fehlerhaft sein.

Die Benutzung und Anwendung ist auf eigene Gefahr und im eigenen Ermessen. Es wird keinerlei Garantie übernommen.

HomeMatic ist ein Produkt aus dem Hause eQ3. Diese Anleitung behandelt nur Produkte der Homematic Reihe.

Lesen des Dokuments

Die Anleitung beginnt mit der Beschreibung von HM und den Prinzipien wie ich sie verstehe. Außerdem wird die Arbeitsweise in FHem dargestellt. Für den Schnellstart - und damit man schon einmal etwas „sehen“ kann sind im hinteren Teil Beispiele aufgeführt – wer will kann also gleich mal probieren und Details sowie Grundlagen bei Bedarf nachlesen.

Philosophie

Als Zentrale wird eine AVM FritzBox 7390 eingesetzt. Die Anbindung zum HM Funknetz erfolgt über HMLAN Interface. Die FritzBox 7390 wird zur Konfiguration, Logging , Überwachung und Datensicherung benötigt. Für operationellen Betrieb kann man in vielen Fällen direkt Sensor und Aktor peeren. Eine Zentrale ist zum Schalten dann nur notwendig, wenn aufwendigere Aktionen benötigt werden wie tageszeitabhängige Schaltungen, uvm. Auch zum Mischen verschiedener Systeme wie HM und FS20 muss man über die Zentrale gehen.

Es wird angestrebt, die Sensoren und Aktoren möglichst direkt zu peeren. Dies hat Vorteile in Ausfallsicherheit und vor allem der Schaltgeschwindigkeit sowie Präzision der Steuerung.

Ansatz ist somit:

- Komponenten über FHEM zu konfigurieren
- Sensoren (speziell Taster / Remotes) direkt mit Aktoren zu peeren. Man kann diese Funktionen somit ohne Zentrale betreiben. Außerdem wird die Schaltgeschwindigkeit und Präzision erhöht.
- FritzBox 7390 wird für Logging verwendet.
- Spezielle Kommandos werden über FHEM gesteuert. Ein Beispiel wäre das Einbinden einer Wetterstation.

Voraussetzungen

Das HMLAN Interface ist eingerichtet. Hierzu kann man ggf. Die mitgelieferte Software von eQ3 verwenden.

Die Konfigurations-Software für Devices wird nicht verwendet, dies wird über FHEM gemacht. Ausnahme ist das einrichten von AES-keys.

Verwendet man statt HMLAN einen CUL, steht die Software HM Konfigurator nicht zur Verfügung. Die Konfiguration der Devices ist dann ausschließlich über FHEM möglich.

Nomenklatur und Konventionen im Dokument

Entity:	Bezeichnet eine Instanz, also ein Element, das mit define erzeugt wird. In CUL_HM ist es ein Kanal oder ein Device.
Device:	Der logische Aufbau von HM Geräten folgt einer festen Struktur. Device ist die logische Einheit in einem HM-Gerät welche im Wesentlichen für das Protokoll verantwortlich ist. Jedes Gerät hat ein Device das für alle Kanäle das Senden übernimmt und Empfangen.
Kanal:	Ist eine Funktionseinheit im Gerät. Jedes Gerät hat mindestens einen Kanal (irgend eine Funktion muss es ja haben). Jeder Button einer Fernbedienung ist ein eigener Kanal, jeder Lichtschalter auch. Kanäle in einem Device können unterschiedliche Funktionen haben, so hat ein TC einen Kanal für den Temperaturfühler, einen für die Heizungssteuerung und einen für die Verbindung zu Fensterkontakten. Alle Kanäle werden in FHEM automatisch angelegt, wenn das Gerät an FHEM angelernt wird. Es werden auch alle notwendigen Attribute gesetzt, die zum Betrieb notwendig sind.
Aktor:	Aktor ist ein ausführender Kanal, beispielsweise ein Lichtschalter oder Dimmer.
Sensor:	Sensoren ist ein Eingabekanal, der Daten erfassen und Trigger sendet. Auch Remotes und Buttons sind Sensoren. Ein Trigger könnte beispielsweise einen Tastendruck oder ein Bewegungsmelder Ereignis sein.
Pairen	Pairen ist das Anlernen des Device an die Zentrale. Dies ist dann für alle Kanäle im Gerät gültig. Pairen ordnet das Gerät der Zentrale unter. Bestimmte Aktionen können nach dem pairen nicht mehr von Device ausgeführt werden (peeren oder 'anlernen' von Kanälen ohne Zentrale). Es stehen aber immer noch alle Kommandos durch die Zentrale

zu Verfügung.

Es wird **strikt** empfohlen alle Devices mit der Zentrale zu pairen. Dies sollte die **erste** Aktion der Inbetriebnahme sein.

Peeren/Peer: Man kann Kanäle miteinander verknüpfen, das heißt peeren . Aktor und Sensor werden zu ‚peers‘. Der Sensor sendet dann Trigger direkt zum Aktor. FHEM kann den direkten Verkehr monitoren. Die Verbindung zwischen 2 Peers ist ein ‚Link‘, die Peers sind der Endpunkt.

Peeren kann man somit nur einen Sensor mit einem Aktor. Sowohl Sensoren als auch Aktoren können meist mehrere peers gleichzeitig haben.

Link Ein Link ist die Verbindung zwischen zwei Kanälen. Ein Link hat 2 Endpunkte, welche als Peer bezeichnet werden.

HMID: 6stellige HM ID (3 Byte hex) die aus dem FHEM log ermittelt werden kann. Die Nummer ist von Hersteller für jedes Gerät unveränderlich vorgegeben.

Die HMID ist die Adresse in allen Messages. Der User hat nur selten etwas mit der HMID zu tun da fast immer den Namen der Entity gearbeitet werden kann/soll.

HMIDch: Für Kanäle wird in FHEM die pseudo HMID genutzt die sich aus HMID und der Kanalnummer in hex zusammensetzt. Bei einem Schalter mit 2 Kanälen ist es 01 und 02. Bei einer Fernbedienung RC12 sind dies die Nummern von 01 bis 0B.

In FHEM sollte der User anstelle der HMID den Namen der Entity benutzen.

HMserial: Seriennummer die auf den Geräten und/oder der Schachtel aufgeklebt ist

HMIDlanIF HMID des HM LAN Interfaces oder besser des IO-device.

HMdev: Name der Device Entity.

HMch Name des channel Entity.

1. Tabelle: Konventionen

HM-Geräte – Allgemein

Hintergrund zu HM-Geräten – allgemein

Bei HM (und sicher nicht nur bei HM) werden Geräte nach Sensoren und Aktoren unterschieden.

Sensoren (auch Buttons und Taster sind Sensoren) senden Trigger aufgrund von Ereignissen, Schwellwerten oder zyklisch. Beispiele sind Fenstersensoren, die ein Öffnen, Kippen und Schließen des Fensters melden. Wasserstandsmelder haben einen Schwellwert, der ggf auch einstellbar sein kann. Fernbedienungen die das Drücken einer Taste erkennen senden den entsprechenden Trigger.

Aktoren ergreifen Aktionen aufgrund von Triggern. Es sind Lichtschalter (nicht der Taster sondern das ‚Relais‘) Heizungsventile, Dimmer... . Die Aktion für einen ankommenden Trigger wird **nicht** im Taster festgelegt sondern im Aktor.

HM bietet Geräte mit Funktionen – meist getrennt nach Sensoren und Aktoren. Ein Gerät kann mehrere Funktionen beinhalten – eine Fernbedienung bis zu 17 Tasten plus Display , ein Dimmer 2 Lampen bedienen. Funktionalen Einheiten im Gerät sind Kanäle. FHEM legt für jeden Kanal eine Entity an, eine steuer- und kontrollierbare, quasi-selbständige Einheit. Bei Geräten mit nur einem Kanal wird zur

Vereinfachung Kanal und Device in einer Entity zusammengefasst. Der User kann bei Bedarf/Geschmack den Kanal selbst erstellen um auch hier eine saubere Trennung zu erhalten.

Anmerkung: Es gibt Device, welche Sensoren und Aktoren beinhalten. Z.B. kann ein Thermostat TC von einem Fenster-Sensor Trigger empfangen, diese Verarbeiten und in der Rolle als Sender die ThermostatVentile steuern. Zusätzlich hat es noch einen eingebauten Sensor zur Temperaturmessung. Dies nur als Beispiel das die Trennung Sensor/Aktor nicht auf Geräte ebene passiert sondern eine Funktion des Kanals ist.

Um eine direkte Kommunikation zwischen HM Kanälen herzustellen werden die Sensoren und die Aktoren gepeert. D.h. man erstellt einen ‚Link‘, also eine Verbindung zwischen dem Sensor und dem Aktor. Ziel ist, dass der Aktor auf Trigger des Sensors reagiert. Umgekehrt erwartet der Sensor eine Rückmeldung des Aktors als Empfangsbestätigung. Peers werden im Aktor **und** Sensor eingetragen – die Beiden “kennen” sich nicht wirklich und stimmen sich auch nicht weiter ab. Ein Sensor hat keine Ahnung ob der Aktor ein Lichtschalter oder eine Heizungssteuerung ist, Hauptsache er antwortet auf den Trigger.

Geräte haben eine gemeinsame Hauptebene – das Device. Es betreibt die Kommunikation, steuert und koordiniert die Reihenfolge der zu sendenden Messages. Ferner kann (sollte) das Device an eine Zentrale angelernt - gepairt- werden. Ist eine Zentrale eingetragen wird sie über Änderungen der Zustände aller Kanäle am Laufenden gehalten. Praktisch wird eine Info-Nachricht an die Zentrale bei Zustandsänderung gesendet, z.B. Licht jetzt ‚ein‘.

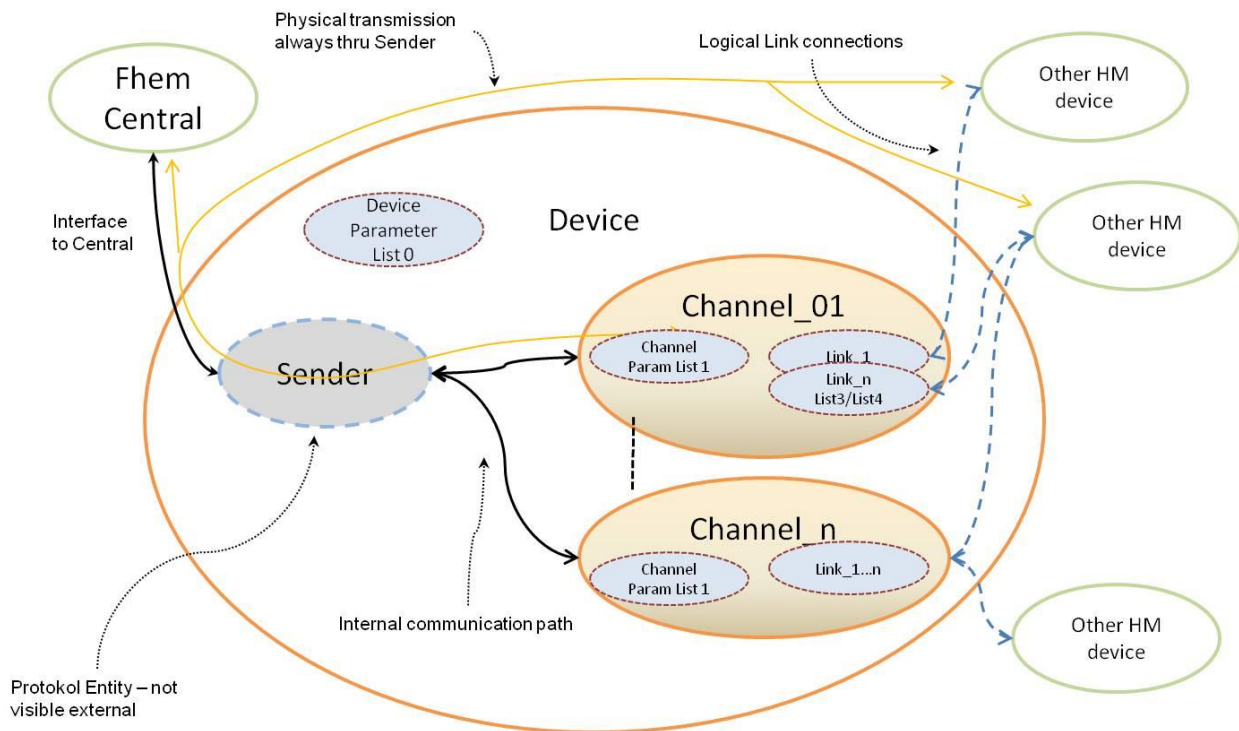
Unterscheiden muss man also zwischen

- ‚**pair**‘: eintragen einer Zentrale an ein **Device**
- ‚peer‘ oder ‚**peerChan**‘: erstellen einer ‚Verbindung‘ oder Link zwischen zwei HM **Kanälen**.

Hintergrund zu HM-Geräten - Aufbau

Dieser Abschnitt gibt eine Übersicht über die Konfigurations**struktur** der HM Devices geben. Evtl. sieht es beim ersten Durchlesen etwas kompliziert aus – man muss es auch nicht gleich komplett verstehen. Beim zweiten und dritten Durchlesen wird das dann schon ;-)

Das Bild soll die Kommunikationswege, Struktur und Parameterverwaltung illustrieren und deren Zusammenhänge und Bedeutung deutlich machen.



1. Abbildung: HM GerätArchitektur

HM Geräte beinhaltet das **Device** (Protokoll-Instanz), sowie je entsprechende Kanäle. Ein 2-fach Schalter besteht aus 2 Kanälen (Aktoren). Eine Fernbedienung mit 12 Tasten hat 12 Kanäle (Sensoren). Komplizierter ist es bei der 19 Tasten Fernbedienung mit Display. Die hat 17 Kanäle für Tasten (Sensoren) (2 weitere Tasten sind nicht extern verfügbar) und einen 18ten Kanal zum Steuern des Display (Aktor). FHem bildet die Struktur ab wie von HM vorgegeben.

Kanäle sind die Funktionseinheiten des Gerätes.

Nun gibt es noch ‚Links‘, logische Verbindungen, die durch peeren („koppeln“) von Kanälen entstehen. Wenn ein Sende-Kanal (z.B. Taste) einem Akteur-Kanal (z.B. Schaltsteckdose) zugeordnet wird, dann wird sowohl im Sendekanal als auch im Akteur-Kanal ein ‚Peer‘ erstellt. Der Peer ist der Endpunkt eines ‚Links‘. Im Akteur wird für diesen Peer die Steuerinformation hinterlegt. So legt der peer im Akteur fest, ob beim Drücken der Taste die Steckdose an, aus, wechselt oder nur für eine bestimmte Zeit angeht (Treppenhauusschalter).

Sowohl Akteur-Kanal als auch Sensor-Kanal kann mehrere peers verwalten. Praktisch heißt dies, ein Lichtschalter kann auf mehrere Buttons hören. Die Reaktion auf jeden Button wird unabhängig festgelegt!

Konfigurieren muss man im Wesentlichen 3 Ebenen

- **Device-Ebene:** Hier geht es um grundsätzliche Einstellungen des gesamten Geräts. Daten werden in der sogenannten List0 verwaltet. Hier habe ich bisher wenig gefunden, was einstellbar ist. Die gepairte Zentrale und das Sichtbarmachen interner Kanäle gehören zu den wenigen Ausnahmen.
- **Kanal-Ebene:** Es werden die Einstellungen je Kanal vorgenommen. Meist ist jedem Kanal eine List1 zugeordnet welche kanalindividuelle Daten wie Laufzeit des Rollo bei einem Rollo-Aktor

beinhalten. Auch wenn ein Gerät nur einen Kanal hat (wie z.B. Rollo-Aktoren) so bleibt die Aufteilung Gerät /Kanal doch bestehen.

- **Peer-Ebene:** Kanäle können mehrere Peers haben. Im Peer wird das Verhalten festgelegt. Auf der Sensorseite ist dies die Art der Kommunikation mit der Aktor-Peer (Burst oder AES, siehe später). Daten für Sensoren sind in List4 zu finden.

Beim Aktor-Peer ist das Verhalten des Kanals zum Verarbeiten eines Triggers meist in List3 einstellbar.

Jeder Peer in einem Kanal hat einen eigenen Registersatz, der erst beim Peeren angelegt und vorbelegt wird.

Hintergrund zu HM-Devices und Kommunikation und Protokoll

Wie vor beschrieben ist das Device für die Kommunikation verantwortlich. Wenn man Kommandos an ein Device sendet erwartet FHEM meist eine Antwort. Es kann vorkommen, dass keine Antwort oder eine negative Antwort (NACK) kommt. FHEM wiederholt die Nachricht ein paar mal und hört dann irgendwann damit auf. Was passiert ist kann man in den Variablen „prot“ sehen. Dargestellt werden:

protState	aktueller Zustand. CMDs_done: alles übertragen. Keine Probleme bei der LETZEN Übertragung CMDs_done_Events xx: Es gab xx Probleme bei der letzten Übertragung. Der Zustand ist vom Anwender zu prüfen. CMDs_processing...: Kommandos werden gerade abgearbeitet. CMDs_pending: Es warten Kommandos auf die Verarbeitung (siehe Transmit-Modi).
protCmdPend	Anzahl der Kommandos, die auf Verarbeitung warten
protLastRcv	Zeitpunkt der letzten Empfangs einer Nachricht
protSnd	Anzahl der gesendeten Messages und der letzte Zeitpunkt
protResnd	Anzahl wiederholter Messages. Dient nur zur Info, ist noch kein Fehler
protResndFail	Anzahl Fehler nach wiederholen. Die Nachrichten wurden nicht übertragen
protCmdDel	Anzahl der durch Protokollfehler gelöschten Kommandos, also Fehler
protNack	Anzahl von Device nicht akzeptierter Messages, also Fehler
protIOerr	Anzahl der Sendeprobleme aufgrund von IODevice Ereignissen, also Fehler
ProtIOdly	Anzahl der Verzögerungen aufgrund von IODevice Ereignissen, kein Fehler.
protTimedOn	Ein timer (on-for-timer oder ähnlich) wurde nicht korrekt übertragen und wiederholt.

2. Tabelle: Protokoll-Infos

Die Zähler sind kontinuierlich ab reboot. Sie sind manuell rücksetzbar mit:

```
set <name> clear msgEvents
```

Wenn Fehler angezeigt werden sollte man die letzten Kommandos kontrollieren.

Wichtig ist, die Modi der Kommunikation zu kennen welche ein Device unterstützt.

Um Energie zu sparen sind es meist batteriebetriebene Devices, die nicht immer auf Empfang sind. Je nach Aufgabe sind hierzu verschiedene Transmit-Modi vorgesehen:

normal	Device ist immer auf Empfang
config	Device ist nach Betätigen von Anlernen kommunikationsbereit. Nur dann kann FHEM an das Device senden. Kommandos an das Device werden zurückgehalten bis Anlernen erkannt wird. Der protState ist solange „pending“. Beispiel sind einige Remotes.
wakeup	Device wacht regelmäßig auf, dann kann gesendet werden. Die Zeit zwischen 2 "Aufwachen" ist unterschiedlich. Sie kann bis zu 24h betragen(Rauchmeldern). Messages an das Device werden bis zum Aufwachen zurückgehalten, der protState ist solange „pending“.
burst	Das Device kann mit einer burst-sequenz aufgeweckt werden. Für den User sieht dies aus wie 'normal'. Die Aufweck-sequenz weckt ALLE burst devices, die dann prüfen müssen, ob diese Nachricht für sie ist. Die Batterielaufzeit wird demnach für alle Geräte dieses Type reduziert. Burst belastet auch die Sende-kapazität des HMLAN. Burst sollte sparsam verwendet werden.
conditionalBurst	Entspricht burst. Es kann aber im Device ein- oder ausgeschaltet werden.
lacyConfig	Funktion und Ablauf noch nicht geklärt, wird aktuell wie config genutzt.

3. Tabelle: Transmit-Modi

Welchen Mode ein Model unterstützt kann man mittels HMinfo und dem Kommandos „models“ feststellen. Hierzu bitte im Commandref nachlesen.

Einige Geräte unterstützen mehrere Transmit-Modi. FHEM nutzt immer den schnellsten. Burst wird nur automatisch genutzt wenn das Device keinen alternativen Mode unterstützt

Alle zu sendenden Kommandos werden von FHEM so lange zurückgehalten bis das Device zur Verarbeitung bereit ist. protState zeigt den Zustand an: CMDs_pending. Devices, die nur auf „config“ reagieren arbeiten die Kommandos erst ab, wenn Anlernen gedrückt wird.

Noch ein Satz zur **Darstellung von Readings**: Wenn Daten verändert werden (Licht on) wird in den Readings der state auf „set_on“ angezeigt. Sobald eine Bestätigung von der HW vorliegt wird der Zustand dann ohne Prefix mit „on“ dargestellt. Dies gilt sinngemäß für alle Readings.

HM-Devices /Empfangspegel Rssi

Die Empfangsleistung wird in den RSSI Werten dargestellt. Es werden die Empfangswerte beider Seiten, also dem IO Device und dem HM-Device ausgegeben. Diese sollten etwa gleich hoch sein. Der Minimalwert ist -128. FHEM stellt min, max und Durchschnitt (avg) dar. Sollte der Wert zu niedrig sein oder zu sehr schwanken kann es ein Grund für Übertragungsprobleme sein.

IO Devices – HMLAN, HMUSB, CUL, CUNO

Damit FHEM sich mit einem HM-device unterhalten kann braucht man ein IO device. Es gibt die Geräte von HM (HMLAN, HMUSB) und „freie“, mit openSource SW betriebene (CUL,CUNO).

HMLAN, HMUSB

Die Devices von HM sind relativ komplex – und leider sind die Optionen nicht offengelegt. Was gibt es zu beachten:

AES: wird aktuell **nur** von den HM-Devices unterstützt. In FHEM kann man den Key nicht ändern. Will man AES nutzen muss man den key mit der PC-SW setzen. Danach muss man den key in den Attributen des HM-IO **hmkey**, **hmkey2** und/oder **hmkey3** setzen. FHEM setzt diese bei jedem restart in IO Device. AES wird vom HMLAN/USB automatisch gesendet.

KeepAlive: Die Geräte prüfen durch einen keep-alive Mechanismus das bestehen der Verbindung zur Zentrale. Das Gerät wird die Verbindung trennen und neu aufsetzen sollte für 30sec kein „keep-alive“ gekommen sein. FHEM macht dies automatisch alle 25sec. Es kann dennoch zu Problemen kommen wenn es zu Verzögerungen kommt, innerhalb oder außerhalb von FHEM. Mit

```
msgKeepAlive dlyMax:0.019 bufferMin:4
```

kann man erkennen, wie viel Puffer man hat und wie groß der maximale Delay eines keep-alive ist. Hier ist es 19ms Verzögerung – und mindestens 4 sec Puffer bis zum Disconnect.

Mit dem Attribut wdTimer kann man das Senden des keep-alive beschleunigen, also den Wert -bei Problemen – heruntersetzen.

```
attr wdTimer 25
```

Besser wäre natürlich den Grund der Verzögerung zu finden.

Sende-Kapazität: ein HMLAN/USB hat eine begrenzte erlaubte sende-rate während einer Stunde. Die liegt bei etwa 1600 Messages wobei man Burst events als sonder-Ereignis, Wiederholungen und Acks berücksichtigen muss. FHEM meldet die aktuelle Auslastung. Die Kalkulation beginnt erst nach einem Restart des Systems beginnt. FHEM ist nicht in der Lage den Wert aus HMLAN auszulesen oder ihn rückzusetzen. Aktuell lässt sich dieser Zustand nur durch Warten oder power-aus Rückstellen.

```
msgLoadEst 1hour:0% 10min steps: 0/0/0/0/0/0
```

msgLoadEst ist die aktuelle Auslastung sowie deren Aufspaltung in 10min Teile – alles nach FHEM Berechnung.

```
prot_Warning-HighLoad last 2013-11-13 07:32:29
prot_ERROR-Overload last 2013-11-13 07:32:29
prot_Overload-released last 2013-11-13 07:32:29
cond ERROR-Overload 2013-11-13 07:32:29
```

Die Zustände „HighLoad“ und „OverLoad“ hingegen werden in HM device selbst erzeugt und in FHEM dargestellt. High-load bedeutet, dass noch 10% Performance verbleiben bis Overload
Overload heißt, dass das Device nur noch empfängt – bis die „load“ entsprechend gesunken ist.

Automatisches ACK: HM IOs senden automatisch ACK zu den Devices. Ebenso wiederholen sie Nachrichten 3 mal falls keine Antwort kommt. Das ist eine effektive und schnelle Methode, einfache Übertragungsprobleme zu beheben.

CUL, CUNO

Oben erwähnte Details werden von CUL/CUNO nicht unterstützt. Das einschneidendste ist, dass AES hier nicht machbar ist.

Eine CUL/CUNO die für FHEM genutzt wird kann nicht gleichzeitig für andere Familien genutzt werden kann.

Allgemein

Zum Unterschied LAN oder USB gibt es aus FHEM-Sicht wenig zu sagen. Beides wird über Standard-System Interfaces des OS betrieben und unterliegt deren jeweiligen Möglichkeiten.

HM-Devices Pairen

Der erste Schritt beim Anmelden eines Gerät ist immer, das Devices mit FHEM zu pairen. Wie beschrieben bedeutet es, dass sich das Device der Zentrale unterordnet und entsprechende Kontrollen abgibt. Je nach Typ ist es leicht unterschiedlich. FHEM bietet mehrere Kommandos, alle mit dem gleichen Ziel:

1. `set <LANIf> hmPairForSec <Sekunden>`
Anlernen am Device auslösen.
FHEM erkennt die Nachricht und richtet das entsprechende Device eingetragen
2. `set <LANIfdefine> hmPairSerial <serialNo>`
`<serialNo>`: Seriennummer des anzulernenden Geräts
Einige Devices lassen sich so pairen
3. `define <name> CUL_HM <HMID>`
`set <name> pair`

pairen kann man mehrfach ohne negative folgen. Ein Device ist immer **nur an einer** Zentrale gepairt - oder eben nicht. Bei 1) und 2) sowie bei jedem Anlernen am Device legt FHEM alle fehlenden Kanäle an und setzt bzw korrigiert notwendige Attribute.

Achtung: Das Anlegen der Kanäle ist **kein** Beweis, dass korrekt angelernt wurde. Auch das Empfangen von Nachrichten nicht.

Um das Pairing zu prüfen **muss** man das Device **auslesen**. FHEM wird dies beim Anlernen selbständig versuchen. Sollte es nicht funktionieren hat das pairen nicht funktioniert oder ein Übertragungsproblem vorliegen. Man kann in jeden Fall noch einmal das auslesen probieren (beachte die transmit-modi verschiedener Devices):

```
set <name> getConfig
```

`getConfig` dauert ein paar Sekunden. Danach **muss** man das **Reading „pairedTo:“** prüfen. Als Wert **muss** die HMID des IO Devices stehen.

Resultat: Nach dem Pairen hat sich das Device der Zentrale „untergeordnet“. Es meldet Statusänderungen und erlaubt, dass die Zentrale Änderungen an der Konfiguration vornimmt. Zu pairen ist nur das Device, nie Kanäle. Kanäle sind als Element der Devices automatisch mit gepairt.

HM-Kanäle anlegen

Nach dem pairen/Anlernen hat FHEM alle Kanäle definiert. Hat ein Gerät nur einen Kanal wird dieser implizit in der Device-entity realisiert. Der Kanal wird also nicht separat erstellt. Die Device-Entity übernimmt in diesem Fall die Aufgaben von Device **und** Kanal 01. Will man trotzdem Kanal und Device trennen kann man den Kanal manuell definieren

```
define Licht_Wohnzimmer CUL_HM <HMID+ch>  
define Licht_Wohnzimmer CUL_HM 12345601 # Kanal Nummer 01
```

Kanäle habe immer eine Device. Sollte ein Device gelöscht werden betrifft dies auch ALLE seine Kanäle.

Zuordnung:

Kanäle eines Device sieht man in den Einträgen „channel_xx“.

Das Device eines Kanals ist im Parameter „device“ zu finden.

Die Namen von Devices und Kanälen kann der User frei vergeben. Hier sollte mit **rename** gearbeitet werden, dass die Software die Möglichkeit hat korrelierte Parameter nachzuziehen

```
rename <oldName> <newName>
```

Zusammenfassung, zu Merken:

- Ein Device ist erst gepairt, wenn man das Reading R-pairCentral lesen kann
- Kanäle können nur definiert werden, wenn es das Device gibt.
- Löscht man ein Device werden alle zugehörigen Kanäle gelöscht.
- Alle fehlenden Kanäle, Settings und Funktionen werden bei JEDEM Anlernen „nach-definiert“, und Parameter ggf. korrigiert.
- Bei Sender und Fernbedienungen entspricht jeder Button einen Kanal.

ACHTUNG: nach erfolgreicher Definition der Devices und Kanäle sollte man nicht vergessen die Konfiguration zu speichern.

```
save          # alle FHEM Parameter werden gespeichert
```

Kanäle und Logfiles

Mit dem Anlegen der Kanäle wird oft (einstellbar) auch ein Logfile erstellt. Meine Empfehlung ist, dies nicht zuzulassen oder zu beschränken. Viele Logfiles erschweren die Übersicht und kosten nicht unerheblich Performance. Daher sollte man Logfiles sinnvoll zusammenfassen. Alle Lichter, alles aus dem Wohnzimmer, oder wenigstens alle Tasten eines Geräts.

Man muss wissen, dass das reine Vorhandensein eines Logfiles Performance kostet, da bei jedem Event geprüft wird, ob geloggt werden muss.

HM-Kanäle peeren

Peeren erstellt einen Link zwischen 2 Kanälen. Das bedeutet, dass der Trigger eines Sensor-Kanals von Aktor-Kanal empfangen und verarbeitet wird, ohne Mitwirkung der Zentrale.

Peeren - nicht zu verwechseln mit pairen - kann man also Kanäle, keine Devices! Peeren kann man demnach auch nur Sensoren mit Aktoren. Ein Sensor kann Trigger senden welchen der Aktor verarbeitet. Aktoren senden keine Trigger und Sensoren verarbeiten keine.

Sensoren sind, wie beschrieben auch Buttons einer Remote. Sie erkennen das Drücken einer Taste.

Wichtig ist, dass nicht der Sender festlegt, was der Aktor tun soll, sondern der Aktor. Dieser legt die Aktion separat für jeden gepeerten Sensor-Kanal fest.

Peeren für Sensoren bedeutet, dass er eine Empfangsbestätigung anfordert, wenn die Aktion beendet ist. Ggf. wird der Trigger wiederholt, sollte kein ACK kommen. Ein Sensor kann mit mehreren Aktoren gepeert werden. Es wird je Peer der Übertragungsmodus im Sensor festgelegt werden, also AES sowie burst (siehe Register).

Peeren für Aktoren bedeuten, dass er eine Aktionsabfolge für genau diesen Peer definiert. Der Aktor legt für jeden gepeerten Sensor einen Registersatz an - und definiert somit ein Verhalten je peer.

Beispiel : ein Button zum Einschalten, einer zum Ausschalten oder/und einen zum Toggeln.

Der Aktor legt beim Peeren ein Default-Verhalten an. Viele Aktoren unterstützen 2 Default- Varianten:

- Wird nur **ein** Kanal gepeert (in einem Kommando!) realisiert der Aktor einen „toggle-peer“. Das Licht wird beim ersten Trigger eingeschaltet oder hochgedimmt. Beim nächsten Trigger wird ausgeschaltet oder runter gedimmt.
- Werden **2 Kanäle auf einmal** gepeert legt der Aktor 2 Peers-defaults an. Einen zum Einschalten und Hochdimmen und den 2. zum Ausschalten und Runter dimmen.

Einige Sensoren (i.a. Taster und Fernbedienungen) können lange oder kurze Trigger schicken. Der Aktor kann dies unterscheiden und legt ein komplett separates Verhalten für jeden fest.

Andere Sensoren (Fensterkontakte, Bewegungsmelder, Fühler) senden immer einen kurzen Trigger. Sie liefern aber zusätzlich einen Messwert, der vom Aktor gefiltert werden kann. Der Aktor entscheidet anhand der Condition Table (CT) ob der Trigger gültig ist oder nicht, also ob eine Aktion ausgeführt wird oder eben nicht.

Die Messwerte sind zwischen 0 und 200 und repräsentieren 0-100% in 0.5% schritten. 200 entspricht also 100%. 3-State Sensoren (z.B.Fensterkontakte) schicken

0 (=0%) für zu
100 = 50% für gekippt
200 = 100% für offen

Sinngemäß ist dies in vielen anderen Sensoren ebenso zu finden.

Wenn ein Peer angelegt ist kann das Verhalten nachträglich verändert werden. Durch Setzen der Register kann man jedes dieser Verhalten umbauen und aus jedem Toggel eine Einschalter bauen oder einen Treppenhausschalter.

Interne Peers: Ein Kanal kann interne Peers haben. Beispiele sind die meisten Schalter oder Dimmer an welchen direkt ein mechanischer Schalter angeschlossen werden kann, oder ein Bedienschalter direkt eingebaut ist. Die Firmware (FW) des Kanals behandelt diesen wie einen externen Peer, nur ist er automatisch eingerichtet. Es stehen die identischen Register wie bei 'normalen' Peers zu Verfügung. Diese Peers sind nicht automatisch sichtbar, können aber sichtbar geschaltet werden. Hierzu ist das Register **intKeyVisib** auf **visib** zu setzen (siehe HM-Konfiguration) - siehe auch getConfig. Danach kann man das Verhalten der „eingebauten Peers“ auslesen und verändern wie von jedem anderen Peer auch. Die internen Peers werden **selfxx** benannt (self01, self02,...).

Praxis: HM-Kanäle peeren, peerChan

Beim peeren steht der **Sensor-Kanal immer vorne**.

Das Kommando peerChan ist das Einzige, dass gleichzeitig 2 Geräte konfiguriert.

```
set <senChan> peerChan 0 <aktChan> single|dual set|unset actor|remote|both
```

Single|Dual legt fest ob 1 oder 2 Kanäle gepeer werden sollen. Siehe dazu den **Peeren für Aktoren** in Hintergrund

Set|unset selbserklärend: setzen oder löschen eines peerings

actor|remote|both erlaubt nur ein Ende des Links zu bearbeiten, nur in Aktor oder nur im Sensor zu setzen. Default ist both. Es kommt zum Einsatz wenn man einen Aktor resetet und neu peeren will. Die Remote (Sensor) müsste man ggf. aus der Wand ausbauen um anlernen zu drücken, obwohl die Remote kein Problem hatte. Also setzt man den Link mit Actor auf. In der Remote bleibt alles wie es war.

Die Liste der peers wird im HM-Gerät verwaltet und permanent gespeichert, also über Spannungsausfall hinweg. Um sie einzusehen muss man sie Rücklesen, am einfachsten mit:

```
set <entity> getConfig
```

Wenn man getConfig auf ein Device ausführt werden auch alle Daten der Kanäle gelesen. Es werden auch alle Register des Geräts ausgelesen.

Die Liste alle peerIDs wird im Attribut **peerIDs** gespeichert und ist im Reading **peerList** im Klartext zu sehen. Das Attribut peerIDs hat keine steuernde Wirkung, es wird nur hier abgelegt um es in FHEM speichern zu können.

Das Attribut peerIDs und das Reading peerList wird nach jedem getConfig upgedated. Das Attribut peerIDs sollte vom User nur in Ausnahmefällen geändert werden. Eine peerID „00000000“ wird intern von HM und FHEM genutzt und sollte nicht gelöscht werden.

ACHTUNG: Man kann keine Peers anlegen indem man peerIDs editiert!!

Beachte: Die Anzahl der Peers eines Kanals ist begrenzt. Will man mehr Kanäle peeren wird dies vom Gerät zurückgewiesen. Der evtl. vorhandene peer muss dann erst gelöscht werden.

Sonderfälle des peerens

Rauchmelder können zu Teams zusammengefasst werden. Des erreicht man ebenfalls mit peerChan. Man muss eine HMId für das Team wählen und alle Rauchmelder in dieses Team eintragen. Die HMId kann die eines der SDs sein oder ein virtuellen Kanal. Sollte die HMId eines SD gewählt werden ist dieser SD mit sich selbst zu peeren!

```
set <Team> peerChan 0 <SD> single set|unset actor
```

Einige Geräte haben sowohl Kanäle für sensoren als auch Aktoren, so z.B. ein TC. Der Climate-Kanal ist ein Sensor (sendet trigger) und der WindowRec-Kanal ist ein Aktor, empfängt trigger von Fenster-kontakt.

```
set <tc_Climate> peerChan 0 <VD> single set|unset actor|remote|both  
set <RHS> peerChan 0 <tc_WindowRec> single set|unset actor|remote|both
```

Konfiguration von HM-Entities mit FHEM (Register setzen)

Es wird die Konfiguration der HM Geräte mittels FHEM beschrieben werden. Die Konfiguration von FHEM selbst ist hier nicht beinhaltet sondern die der HM-Geräte in deren flash.

Die HM Geräte werden über Register konfiguriert. Das sind nicht flüchtige Variablen/Einstellungen im Gerät. Sie sind in Listen eingeteilt und den Funktionseinheiten im Gerät zugeordnet – dem Device, dem Kanal oder dem Kanal/Peer.

Es werden (bisher) 9 Listen unterschieden, wobei nicht jedes Gerät alle Listen unterstützt.

- | | |
|-----------|--|
| Liste 0 | Einstellung der Device-Ebene. Z.B. kann man hier die Tastenbeleuchtung einer RC12 einstellen. List0 gibt es demnach einmal in einem Gerät . |
| Liste 1 | Parameter zur Steuerung eines Kanals . Z.B. RC12 „long press time“ oder Jalousie „hoch“ und „runter“ Fahrzeiten. List1 gibt es maximal einmal je Kanal. |
| Liste 2 | Wird von Repeatern genutzt um beispielsweise Listen der HMIds zu verwalten, die wiederholt werden sollen. |
| Liste 3 | Für jeden Peer eines Aktor-Kanals wird eine List3 angelegt. Es werden die Aktionen und Parameter festgelegt die ablaufen sollen, wenn ein Trigger dieses Peers verarbeitet wird. Es wird die Treppenhausfunktion, An- oder Ausschalter, oder Toggle-Funktion definiert und parametrisiert. List3 gibt es nur bei Aktoren. |
| Liste 4 | Dies ist das Pendant zu List3 für Sensoren . Es beinhaltet, wie ein Sensor mit einem Peer kommunizieren soll, z.B. RC12, „Peer needs burst“ oder „expect_AES“. |
| Liste5 -9 | gibt es bei Wetterstationen und Thermosteuerung . Hier werden u.a. Regelwerte einer Woche hinterlegt. |

List 3 und List4 werden angelegt, wenn Kanäle gepeert werden. Wenn nichts gepeert ist gibt es diese Listen auch nicht.

MERKE: Die Register sind Konfigurationen. Hier sind **keine operationellen Zustände** oder Stati abzufragen.

MERKE: Die Register sind im Gerät gespeichert. FHEM kann die Werte lesen und setzen. Will man sicher sein, was im Gerät gerade gespeichert ist muss man die Daten wieder lesen (getConfig).

Register Lesen:

Auslesen der Konfiguration und in FHEM darstellen kann man mit

```
set <devName> getConfig
```

Führt man es auf ein Device aus werden auch all seine Kanäle gelesen. Führt man es auf eine Kanal-Entity aus wird nur die Kanal-entity upgedated (geht etwas schneller). Die Ausführung dauert, je nach Anzahl der Kanäle und Parameter. Die vollständige Abarbeitung kann man kontrollieren in dem man die Einträge in Protokoll sieht.

Resultat:

Gelesen werden alle Register und Peers. Dargestellt werden die Ergebnisse im web-interface. Mit dem Attribut „expert“ lässt sich die Sichtbarkeit ein wenig steuern.

Alternativ kann man alle Register sehen mit:

```
get <name> reg all
```

ACHTUNG: Beides sind nur Darstellungen der Daten in FHEM. Der User muss selbst sicherstellen, dass die Daten aktuell sind. Ggf. muss noch einmal gelesen werden (vom Gerät nach FHEM übertragen - getConfig) und ggf. das web-frontend refreshed.

Beispiel:

```
LichtFH type:switch -
list:peer      register      :value
  1:           sign          :off
  3:FB_Btn_11  lgActionType   :jmpToTarget
  3:FB_Btn_11  lgCtDlyOff     :geLo
  3:FB_Btn_11  lgCtDlyOn      :geLo
  3:FB_Btn_11  lgCtOff        :geLo
  3:FB_Btn_11  lgCtOn         :geLo
  3:FB_Btn_11  lgCtValHi      :100
  3:FB_Btn_11  lgCtValLo      :50
  3:FB_Btn_11  lgOffDly       :0 s
  3:FB_Btn_11  lgOffTime      :111600 s
  3:FB_Btn_11  lgOffTimeMode  :absolut
  3:FB_Btn_11  lgOnDly        :0 s
  3:FB_Btn_11  lgOnTime       :111600 s
  3:FB_Btn_11  lgOnTimeMode   :absolut
  3:FB_Btn_11  lgSwJtDlyOff    :off
  3:FB_Btn_11  lgSwJtDlyOn     :on
  3:FB_Btn_11  lgSwJtOff      :dlyOn
```

Einige Register haben einen prefix „lg“ = Long oder „sh“ = Short. Zu wissen ist, dass einige (viele) remotes zwischen langem und kurzem Tastenruck unterscheiden. Der Aktor an dies auch. Die verfügbaren Register sind quasi identisch. Somit kann man je Button 2 Reaktionen am Aktor stimulieren, indem man kurz oder lange drückt. Man kann auch nur auf kurz oder nur auf lang reagieren und den jeweils anderen ignorieren.

Was kann man ändern:

die Register, welche ein einer Entity zu ändern sind kann man mit

```
get <name> regList
```

erfragen. Wie beschrieben ist die Liste je Entity – ist also unterschiedlich für Device und Kanal und kann auch von Kanal zu Kanal unterschiedlich sein. Hier erhält man auch die Information, ob ein Register zum Registersatz eines Peers gehört, ob beim Setzen also ein Peer angegeben werden muss. Ferner erhält man Informationen über Wertebereich, Einheit und ggf. Optionen. Optionen sind angegeben, wenn nicht nur Zahlen sondern mit ‚Namen‘ gearbeitet wird, also mit „on“ und „off“.

Beispiel Schalter Kanals:

```
get LichtFH regList
```

```
list:      register | range      | peer      | exp | description
1: sign      |- to -      |           |     | signature (AES) options:on,off
3: lgActionType |- to -      | required |     | options:toggleToCntInv,off,toggleToCnt,jmpToTarget
3: lgCtDlyOff  |- to -      | required | exp | Jump on condition from delayOff options:geLo,between,outside,ltLo,geHi,ltHi
3: lgCtDlyOn   |- to -      | required | exp | Jump on condition from delayOn options:geLo,between,outside,ltLo,geHi,ltHi
3: lgCtOff     |- to -      | required | exp | Jump on condition from off options:geLo,between,outside,ltLo,geHi,ltHi
3: lgCtOn      |- to -      | required | exp | Jump on condition from on options:geLo,between,outside,ltLo,geHi,ltHi
3: lgCtValHi   |0 to 255    | required | exp | Condition value high for CT table
3: lgCtValLo   |0 to 255    | required | exp | Condition value low for CT table
3: lgMultiExec |- to -      | required |     | multiple execution per repeat of long trigger options:on,off
3: lgOffDly    |0 to 111600s| required | exp | off delay
3: lgOffTime   |0 to 111600s| required | exp | off time, 111600 = infinite
3: lgOffTimeMode |- to -      | required | exp | off time mode options:minimal,absolut
3: lgOnDly     |0 to 111600s| required | exp | on delay
3: lgOnTime    |0 to 111600s| required | exp | on time, 111600 = infinite
```

Register Schreiben

Schreiben von Registern kann man mit regSet:

```
set <name> regSet <value> [<peer>]
set LichtFH regSet lgActionType jmpToTarget FB_Btn_01
set LichtFH regSet lgOnTime 100 FB_Btn_01
set LichtFH regSet sign off
```

Register die Peers zugeordnet sind benötigen die Angabe des Peers. Der wird als letzter Parameter im Kommando angegeben.

Einige Werte sind nur Teile von Bytes. FHEM ändert nur den entsprechenden Teil – HM erlaubt aber nur das Schreiben ganzer Bytes. FHEM benötigt also den Rest des Bytes, der nicht geändert werden soll um den zu schreibenden Wert zu errechnen. Will man so ein Register ändern müssen die Register in FHEM vorliegen. Wenn dies nicht der Fall ist wird FHEM nicht schreiben und den User auffordern, die Register zu lesen.

Nach dem Ändern wird empfohlen die Daten noch einmal zu lesen (siehe auch autoReadReg).

Register Sichern

Wenn man Einstellungen fertiggestellt und getestet hat kann man die Daten sichern:

```
Set <devName> saveConfig <file>
```

Das Kommando sollte man immer auf ein Device ausführen. Es werden alle Register und Peers aller untergeordneten Kanäle gesichert. Das File ist kumulativ, alle Änderungen werden an das Ende angehängt, nie überschrieben.

ACHTUNG: Gesichert werden nur die Daten, die in FHEM vorliegen. Es empfiehlt sich vorher ein Lesen und dann eine Prüfung der Protokollereignisse um sicherzustellen, dass die Daten aktuell sind.

Diese Sicherung hat nichts mit „save“ von FHEM zu tun. Hier geht es um die HM Geräte Daten, save kümmert sich um die FHEM Parameter.

Rückschreiben der Konfiguration

Hat man eine Sicherung kann man Einstellungen kann man es auch wieder zurückschreiben. Dazu muss man das Sicherungsfile in einem text-editor öffnen. Man sucht die entsprechenden Daten und kann diese per copy/paste in einen Terminal ausführen lassen.

Zusammenfassung:

Ein sinnvoller Ablauf könnte also sein

```
get <HMdev> regList          # Übersicht Device
get <HMch> regList           # Übersicht Kanal
set <HMdev> getConfig        # Auslesen, Kanal und Device gleichzeitig
```

Warten bis die Daten gelesen sind, evtl. Fehlermeldungen beachten, siehe Protokollabschnitt

```
get <HMdev> reg all          # aktuelle Werte ansehen
get <HMch> reg all           #
set <HMch> regSet driveUp 64.3 # setzen eines blindaktor Registers
set <HMch> regSet shOnDly 10 FB_Btn01 # Register eines Links
```

Überprüfen:

```
set <HMdev> getConfig        # Daten werden gelesen
get <HMch> reg all           # Auch Veränderung sollte jetzt lesbar sein
set <HMdev> saveConfig myfile # sichern
```

ACHTUNG: Sender-Devices verarbeiten diese Kommandos nicht freiwillig. Deshalb werden die Kommandos in FHEM gequeued und erst übertragen sobald man das Gerät in den Anlernmode versetzt. Konkret betroffen sind alle Devices welche nur Configmode unterstützen. Siehe Abschnitt Protokoll

Sammlung wichtiger Befehle für HM und Parameter in Kurzform

Kommandos:

```
set <Device> getConfig          # auslesen der Daten aus dem HM Device und der seiner
                                Kanäle

set <channel> peerChan....      # direktes „verlinken“ von 2 Kanälen

get <Device|channel> regList     # liste aller Register, die von dieser entity
                                unterstützt werden. Muss für Kanal und Device
                                gesondert ausgeführt werden. Es werden Wertebereiche
                                und Optionen angegeben, nicht die aktuellen Inhalte!

get <Device|channel> reg all     # alle Daten, die für diese Entity gelesen
                                (getConfig) wurden. Muss für jeden Kanal separat
                                ausgeführt werden. Man erhält die aktuellen Inhalte,
                                wenn vorher ein getConfig gemacht wurde.

get <Device> saveConfig <file># speichern der gelesenen Daten in ein File. Daten
                                werden für das ganze Device incl. Kanälen
                                geschrieben. Die Daten können dann ggf. mir regBulk
                                wieder geschrieben werden.

set <channel> peerBulk ...       #schreiben der gespeicherten Links in den channel

set <Device|channel> regBulk ... #schreiben der gespeicherten Daten in das Device oder
                                den Channel
```

Attribute

Folgende Attribute sollte automatisch gesetzt werden und sind nicht vom User.

model, subtype, serialNr, firmware, peerIDs

Der User sollte folgende **Attribute** berücksichtigen:

autoReadReg	Empfehlung es bei allen Devices auf '3' zu setzen. Ausgenommen sind Devices, die nur mit ‚config-mode‘ ausgelesen werden können. Das sind die meisten remotes und push-buttons. Kanäle sollte das Attribut nicht erhalten, sie werden mit den Devicedaten zusammen gelesen. Es reicht, dies am Device einzustellen.
expert	hiermit kann man die Darstellung der Register im webinterface grob steuern. Es reicht, dies am Device einzustellen.
webCmd	hiermit werden die im Webinterface dargestellten Kommandos gesteuert. Für HM werden ggf. einige Kommandos vorgewählt. Diese kann der User an seine Bedürfnisse anpassen
actCycle	dient der Prüfung, ob ein Device noch ‚lebt‘. Man gibt ein Intervall vor in dem sich das Device mindestens einmal melden muss. Für Geräte die sich automatisch melden wird der Wert entsprechend vor besetzt. Der User kann dieses Attribut bei allen Devices setzen, und erhält die Statusmeldung, wenn das Device die Regeln verletzt.

Beispiele

Hier folgen ein paar Beispiele, wie man Geräte programmieren kann. Bei allen wird vorausgesetzt, dass das pairing mit FHEM stattgefunden hat und entsprechende peers eingetragen sind. Hier wird auf die Programmierung der Register eingegangen und auf bekannte Besonderheiten.

StateMachines

Aktoren arbeiten in Zustandsmaschinen. Einige sind mit Bildern illustriert. Hier eine (sehr) kurze Einführung in Zustandsmaschinen an Beispiel des Lichtschalters: Der Kanal befindet sich immer in einem Zustand, von denen der Lichtschalter 4 hat:

„on“: Licht ist an

„off“: Licht ist aus

„dlyOn“: Licht ist noch aus, warten auf an

„dlyOff“: Licht ist noch an, warten auf aus.

Der Kanal geht nur in einen neuen Zustand, wenn ein passender Trigger kommt. Das könnte das Ablaufen eines Timers oder ein externen Trigger sein. Ohne Trigger/Timer wird der Zustand beibehalten:

on->warten bis OnTime vorbei ist, gehe dann nach dlyOff

dlyoff->warten bis dlyOffTime vorbei ist, gehe nach Off

off->warten bis OffTime vorbei ist, gehe nach dlyOn

dlyOn->warten bis dlyOnTime vorbei ist, gehe nach dlyOn

Die Timer OnTime und OffTime, wenn auf max gesetzt, bedeuten „unendlich“, der Kanal wird also den Zustand NIE verlassen.

Wenn ein gültiger Trigger von einem Peer kommt, egal welcher, wird der Kanal seinen Zustand ändern UND die Timer versenden, die hier definiert sind.

SwJtOn : wenn der Kanal im Zustand On ist soll er nach... gehen

SwJtOff :wenn der Kanal im Zustand Off ist soll er nach... gehen

SwJtDlyOn : wenn der Kanal im Zustand DlyOn ist soll er nach... gehen

SwJtDlyOff : wenn der Kanal im Zustand DlyOff ist soll er nach... gehen

und gehen kann er dann nach

no keine Reaktion, bleib wo du bist

dlyOn gehe sofort nach dlyOn

on gehe sofort nach On

dlyOff gehe sofort nach dlyOff

off gehe sofort nach off

und benutze die Timer, die zu diesem peer/long-short gehören.

Eine Zustandsänderung von „on“ nach „on“ startet den timer neu! Wenn man also einen Treppenausschalter hat mit 20sec on und nach 10 sec noch einmal drückt kommt es darauf an:

SwJtOn = on : Die onTime wird noch einmal gestartet, licht geht nach gesamt 30sec aus

SwJtOn = no : Die onTime wird nicht neu gestartet, licht geht nach gesamt 20sec aus

Condition Table

Die Register „CT“ gehören zur ConditionTable. Bei ConditionTable ist relevant, wenn trigger mit „Werten“ gesendet werden. Dies ist beispielsweise der Fall bei Bewegungsmeldern.

Ein bewegungsmelder sendet immer, wenn eine Bewegung erkannt wird einen Triggern an seine Peers. Dies geschieht unabhängig von der Helligkeit. Die Helligkeit wird aber an den Peer mit übertragen. Es ist dann (wiederum) der Aktor, der die Helligkeit auswerten muss. Dazu hat er die Condition-Table.

Ein Aktor hat 2 Werte, den high und den low Wert (CtValHi, CtValLo). Die beiden Werte sind gleichwertig, high und low sind nur Namen, keinen Bedeutungen!

In den Registern (je einmal für long/short und jeden peer)

CtDlyOff	Abfrage wennstatemachine im Zustand „dlyoff“
CtDlyOn	Abfrage wennstatemachine im Zustand „dlyon“
CtOff	Abfrage wennstatemachine im Zustand „off“
CtOn	Abfrage wennstatemachine im Zustand „on“

kann man festlegen, ob ein Trigger gültig ist.

geLo	größer als low
between	zwischen low und high
outside	kleiner low oder größer high
ltLo	kleiner low
geHi	größer high
ltHi	kleiner high

Die Werte sind in Prozent und in 0.5% Schritten. Somit sind Werte von 0 bis 200 (0-100%) üblich.

Man kann also einen Bewegungsmelder mit mehreren Aktoren peeren. Ab welcher Helligkeit ein Aktor das Licht einschaltet ist in jedem Aktor festzulegen, auch wenn der selbe Bewegungsmelder genutzt wird.

Schalter

Verwendet wird ein **HM-LC-SW2-FM**.

Es handelt sich um einen 2 Kanal Unterputzschalter. FHEM richtet demnach ein Device und 2 Kanal-Entities ein. HM ermöglicht Funktionen der Kanäle wie

- Lichtschalter
- Treppenhausschalter
- Invers Treppenhaus: Licht aus, geht wieder an nach best. Zeit
- Blinken
- Eigene Kreationen

Das Gerät

Entities sind (Namen mit rename geändert)

Licht_Switch1 #Device

Licht_Wohn # Kanal 1

Licht_Flur # Kanal 2

Die Schalter wurden gepeert mit

```
set FB1_Btn_01 peerChan 0 Licht_Flur single set # ein toggleschalter
set FB1_Btn_03 peerChan 0 Licht_Wohn dual set # 2 Buttons werden gepeert, einer für
                                         ein, einer für aus
```

Vorgabe: Licht_Flur soll bei kurzem Druck nach 10 sec abschalten, bei langem Druck toggeln, also Dauer an oder aus

```
set Licht_Flur regSet shOnTime 10 FB1_Btn_01
                        #wir sind sparsam: kurzer Druck auf Btn1, 10 sec Licht
```

```
#Lang Button 1:
set Licht_Flur regSet lgActionType jmpToTarget FB1_Btn_01#
set Licht_Flur regSet lgSwJtOn      dlyOff FB1_Btn_01      #
set Licht_Flur regSet lgSwJtOff     dlyOn FB1_Btn_01       #
set Licht_Flur regSet lgSwJtDlyOn   no FB1_Btn_01          #
set Licht_Flur regSet lgSwJtDlyOff  no FB1_Btn_01          #
set Licht_Flur regSet lgOnTime      111600 FB1_Btn_01      # max
set Licht_Flur regSet lgOffTime     111600 FB1_Btn_01      # max
set Licht_Flur regSet lgOnDly       0 FB1_Btn_01           #
set Licht_Flur regSet lgOffDly      0 FB1_Btn_01           #
```

Entsprechend kann man programmieren

- Einschaltverzögerung (OnDly verändern)
- Ausschaltverzögerung(OffDly verändern)
- Einschaltdauer (Treppenhausfunktion)
- Ausschaltdauer (wieder einschalten nach Dauer)

Jalousie

Verwendet wird ein **HM-LC-BI1-PBU-FM**. Dieser ist nach eQ-3 Angaben von der Steuerung identisch dem HM-LC-BI1-PB-FM.

Kanal Parameter (List 1)

Man kann dem Aktor Befehle geben wie z.B. 30% schließen. Da der Aktor aber keinen Sensor hat und somit auch nicht wissen kann, wann der Rollo oben oder unter ist kann er auch prinzipiell nicht wissen, wann 30% geschlossen ist.

Erreicht wird dies, indem man dem Schalter die Fahrzeit des Rollos programmiert. Hierzu wird die Fahrzeit „hoch“, „runter“ und (relevant für Jalousien) die Wendezeit der Lamellen

„Change“ programmiert. Diese werden in **Liste 1** Verwalter – siehe Anhang.

Einfacher ist

```
set <HMchn> regSet driveUp <time>
set <HMchn> regSet driveDown <time>
set <HMchn> regSet driveTurn <time>
```

Beispiel: Programm runter 25,6sec, hoch 51,3sec, Motor entladen 0,5 sec.

```
set Rollo1 regSet driveUp 51,3
set Rollo1 regSet driveDown 25,6
set Rollo1 regSet driveTurn 0,5
```

Achtung: driveTurn sollte nie auf '0' gesetzt werden, wenn Motoren gesteuert werden. Die vorgegebene Zeit stellt ein wie lange der Aktor wartet um dem Motor das Entladen zu erlauben. Default sind 0.5s was für viele Motoren reichen sollte.

Peer Parameter

Ein RolloAktor hat mehr Zustände als ein Lichtschalter, die Funktion ist aber immer noch gleich:

Zustände

On/Off: Rollo ist im Ruhezustand, offen, zu oder teil geöffnet. Der Zustand wird nach On/Off_Time automatisch verlassen (schwarzer Pfeil) (Per Default ist die Zeit auf „Unendlich“ gesetzt)

On/Off Delay: ein Wartezustand ohne Aktionen. Die Zeit ist in On/OffDelay_Time festgelegt

RefOn/Off: ist zur Berücksichtigung der Entladezeit der Motoren und Kondensatoren. Wenn die Fahrriichtung geändert wird ist es notwendig eine Wartezeit einzuhalten. Ansonsten läuft man Gefahr das Device zu zerstören, da der entstehenden hohe Strom den Schaltkontakt durchbrennt oder verschweißt.

RampOn/Off: In diesem Zustand wird das Rollo hoch bzw. runter gefahren. Wenn 100% erreicht sind wird automatisch in den Zustand On/Off übergegangen.

State-Übergänge

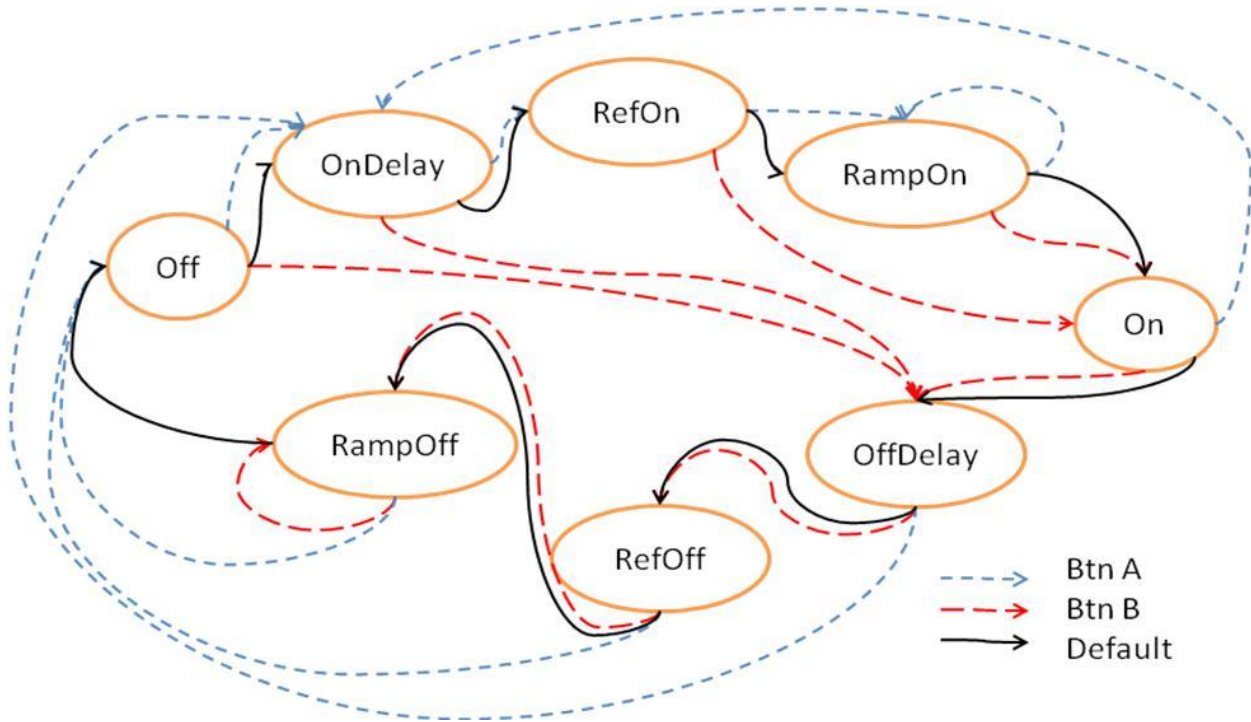
Nach abarbeiten eines Zustandes wird in den nächsten Zustand (gemäß schwarzem Pfeil) übergegangen.

- Kommt ein Trigger kommen wird nach JT Tabelle der nächste Zustand angesprungen. Sind 2 Tasten angelernt wird jedem eine eigene JT Tabelle zugewiesen (blauer oder roter Pfeil)
- Per Default ist kein Unterschied zwischen 'long' und 'short' in den JT Tabellen. Der Unterschied wird durch 'multiexecution' und die Dauer in MAX_TIME_FIRST_DIR erreicht.
Im Zustand 'ON' bei einem Trigger "Short" (rote Pfeilrichtung) gibt es einen Übergang nach 'OffDelay'->'RefOff'->'RampOff. MAX_TIME_FIRST_DIR wird nicht benutzt ('FF'). Nach Ende der Fahrzeit (allgemeine Einstellungen REFERENCE_RUNNING_TIME_TOP_BOTTOM) geht es in den Zustand 'OFF'.

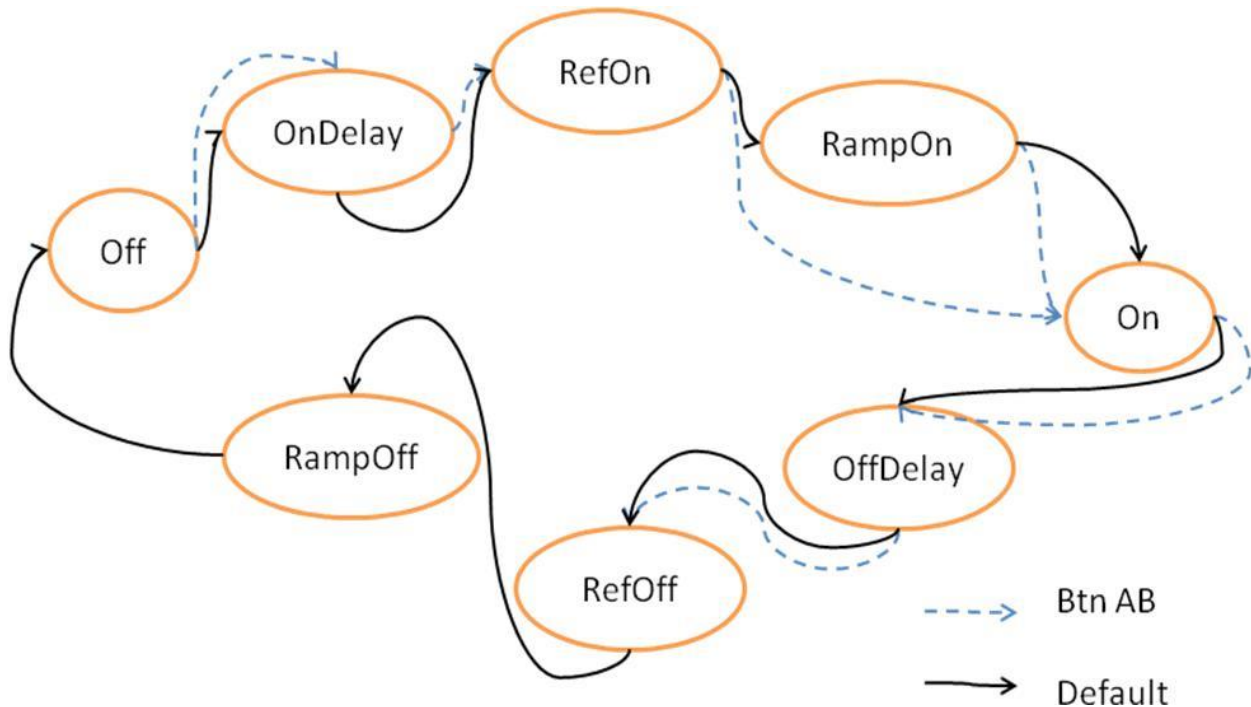
Im Zustand 'ON' bei einem Trigger "Long" (rote Pfeilrichtung) gibt es einen Übergang nach 'OffDelay'->'RefOff'->'RampOff. MAX_TIME_FIRST_DIR steht auf 0,5s. Sollte kein weiterer Trigger kommen wird in den Zustand 'Off' übergegangen. Bleibt die Taste gedrückt wird regelmäßig ein Trigger gesendet der Zustand RampOff erneut angesprungen. Der Timer MAX_TIME_FIRST_DIR startet erneut.

MultiExecution muss also auf ,on' stehen, sonst wird trotz langem Drücken nur ein Trigger pro ,Long-Press' ausgegeben.

Nach peerChan für 2 Taster sieht die Zustandsmaschine wie folgt aus:



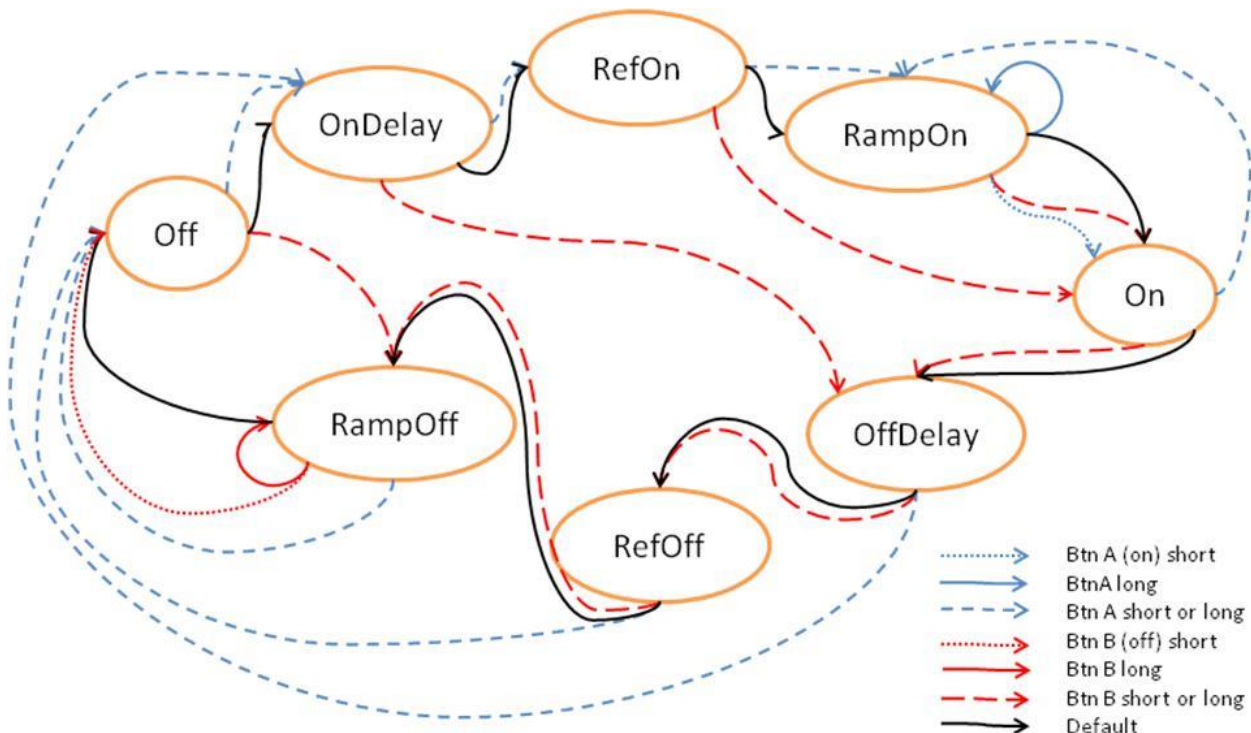
Die Zustandsmaschine bei nur **einem Taster (AB mode)** ist per Default



Die CT Steuerung ist im Default Setting nicht verwendet. Man kann hiermit Übergänge auf Grund vom Zustandswert der Steuerung veranlassen. Ich habe es nicht getestet, daher ist mir nicht klar, wie es im Detail funktioniert. Es scheint bei erfüllter Bedingung den Zustand in ‚Defaultrichtung‘ zu verlassen.

Meine Zustandsmaschine für Jalousien, 2 Tasten Betrieb:

Funktionale Änderung: Nach kurzem Tastendruck fährt die Jalousie (hoch oder runter). Stoppen erfolgt jetzt durch kurzen Druck auf Button hoch **oder** runter. Dies hat speziell bei der Schrägstellung der Jalousie Vorteile, da man einfach schneller ist und die Schrägstellung genauer einstellen kann.



Programmierung **meiner** Zustandsmaschine nach pairing mit 2 Tasten:

Es werden je Pairkanal 6 Zustandsübergänge überschrieben, da ein Byte 2 Übergänge behandelt. Die Grafik dazu ist oben die Register und Werte sind der Register Tabelle im Anhang zu entnehmen.

Hier wie Button A umprogrammiert wird. Button B entsprechend aus dem Bild „abschreiben“

set Rollo1 regSet SHBlern	rampOn BtnA
set Rollo1 regSet shBlJtOff	dlyOn BtnA
set Rollo1 regSet shBlJtDlyOn	refOn BtnA
set Rollo1 regSet shBlJtDlyOff	off BtnA
set Rollo1 regSet shBlJtRampOn	on BtnA
set Rollo1 regSet shBlJtRampOff	off BtnA
set Rollo1 regSet shBlJtRefOn	rampOn BtnA
set Rollo1 regSet shBlJtRefOff	off BtnA
set Rollo1 regSet lgBlJtOn	rampOn BtnA
set Rollo1 regSet lgBlJtOff	dlyOn BtnA
set Rollo1 regSet lgBlJtDlyOn	refOn BtnA
set Rollo1 regSet lgBlJtDlyOff	off BtnA
set Rollo1 regSet lgBlJtRampOn	rampOn BtnA
set Rollo1 regSet lgBlJtRampOff	off BtnA
set Rollo1 regSet lgBlJtRefOn	rampOn BtnA
set Rollo1 regSet lgBlJtRefOff	off BtnA

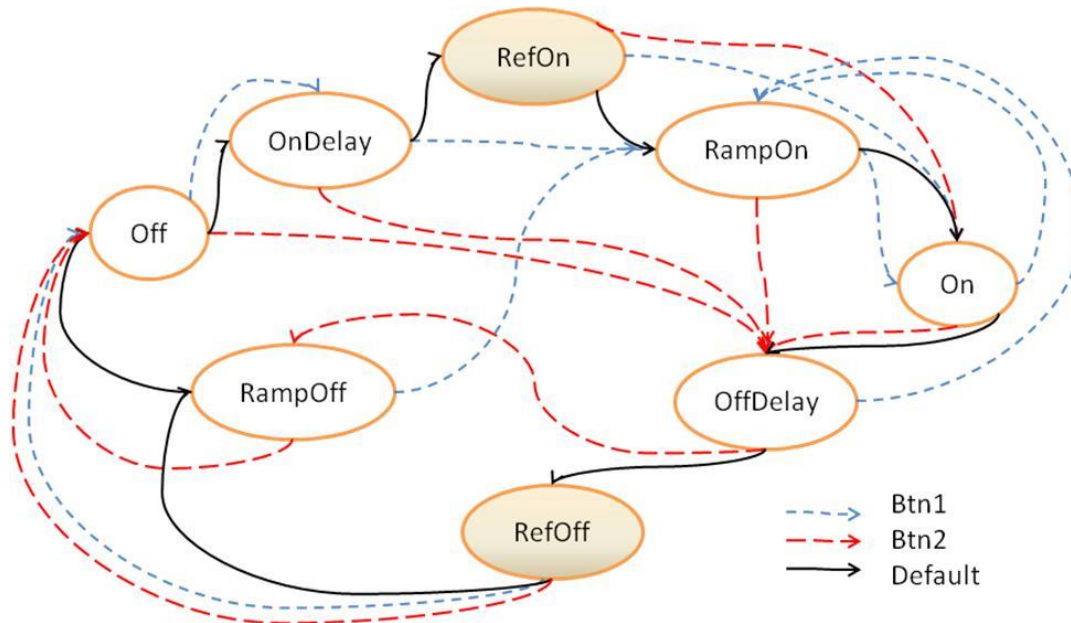
Dimmer

Verwendet wird ein HM-LC-Dim1-PBU-FM.

Generell erfolgt die Programmierung wie bei der Jalousie.

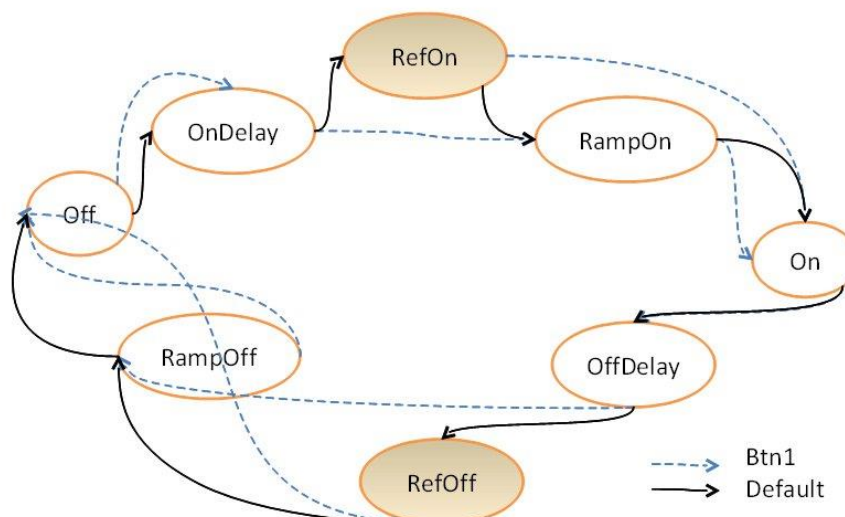
Per default benutzen Long Button nicht die Zustandsmaschine sondern DimUp/Down. Zu erkennen ist dies, da „action_Type“ für Long nicht auf JumpToTarget sondern auf einem dim steht. Siehe Registertabelle. Button short benutzt die Zustandsmaschine.

Danach ist die Zustandsmaschine **für 2 Taster** wie folgt programmiert:



Danach ist die Zustandsmaschine **für einen Taster** wie folgt programmiert:

Trigger Events Dimmer, AB mode



Die Diagramme zeigen die default Übergänge der Dimmersteuerung. Eingänge der States sind oben/links, Ausgängen unten und rechts.

States

- On/Off: Die aktuelle Helligkeit des Dimmer wird eingefroren, es erfolgt keine Änderung. On/Off spiegelt **nicht** die Stellung des Dimmer wieder (ganz an oder aus) sondern nur einen logischen Zustand der Steuerung!
- RampOn/OFF: Die Helligkeit wird langsam erhöht oder reduziert.
- RefOn/Off: Dieser Zustand ist für Dimmer nicht relevant. Es hat den Anschein, dass die Zustände existieren aber im default nicht angesprungen werden.
- On/OffDelay: Ein/Ausschaltverzögerung der Dimmersteuerung. Nach dem Trigger wird 'Delay' gewartet bis die in den nächsten Zustand übergegangen wird.

State Übergänge

- Nach abarbeiten eines Zustandes wird automatisch der nächste Zustand (gemäß schwarzem Pfeil) eingenommen
- Sollte ein Trigger kommen wird nach JT Tabelle der entsprechende Zustand angesprungen. Jede angelernte Taste hat ihre eigene JT Tabelle.
Werden 2 Tasten gleichzeitig angelernt so werden die jeweiligen JT Tabelle unterschiedlich ‚befüllt‘ um eine Taste für ‚Heller‘ und eine für ‚Dunkler‘ zu erhalten, ohne weitere Programmierung.
Per Default ist kein Unterschied zwischen ‚long‘ und ‚short‘ in den Sprungtabellen.

Action Type:

ActionType kann mit regSet gesetzt werden. Die Statemachine wird verwendet, wenn [lg|sh]ActionType = jmpToTarget gesetzt ist. Im Default ist dies nur für short gesetzt, long benutzt die verschiedene DIM actiontypes.

Virtuelle Dimmer-Kanäle:

Einige Dimmer verfügen über virtuelle Kanäle. Hierbei sind jedem physikalischen Kanal zwei zusätzliche virtuelle Kanäle zugeordnet. Diese Kanäle werden bei m pairen automatisch in FHEM eingerichtet.

Diese virtuellen Dimmer Kanäle darf man nicht mit den FHEM virtual-channels verwechseln. Während die FHEM Kanäle nur in FHEM existieren sind die Dimmer- Kanäle in den HM Devices vorhanden.

Zum einfachen Betrieb braucht man diese Kanäle nicht. Sie sind für komplexere Funktionen gedacht. Den meisten Usern kann ich raten, diese Kanäle zu ignorieren.

Prinzipiell funktioniert jeder dieser Kanäle wie auch der physikalische Kanal. Man kann ihn also peeren, ansteuern, Register setzen. Als Ergebnisse erzeugt der Physikalische sowie die beiden Virtuellen einen Stellwert für den Dimmer.

Insgesamt haben wir nun also **3 Stellwerte** für einen Dimmer. Der physikalische Stellwert wird als Kombination aus den 3 Werten errechnet.

Hierzu ist das Register „**logicCombination**“ zu setzen. Beim ersten Kanal steht es auf ‚and‘, bei den beiden folgenden auf ‚inactive‘. Somit werden die beiden Virtuellen ignoriert.

Möglichkeiten der Kombination hat man viele, siehe Registeroptionen (get regList, siehe oben). Beispielsweise kann man die Werte addieren, subtrahieren oder auch multiplizieren.

Anwendungsfälle nach HM:

Flurlampe mit Bewegungsmelder gekoppelt, die Helligkeit kann abhängig von der gemeldeten Helligkeit des Bewegungsmelders eingestellt werden – also am Tag heller, in der Nacht dunkler.

Eine Zimmerbeleuchtung soll gedrosselt werden wenn die Leselampe eingeschaltet wird.

Rauchmelder – SD Sensor

Der einzelne Rauchmelder hat eigentlich keine Einstellungen. Zu beachten ist nur die Gruppierung. HM Rauchmelder arbeiten in Gruppen – so genannten TEAMS. Diese können anhand der peerlisten erstellt und überwacht werden.

Meldungen und Alarme des Teams werden über die HMID eines der Rauchmelder signalisiert. In FHEM wird der Alarm sowohl an die teamID als auch am EinzelDevice signalisiert.

Auslesen der Peerlisten

```
set <HMdev> getConfig
```

Addieren eines Rauchmelders zum Team wird mit peerChan erreicht:

```
set <teammaster> peerChan 0 <neuerSD> single set actor
```

Entfernen aus der Gruppe mit:

```
set <teammaster> peerChan 0 <neuerSD> single unset actor
```

Teammaster:

Als Teammaster wird von HM immer einer der Melder ausgewählt. Der Grund liegt auf der Hand, HM will vermeiden eine doppelte im System zu benutzen.

Übersichtlicher wird das Ganze wenn man eine separate TeamID nutzt. Damit kann man team-events von Device-events trennen.

Eine TeamID kann man mit Hilfe von virtuellen Devices generieren :

```
define <TeamDev> CUL_HM <myUniqueHMID> #HMID muss einzig im System sein
set <TeamDev> virtual 1 # nur ein channel notwendig
```

Als Teammaster wird nun der channel 1 benötigt, also <TeamDev>_BTN1. Mit rename kann man dies etwas schöner gestalten. Beispiel:

```
define TeamDev CUL_HM 100010
set TeamDev virtual 1
rename TeamDev_Btn1 Sdteam
set Sdteam peerChan 0 SD_1 single set actor
set Sdteam peerChan 0 SD_2 single set actor
set Sdteam peerChan 0 SD_n single set actor
```

Alarmer werden am Team gemeldet. Der Auslöser wird mitgeteilt und auch beim Auslöser wird der Status entsprechend gesetzt. Ggf. machen SDs ein forward des Alarms, sie wiederholen also die Meldung. Auch dies wird angezeigt.

Raumklima Heizungssteuerung über – TC, VD und RHS Sensoren

Die prinzipielle Funktion dieser Kombination sollte allen klar sein, die sich so etwas zugelegt haben. In Kürze erklärt ist der TC eine Art Zentrale. Er misst die Raumtemperatur und reguliert ein oder mehrere Heizungsventile. Zusätzlich kann man den TC mit einem Fensterkontakt koppeln um einen anderen Temperaturmode zu fahren (also Heizung aus, wenn Fenster offen).

Das komplexe Teil dieser Kombination ist der TC. Er teilt sich intern in 3 Funktionseinheiten also 3 Kanäle.

- Channel 1 ist der „Weather“ Kanal. Er ist für die Temperaturmessung zuständig. Einstellungen kann man hier praktisch keine vornehmen.
- Channel 2 nennt sich „Climate“ und hat die Aufgabe, die Ventile einzustellen. Hier muss man die meisten Einstellungen vornehmen – Temperaturprofile für die Woche, Operationsmodi,...
- Channel 3 – „WindowRec“ stellt die Kommunikation zu den RHS, den Fenster-Sensoren sicher.

Zusammenstellen der Konfiguration – das pairing

Erst einmal will man die VDs mit den TCs kombinieren mit:

```
set <TC_Climate> peerChan 0 <VDname> single set
```

und ausgelesen wie immer mit:

```
set <TC> getConfig # es werden gleich alle Daten ausgelesen, auch die des  
                  "WindowRc" wenn TC angewendet wird.
```

Analog geschieht das Paaren mit einem RHS. Zu beachten ist, dass der RHS der Sender ist und der TC der Akteur. Demnach sind die Rollen gegenüber den VD vertauscht:

```
set <RHS> peerChan 0 <TC_WindowRec> single set
```

Auslesen der Ergebnisse siehe oben. Die Peerings sind in den entsprechenden Kanälen zu sehen – also die VD in „Climate“ und RHS in „WindowRec“.

Wie immer können mehrere peerings auf einen TC vorgenommen werden. Ein RHS kann auch an mehrere TCs senden.

Ein VD kann erhält seine Einstellungen **immer von einem TC**, hier ist ein mehrfach peering ausgeschlossen. Wenn ein VD bereits gepeert ist muss man dies rückgängig machen, also ein **unset**.

Anmerkung: Fortgeschrittenen und Spieler können auch pairings mit anderen Geräten vornehmen. Beim Pairing wird nicht darauf geachtet, ob der Empfänger ein VD ist oder der Sender ein RHS. Wer will kann also einen physikalischen (RC12,...) oder eine virtuellen Schalter mit seinem TC WindowRec pairen.

Anmerkung: Die Kommunikation mit TC und VD ist zwecks Batteriesparens in einem sog. 'wakeuptime' ausgelegt. Praktisch heißt dies, jegliche Kommunikation erst durchgeführt werden kann, wenn das Geräte seine Statusmeldung ausgibt.

Mit anderen Worten: Es kann bei einem TC bis zu 4min dauern bis ein Kommando ausgeführt wurde. Also nicht nervös werden, warten.

Zu beachten ist, dass bei einem TC auch burst eingeschaltet werden kann. Siehe hierzu conditionalBurst Seite 680

Register und Einstellungen VD

Ein VD hat nur wenigen Einstellungen

- Offset: eine Möglichkeit eines Abgleichs wenn mehrere Heizkörper von einem TC gesteuert werden und man verschiedene Charakteristiken berücksichtigen will
- ErrorPosition: Die Position, die eingenommen wird sollte der Kontakt zum TC abreißen.

Lesen und Setzen kann man die Werte über die üblichen Funktionen

„set getConfig“, „get reglist“ und „get reg all“

Beim peeren mit einem TC ist folgendes zu beachten:

Der VD wird mit dem Climate-channel des TC gepeert. Ein VD kann nur mit einem TC gepeert werden (ist logisch, es kann nur einer die Stellwerte übermitteln). Wenn ein VD mit einem TC gepeert ist muss man dies peering erst löschen um ein Neues eintragen zu können.

Zum Löschen muss die Option „unset“ in peerChan gewählt werden.

Register und Einstellungen RHS

Spezielle Einstellungen können bei einem RHS vorgenommen werden, sind aber zum jetzigen Zeitpunkt nicht ausdekodiert. Register können im raw Format bearbeitet werden.

Der Funktionsumfang eines RHS beschränkt sich auf die Konfiguration

- Welches Signal bei welcher Schalterstellung
- Meldungsverzögerung
- Schaltzeiten der LED

Register und Einstellungen TC

Die Registereinstellungen des TC sind sehr vielfältig, konzentrieren sich aber auf den Climate Kanal. Im Weather Kanal sind keine Einstellungen vorhanden.

Im WindowRec Kanal kann man die gewünschte Temperatur bei geöffnetem Fenster festlegen.

Die Einstellungen des Climate Kanals sind, wie angesprochen sehr vielfältig und hier noch nicht zusammengestellt.

Register-Definitionen

Register Definition des Schalt Aktor – HM-LC-SW2

Switch											
List 1											
									addr	Value	
								???	8	0	
							</				

Register Definition des Blind Aktor – HM-LC-Blx

Blind Actor								List 1		
									Addr	Size
x/10[0.1..6000]	50							REFERENCE_RUNNING_TIME_TOP_BOTTOM	11	2
x/10[0.1..6000]	50							REFERENCE_RUNNING_TIME_BOTTOM_TOP	13	2
x/10[0.1..25.5]	0.5							CHANGE_OVER_DELAY	15	1
								REFERENCE_RUN_COUNTER	16	1

JT-Target		
no Jump	0	
OnDelay	1	
RefOn	2	
On	3	
OffDelay	4	
RefOff	5	
Off	6	
--	7	
RampOn	8	
RampOff	9	

CT - condition			Short
0	GE CondValueLo		> Lo
1	GE CondValueHi		> Hi
2	LT CondValueLo		< Lo
3	LT CondValueHi		< Hi
4	CondValueLo=< x <CondValueHi		In
5	< CondValueLo or > CondValueHi		Out

Blind Actor												
List3 values												
		Single Button AB		Deal Button A B					Addr		Size	Comment
Definition	Value	Long	Short	Long	Short	Long	Short	Name	Short	Long		
		0	0	0	0	0	0	CT_RAMPOFF	0x1.4	0x81.4	0.4	
		0	0	0	0	0	0	CT_RAMPON	0x1.0	0x81.0	0.4	
		0	0	0	0	0	0	CT_OFFDELAY	0x2.4	0x82.4	0.4	
		0	0	0	0	0	0	CT_ONDELAY	0x2.0	0x82.0	0.4	
		0	0	0	0	0	0	CT_OFF	0x3.4	0x83.4	0.4	
		0	0	0	0	0	0	CT_ON	0x3.0	0x83.0	0.4	
0-255	50	32	32	32	32	32	32	COND_VALUE_LO	0x04	0x84	1	
0-255	100	64	64	64	64	64	64	COND_VALUE_HI	0x05	0x85	1	
0-111600.0 \$		00	00	00	00	00	00	ONDELAY_TIME	0x06	0x86	1	
0-108000.0 \$ / FF = Not Used		FF	FF	FF	FF	FF	FF	ON_TIME	0x07	0x87	1	
0-111600.0 \$		00	00	00	00	00	00	OFFDELAY_TIME	0x08	0x88	1	
0-108000.0 \$ / FF = Not Used		FF	FF	FF	FF	FF	FF	OFF_TIME	0x09	0x89	1	
0=Absolute / 1=minimal		0	0	0	0	0	0	ON_TIME_MODE	0xA.7	0x8A.7	0.1	
0=Absolute / 1=minimal		0	0	0	0	0	0	OFF_TIME_MODE	0xA.6	0x8A.6	0.1	
		1	-	1	-	1	-	MULTIEXECUTE		0x8A.5		Only valid for "Long".
0=Inactive 1=JumpToTarget 2=ToggleToCounter 3=ToggleInvToCounter				1	1	1	1	ACTION_TYPE	0xA.0	0x8A.0	0.2	
		OnDelay	OnDelay	OnDelay	OnDelay	OffDelay	OffDelay	JT_OFF	0xB.4	0x8B.4	0.4	
		OffDelay	OffDelay	OnDelay	OnDelay	OffDelay	OffDelay	JT_ON	0xB.0	0x8B.0	0.4	
		RefOff	OffDelay	OnDelay	OnDelay	RefOff	RefOff	JT_OFFDELAY	0xC.4	0x8C.4	0.4	
		RefOn	RefOn	RefOn	RefOn	OffDelay	OffDelay	JT_ONDELAY	0xC.0	0x8C.0	0.4	
				Off	Off	RampOff	RampOff	JT_RAMPOFF	0xD.4	0x8D.4	0.4	
		On	On	RampOn	RampOn	On	On	JT_RAMPON	0xD.0	0x8D.0	0.4	Statemachine aktiv, wenn ACTION_TYPE =1 (Jump_To_Target)
x/2 [%] range:[0-100%]	0%	00	00	00	00	00	00	OFF_LEVEL	0xF	0x8F	1	
x/2 [%] range:[0-100%]	100%	C8	C8	C8	C8	C8	C8	ON_LEVEL	0x11	0x91	1	
		00	00	00	00	00	00		0x12	0x92		
		00	00	00	00	00	00	RAMPON_TIME (unused)	0x13	0x93		
		00	00	00	00	00	00	RAMPOFF_TIME (unused)	0x14	0x94		
		00	00	00	00	00	00		0x15	0x95		
		00	00	00	00	00	00		0x16	0x96		
		00	00	00	00	00	00		0x17	0x97		
		00	00	00	00	00	00		0x18	0x98		
		00	00	00	00	00	00		0x19	0x99		
		00	00	00	00	00	00		0x1A	0x9A		
		00	00	00	00	00	00		0x1B	0x9B		
		0	0	0	0	0	0	CT_REFOFF	0x1C.4	0x9C.4	0.4	
		0	0	0	0	0	0	CT_REFON	0x1C.0	0x9C.0	0.4	
x/10 [s] range:[0-25.4] ff=not used	0.5s			5	FF	5	FF	MAX_TIME_FIRST_DIR	0x1D	0x9D	1	Long wird alle .5s wiederholt. Pro Trigger wird MAX_TIME_FIRST_DIR gefahren. FF: Fahrzeit unbegrenzt, Fahrt bis zum Ende (Siehe List1 Parameter)
		On	On	Off	Off	RampOff	RampOff	JT_REFOFF	0x1E.4	0x9E.4	0x0.4	
				RampOn	RampOn	On	On	JT_REFON	0x1E.0	0x9E.0	0x0.4	
DRIVE_DIRECTLY DRIVE_VIA_UPPER_END_POSITION DRIVE_VIA_LOWER_END_POSITION DRIVE_VIA_NEXT_END_POSITION		0	0	0	0	0	0	DRIVING_MODE	0x1F	0x9F	1.0	

Register Definition des Dimmer Aktor – HM-LC-Dim1T

Dimmer

List 0

						Value		Addr	Size
						0	InternalKeysVisible	2.7	0.1
[1-254]	min	5					ConfButtonTime	21	1

List 1

						Value		Addr	Size
[0-10]						0x06	TRANSMIT_TRY_MAX	48	1
[30-100]	C	80				0x50	OVERTEMP_LEVEL	50	1
[0-2.55]	s	1				0x64	FUSE_DELAY	51	1
[30-100]	C	75				0x4B	REDUCE_TEMP_LEVEL	52	1
%*200 [0-100%]	%	40%				0x50	REDUCE_LEVEL	53	1
PowerUp-on or off	bool					0x00	powerUp Action	86	0.1
%/2 [0.5-15.5]	s	2				0x4	StatusinfoMindelay	87.0	0.5
[0.0-7.0]	s	1				0x1	StatusinfoRandom	87.5	0.3
						0x01	??	89	

JT-Target

no Jump	0
OnDelay	1
RampOn	2
On	3
OffDelay	4
RampOff	5
Off	6
--	7
--	8
--	9

CT - condition

		Short
0	GE CondValueLo	> Lo
1	GE CondValueHi	> Hi
2	LT CondValueLo	< Lo
3	LT CondValueHi	< Hi
4	CondValueLo=< % <CondValueHi	In
5	< CondValueLo or > CondValueHi	Out

Dimmer											
List 3											
Definition	Value	Single Button		Dual Button				Name	Short	Long	
		AB		A		B					
		Short	Long	Short	Long	Short	Long				
		0	0	0	0	0	0	CT_RAMPOFF	0x14	0x814	0.4
		0	0	0	0	0	0	CT_RAMPON	0x10	0x810	0.4
		0	0	0	0	0	0	CT_OFFDELAY	0x24	0x824	0.4
		0	0	0	0	0	0	CT_ONDELAY	0x20	0x820	0.4
		0	0	0	0	0	0	CT_OFF	0x34	0x834	0.4
		0	0	0	0	0	0	CT_ON	0x30	0x830	0.4
0-255	50	0x32	0x32	0x32	0x32	0x32	0x32	COND_VALUE_LO	0x04	0x84	1
0-255	100	0x64	0x64	0x64	0x64	0x64	0x64	COND_VALUE_HI	0x05	0x85	1
0-111600.0 S		0	0	0	0	0	0	ONDELAY_TIME	0x06	0x86	1
0-108000.0 S / FF = Not Used		0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	ON_TIME	0x07	0x87	1
0-111600.0 S		0	0	0	0	0	0	OFFDELAY_TIME	0x08	0x88	1
0-108000.0 S / FF = Not Used		0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	OFF_TIME	0x09	0x89	1
0= Absolute / 1= minimal		0	0	0	0	0	1	ON_TIME_MODE	0xA.7	0x8A.7	0.1
0= Absolute / 1= minimal		0	0	0	0	0	0	OFF_TIME_MODE	0xA.6	0x8A.6	0.1
Bool 0= INACTIVE 1= JUMP_TO_TARGET 2= TOGGLE_TO_COUNTER 3= TOGGLE_INVERS_TO_COUNTER 4= UPDIM 5= DOWNDIM 6= TOGGLEDIM 7= TOGGLEDIM_TO_COUNTER 8= TOGGLEDIM_INVERS_TO_COUNTER		0	1		1		1	MULTIEXECUTE		0x8A.5	0.1
		1	6	1	4	1	5	ACTION_TYPE	0xA.0	0x8A.0	0.4
		OnDelay		OnDelay	OnDelay	OffDelay	OffDelay	JT_OFF	0xB.4	0x8B.4	0.4
		OffDelay		RampOn	RampOn	OffDelay	OffDelay	JT_ON	0xB.0	0x8B.0	0.4
		RampOff		RampOn	RampOn	RampOff	RampOff	JT_OFFDELAY	0xC.4	0x8C.4	0.4
		RampOn		RampOn	RampOn	OffDelay	OffDelay	JT_ONDELAY	0xC.0	0x8C.0	0.4
		Off		RampOn	RampOn	Off	Off	JT_RAMPOFF	0xD.4	0x8D.4	0.4
		On		On	On	OffDelay	OffDelay	JT_RAMPON	0xD.0	0x8D.0	0.4
Bool	0			0	0	0	0	ONDELAY_MODE	0xE.7	0x8E.7	0.1
Bool	0			0	0	0	0	ON_LEVEL_PRI0	0xE.6	0x8E.6	0.1
Bool	1			1	1	1	1	OFFDELAY_BLINK	0xE.5	0x8E.5	0.1
%2 [%] Range [0-100%]	0% 0%			0	0	0	0	OFF_LEVEL	0xF	0x8F	1
%2 [%] Range [0-100%]	10% 0x14	0x14		0x14	0x14	0x14	0x14	ON_MIN_LEVEL	0x10	0x90	1
%2 [%] Range [0-100%]	100% 0xC8	0xC8		0xC8	0xC8	0xC8	0xC8	ON_LEVEL	0x11	0x91	1
%2 [%] Range [0-100%]	5% 0x0A	0x0A		0x0A	0x0A	0x0A	0x0A	RAMP_START_STEP	0x12	0x92	1
0-111600.0 S	5	5		5	5	5	5	RAMPON_TIME	0x13	0x93	1
0-111600.0 S	5	5		5	5	5	5	RAMPOFF_TIME	0x14	0x94	1
%2 [%] Range [0-100%]	0% 0	0		0	0	0	0	DIM_MIN_LEVEL	0x15	0x95	1
%2 [%] Range [0-100%]	100% 0xC8	0xC8		0xC8	0xC8	0xC8	0xC8	DIM_MAX_LEVEL	0x16	0x96	1
%2 [%] Range [0-100%]	5% 0x0A	0x0A		0x0A	0x0A	0x0A	0x0A	DIM_STEP	0x17	0x97	1
%2 [%] Range [0-100%]	5% 0x0A	0x0A		0x0A	0x0A	0x0A	0x0A	OFFDELAY_STEP	0x18	0x98	1
(x+1)/10 [s] Range [0.1-256s]	0.5s 4	4		4	4	4	4	OFFDELAY_NEWTIME	0x19	0x99	1
(x+1)/10 [s] Range [0.1-256s]	0.5s 4	4		4	4	4	4	OFFDELAY_OLDTIME	0x1A	0x9A	1
	0	0x20		0	0	0	0		0x1B	0x9B	
		1	1	1	1	1	1	CT_REFOFF(unused)	0x1C.4	0x9C.4	0.4
		4	4	4	4	4	4	CT_REFON(unused)	0x1C.0	0x9C.0	0.4
		0x52	0x52	0x52	0x52	0x52	0x52	????	0x1D	0x9D	1
		Off	Off	Off	Off	Off	Off	JT_REFOFF(unused)	0x1E.4	0x9E.4	0x0.4
		On	On	On	On	On	On	JT_REFON(unused)	0x1E.0	0x9E.0	0x0.4

Statemachine aktiv, wenn ACTION_TYPE = 1 (Jump_To_Target)

Statemachine aktiv, wenn ACTION_TYPE = 1 (Jump_To_Target)
Aus BlindActor ableitet. Evtl wird die selbe Statemachine verwendet

Register Definition der Fernbedienung RC12

RC12				
Definition		Index	Size	
List 0				
[s] range [0 - 255]	BACKLIGHT_ON_TIME	14	1	
bool	BACKLIGHT_AT_KEYSTROKE	13.7	0.1	
bool	BACKLIGHT_AT_MOTION	13.6	0.1	
List 1				
[100ms] range [0.3 - 1.8]	LONG_PRESS_TIME	4.4	0.4	
[100ms] range [0.0 - 1.5]	DBL_PRESS_TIME	9.0	0.4	
List 4 (one per Channel)				
bool	PEER_NEEDS_BURST	1	0.1	
bool	EXPECT_AES	1.7	0.1	

Register Definition der Fernbedienung RC19

RC19				
Definition		Index	Size	
List 0				
	LANGUAGE	7	7	
[s] range [0 - 255]	BACKLIGHT_ON_TIME	14	1	
bool	BACKLIGHT_AT_KEYSTROKE	13.7	0.1	
bool	BACKLIGHT_AT_MOTION	13.6	0.1	
bool	BACKLIGHT_AT_CHARGE	13.5	0.1	
List 1				
[100ms] range [0.3 - 1.8]	LONG_PRESS_TIME	4.4	0.4	
[100ms] range [0.0 - 1.5]	DBL_PRESS_TIME	9.0	0.4	
[0.5s] range [1 - 120]	MESSAGE_SHOW_TIME	45	1	
	BEEP_AT_ALARM	46.0	0.2	
	BEEP_AT_ALARM	46.2	0.2	
none/Tone1..3 [0..3]	BEEP_AT_STATUSINFO	46.4	0.2	
	BACKLIGHT_AT_ALARM	47.0	0.2	
off/on/blinkSlow/blinkFast [0..3]	BACKLIGHT_AT_SERVICE	47.2	0.2	
	BACKLIGHT_AT_STATUS	47.4	0.2	
List 4 (one per Channel)				
bool	PEER_NEEDS_BURST	1	0.1	
bool	EXPECT_AES	1.7	0.1	
NONE,BULB,SWITCH,WINDOW,DOOR,BLIND,SCENE,PHONE,BELL [0..255]	LCD_SYMBOL	2	1	
none,Light,Blind,Marquee,Door,Window [0..255]	LCD_LEVEL_INTERP	3	1	

Hints, FAQs, Ideas

Eine Sammlung aus Ideen, Fragen(und Antworten):

Schalter entprellen

Verhindert das mehrfache Auslösen eines Triggers in einem kurzen Zeitabstand. Im Beispiel wird ein weiterer Trigger von vb_Btn4 ignoriert, wenn der innerhalb von 10 sec noch einmal kommt. Kam zum Einsatz um eine Türklingel zu Entprellen und hektische Gäste zu beruhigen.

```
define d dummy
define n1 notify vb_Btn4:virtActStat.* \
  {if (ReadingsVal("d","state","") ne "on")\
    {fhem("set d on");; \
     fhem("define a1 at +00:00:10 set d off")}}
```

Peer im Aktor löschen wenn der Sensor nicht mehr existiert

Das Kommando peerChan braucht immer einen Sensor-channel als Basis. Das wirft ein Problem auf wenn der Sensor aus welchen Gründen auch nicht mehr existiert. Abhilfe ist, sich temporär eine Sensor zu schaffen. Annahme: der Aktor Kanal „act_ch2“ hat noch einen Peer 123456, channel 4 eingetragen, also 12345604.

```
define vrt CUL_HM 123456
set vrt virtual 4
set vrt_Btn4 peerChan 0 act_chn2 single unset
delete vrt
```

Peer Verhalten kopieren

Hat man mühsam eine besondere Funktion in einem Peer realisiert und möchte diese jetzt auf andere Channels übertragen ist dies möglich. Voraussetzung:

Actor Kanal a1_c2 ist gepeert mit Sensor Kanal s1_b3. Das Ergebnis ist wie gewünscht uns soll nun kopiert werden auf mehrere „peers“:

```
a1_c2->peer s1_b3 # ist der prototype
a1_c2->peer s1_b4 # gleicher Aktor, anderer Sensor
a1_c3->peer s1_b3 # gleiches Device anderer Kanal, gleicher Sensor
a3_c7->peer s1_b5 # anderes Device anderer Kanal, anderer Sensor
```

Einstellungen sichern mit

```
set a1 saveConfig myFile
```

Peeren muss man wie immer. Achtung: Prototyp nicht mehr peeren, sonst sind die Einstellungen futsch!!

```
set s1_b4 peerChan 0 a1_c2 single
set s1_b3 peerChan 0 a1_c3 single
set s1_b5 peerChan 0 a3_c7 single
```

myfile im Editor öffnen und den Eintrag suchen mit „set a1_c2 regBulk 3:s1_b3“. Die Zeile kopieren und die Namen ersetzen

```
set a1_c2 regBulk 3:s1_b4 ...
set a1_c3 regBulk 3:s1_b3 ...
set a3_c7 regBulk 3:s1_b5 ...
```

Tips SystemÜbersicht, Hminfo

Wie im vorigen Teil beschrieben sind bei HM einige Details zu beachten. Einiges geht automatisch, aber man muss doch immer im Details nachsehen, ob alles funktioniert. In FHEM werden einigen Hilfestellungen zu Verfügung, die den Umgang mit HM erleichtern/ermöglichen sollen.

Sinnvoll ist es daher sich eine Instanz von HMinfo zu erstellen:

```
define hm Hminfo
```

Danach stehen einige Methoden zu Verfügung auf die hingewiesen werden sollte:

protoEvents

```
set hm protoEvents short
```

kann man eine Tabelle der form Erhalten

```
protoEvents done:
  name      :State |CmdPend |Snd |Resnd #CmdDel |ResndFail |Nack |IOerr
  LichtDev  : done | -      |4: | -    # -    | -      | -    | -
  LichtLDev : done | -      |6: | -    # -    | -      | -    | -
  ...
  RolloFH   : done | -      |2: | -    # -    | -      | -    | -
  RolloSL   : done | -      |2: | -    # -    | -      | -    | -

  CUL_HM queue:0

  autoReadReg pending: recent:none
  status request pending:
  autoReadReg wakeup pending:FB,FB2,TastDev
  status request wakeup pending:
  IODevs:LANIf1:opened pending=0 condition:ok
  msgLoadEst: 1hour:2% 10min steps: 0/0/2/0/0/0
```

State: aktueller Zustand der Messageübertragung: processing, pending oder done
CmdPend: wenn Nachrichten pending sind ist hier aufgeführt, wieviele es sind
Snd: Anzahl der gesendeten Nachrichten
Resnd: Anzahl der Nachrichten, die von FHEM wiederholt werden mussten
#: nach dem Hash stehe Fehler. Wenn hier Werte stehen sit etwas schief gegangen
CmdDel: Anzahl **vor** der Übertragung gelöschter Komamndos
ResndFail: Trotz wiederholen nicht erfolgreiche Übertragung
Nack: Anzahl von Device zurückgewiesener Messages
IOerr: Anzahl Probleme aufgrund von Fehlern des IO device (overload,...)

Die Zähler sind fortlaufend. Man kann die Zähler und rücksetzen je device mit

```
set <device> clear msgEvents
```

oder für **alle HM devices** in einem Kommando:

```
set hm clear Protocol
```

CUL_HM queue:Anzahl der Entities, die auf Verarbeitung warten. Die Begrenzung findet statt durch ein Problem des IO Device sowie bei HMLAN das Attribut hmLanQLen.

Pending: Entities die auf eine automatische Abfrage von Status ooder getConfig warten. Dies ist zum einen last-gesteuert und hängt zum Zweiten vom Modus des devices ab (config/wakeup). FHEM verzögert die Übertragung entsprechend.

IODevs: zustand der IO devices, die HM nutzt. Für HMLAN/USB wird auch die belastung der aktuellen Stunde angezeigt um zu sehen, wie nahe man dem Overload ist

RSSI

```
set hm rssi
```

liefert die Empfangslevel der Devices auf beiden Seiten

rssi done:

Device	:receive	from	last	avg	min<max	count
LichtDev	:LANIf1	LichtDev	-65.0	-65.5	-66.0< -65.0	4
LichtDev	:LichtDev	LANIf1	-69.0	-68.5	-69.0< -68.0	2
LichtLDev	:LANIf1	LichtLDev	-83.0	-85.0	-88.0< -83.0	6
LichtLDev	:LichtLDev	LANIf1	-84.0	-86.7	-89.0< -84.0	3
RolloFH	:LANIf1	RolloFH	-62.0	-62.0	-62.0< -62.0	2
RolloFH	:RolloFH	LANIf1	-64.0	-64.0	-64.0< -64.0	1
RolloFH	:rpt_LANIf1	RolloFH	-75.0	-75.0	-75.0< -75.0	1
RolloSL	:LANIf1	RolloSL	-75.0	-75.0	-75.0< -75.0	2
RolloSL	:RolloSL	LANIf1	-74.0	-74.0	-74.0< -74.0	1

Kanngelöscht werden mit

```
set hm clear rssi
```

peerXref

```
set hm peerXref
```

liefert das peering zwischen den Devices. Die Tabelle spiegelt nur die von FHEM gelesenen Werte wieder.

peerXref done:

```
x-ref list
  LichtF =>FB_Btn_12
  LichtF =>LichtF
  LichtF =>Tast_02
  LichtF =>vb_Btn2
```

peerCheck

```
set hm peerCheck
```

prüft, ob das peering symmetrisch ist. Die Listen sollten eigentlich leer sein, wenn alles stimmt. . Die Tabelle spiegelt nur die von FHEM gelesenen Werte wieder.

peerCheck done:

```
peer list not read
peer list incomplete
peer not verified
  LichtF p:vb_Btn2
  h_FstH_WindowRec p:11000002
  th_Climate p:vb_Btn3
```

checkConfig

```
set hm checkConfig
```

prüft, ob die in FHEM vorhandenen Daten komplett sind und konsistent. Es wird nicht geprüft, ob die Daten alt sind! PeerCheck wird implizit mit ausgeführt

autoReadReg

```
set hm autoReadReg
```

löst ein Lesen der Configuration aller Devices aus, die per Attribut autoReadReg entsprechend freigegeben sind. Sinn ist ein einfacher und kompletter update der Konfigurationsdaten.

param

```
set hm param [<filter>] <parameter>
```

erstellt eine Tabelle entsprechender Parameter aller HM devices.

```
set hm param -de PairedTo # alle Devices und zu wem sie gepeert sind  
set hm param -de PairedTo model# und die models in einer Liste
```

HMinfo web Ansicht

Die web-ansicht von HMinfo versucht eine Darstellung aller wichtigen Daten und Ereignisse. So werden per default batterie-warnungen angezeigt, Protokoll-fehler, motor-error,...

Einstellbar ist dies vom User über die Attribute von HMinfo.

Das Kommando „update“ erneuert die web-ansicht. Mit „autoUpdate“ kann man einstellen, dass die web-ansicht automatisch refreshed wird.