

Heizungs-
Datenerfassung
mit
Raspberry Pi ®
und
Laptop/PC

Inhaltsverzeichnis

Kurzbeschreibung.....	3
1 Hardware.....	4
1.1 Adapter für Raspberry Pi® (HT3-Miniadapter).....	4
1.1.1 Schaltplan und Stückliste.....	5
1.1.2 Funktionsbeschreibung.....	6
1.1.3 Inbetriebnahme.....	7
1.1.4 Realisierungsbeispiel.....	8
1.2 USB-Adapter für PCs und Laptops (HT3-Microadapter).....	10
1.2.1 Schaltplan und Stückliste.....	11
1.3 Transceiver Frontend (ht-transceiver) für Raspberry Pi® / USB.....	12
1.3.1 Adapter 'ht_piduino'.....	13
1.3.1.1 'ht_piduino' Bild, Schaltplan und Stückliste.....	14
1.3.2 Adapter 'ht_pitiny'.....	16
1.3.2.1 'ht_pitiny' Bild, Schaltplan und Stückliste.....	16
1.3.3 USB-MotherBoard für Adapter 'ht_transceiver' und UM2102N.....	18
2 Software.....	19
2.1 Verzeichnis-Struktur.....	20
2.2 Konfiguration.....	21
2.3 Schnittstellen.....	30
2.3.1 Datenbanken.....	30
2.3.1.1 Datenbank 'SQLite'.....	30
2.3.1.2 Datenbank 'rrdtool'.....	32
2.3.2 'MQTT' - Client.....	33
2.3.3 'SPS' – ht_Server.....	35
2.3.4 'HomeAssistant'-IF (mqtt client-daemon).....	37
2.4 Applikationen.....	40
2.4.1 HT3-Analyser.....	40
2.4.2 HT3_Systemstatus.....	43
2.4.3 ht_collgate.....	44
2.4.4 ht_2hassio.....	46
2.5 Comport <=> Socket proxy (Server & Client).....	47
2.5.1 ht_proxy (proxy-server).....	47
2.5.2 ht_client_example (proxy-client Beispiel).....	48
2.5.3 ht_netclient (Heizungssteuer-Client).....	49
2.5.3.1 ht_netclient Steuerbefehle.....	50
3 Installation.....	54
3.1 Betriebssystem.....	54
3.2 UART Schnittstelle.....	55
3.3 Komponenten für MQTT.....	55
3.3.1 HomeAssistant (Hassio) Schnittstelle.....	57
3.4 Applikationen.....	58
3.4.1 Kurzbeschreibung der Softwaremodule.....	62
3.4.2 Mögliche Konfigurationen (HW/SW).....	65
4 HT Applikation im Betrieb.....	66
5 Weiterführende Literatur und URL's.....	68
6 Historie.....	69

Kurzbeschreibung

Diese Anleitung beschreibt die Hardware- und Software-Anteile einer Heizungs-Datenerfassung.

Die Adaption beschränkt sich z.Zeit auf die Protokolle von Heizungsanlagen mit Heatronic©- /EMS2-Bus der Firma Junkers/Bosch.

Es werden die Heizungs- und Solaranlagen Informationen grafisch dargestellt und für die aktuelle und spätere Auswertung gespeichert.

Die Erfassung und die weitere Nachverarbeitung beeinflussen nicht den Heizungs-Betrieb.

Eine Regelung der Heizung ist nicht realisiert und auch nicht vorgesehen, eine Steuerung der Heizung kann mit den 'ht_transceiver' erreicht werden.

Die Hardware ist vorrangig für das Board 'Raspberry Pi ®', kann jedoch ohne größeren Aufwand an andere Hardware adaptiert werden. Es ist nur ein UART erforderlich, der die Datenerfassung durchführt.

Als Alternative ist ein USB-Adapter beschrieben, der die Erfassung der Heizungsdaten durch jeden PC oder Laptop ermöglicht.

Es wird die Programmiersprache 'Python' verwendet. Diese realisiert u.A. einen HT3- Telegrammanalysator mit zugehöriger grafischen Anzeige.

Durch Konfigurationsänderungen lässt sich die Software zu einer reinen grafischen Statusanzeige ändern.

Zusätzlich können dekodierte Heizungsdaten an ein MQTT- oder SPS-Interface gesendet werden wenn diese Schnittstellen in der Konfiguration aktiviert sind.

Die Daten werden nach Erfassung eines gültigen Telegramms in eine SQLite - Datenbank geschrieben. Im Rhythmus von 60 Sekunden (default) werden diese Daten zusätzlich in eine weitere Datenbank (rrdtool) übertragen. Mit dem rrd-Tool wird auch die grafische Darstellung realisiert.

Details sind in den folgenden Kapiteln zu finden.

Gewährleistung, Haftung und Ansprüche durch Fehlfunktionen an Heizung oder Adaption sind hiermit ausdrücklich ausgeschlossen.

Der Nachbau und die Inbetriebnahme der Adaption ist auf eigene Gefahr und die Beschreibung und die Software erheben nicht den Anspruch auf Vollständigkeit. Eine Änderung an Software-Modulen und Hardware-Beschreibungen ist jederzeit ohne Vorankündigung möglich.

Die hier verwendeten Handels- und Gebrauchsnamen können auch ohne besondere Kennzeichnung Marken sein und somit den gesetzlichen Bestimmungen unterliegen.

Anregungen, Fragen und mehr an:
Email: junky-zs at gmx dot de

1 Hardware

Die Hardware besteht aus einem Adapter, der die Pegel-Wandlung und Anpassung der Heatronic-/EMS-Bussignale in einen seriellen Datenstrom für die UART-Erfassung durchführt. Der Adapter gewährleistet eine galvanische Trennung zwischen der Heizungsanlage und der Datenerfassung.

Die Daten des Heatronic-/EMS-Bus werden mit folgenden Parametern übertragen:

Baudrate	:	9600
Datenbits	:	8
Parity	:	keine
Stopbit	:	1
Kurzform	:	9600, 8N1

Wenn als Erfassungsfrontend der 'ht_transceiver'-Adapter verwendet wird, so ist dazu das Interface mit folgenden Parameter-Werten einzustellen:

Baudrate	:	19200
Datenbits	:	8
Parity	:	keine
Stopbit	:	1
Kurzform	:	19200, 8N1

Details zum 'ht_transceiver' siehe auch im Kapitel:

Transceiver Frontend (ht-transceiver) für Raspberry Pi® / USB.

Die Realisierung ist im folgenden für den Computer 'Raspberry Pi®' und für die Schnittstelle USB (Hardware-neutral) beschrieben.

1.1 Adapter für Raspberry Pi® (HT3-Miniadapter)

Der Adapter ist in der Größe und in der Anschlussbelegung für den Einbau in einen Raspberry Pi® ausgelegt.

Für den Bus Anschluss und für die Betriebsstatus-LED's sind Durchführungen im Gehäuse erforderlich. Die mechanische Befestigung des Adapters wird durch die zweireihige 13-polige Buchsenleiste auf dem Adapter realisiert. Der Aufbau des Adapters kann auf einer Lochraster-Platine durchgeführt werden. SMD-Bauteile sind nicht erforderlich, da genügend Platz auf dem Adapter vorhanden ist.

Die Spannungsversorgung der Adapter-CPU ist durch die Betriebsspannung des Raspberry Pi realisiert. Die Stromaufnahme beschränkt sich dabei auf wenige Milliampere und wird einzig durch den Pullup-Widerstand am UART-Rx Eingang, dem LED / Vorwiderstand am UART-TX Ausgang und dem Optokoppler bestimmt.

Die Spannungsversorgung des Adapters auf der Heatronic-Bus Seite wird durch die Heizung realisiert und ist für 15Volt und 3mA ausgelegt.

1.1.1 Schaltplan und Stückliste

Im folgenden sind der Schaltplan und die zugehörige Stückliste aufgeführt:

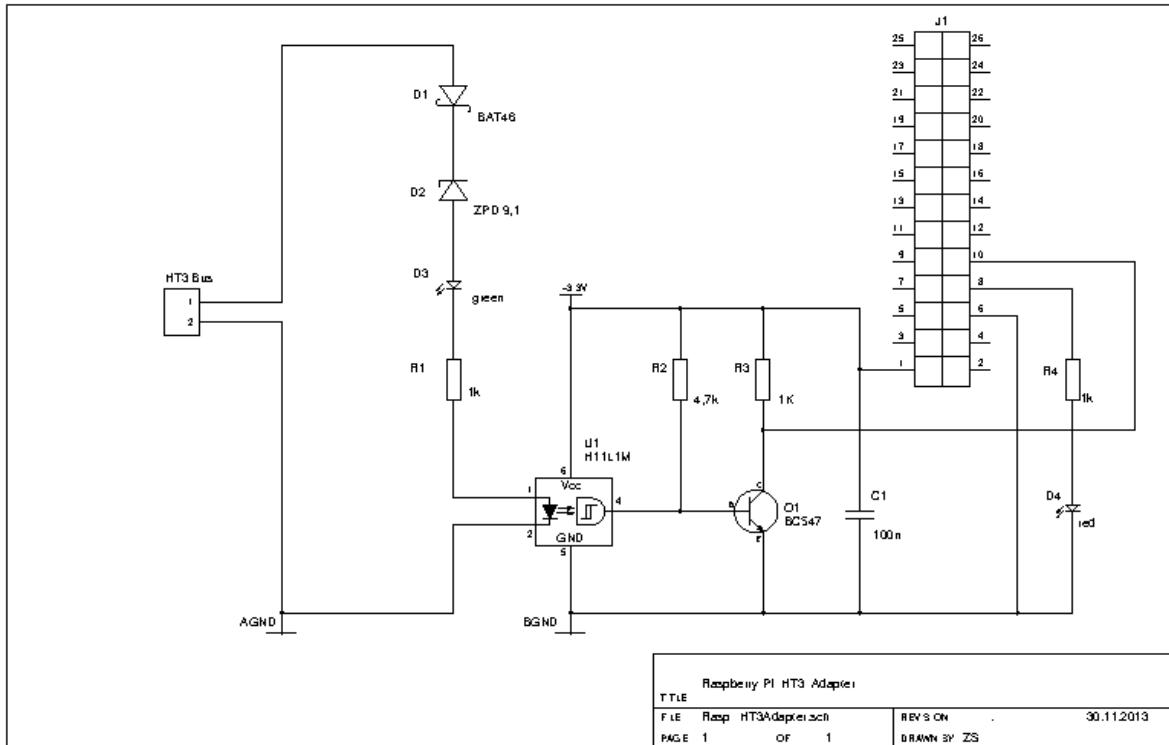


Abbildung 1: Schaltplan HT3 Mini-Adapter für Raspberry Pi

Name	Bezeichnung / Type	Anzahl	Bemerkung
U1	H11L1M Optokoppler oder PC900V	1	Versionen H11L2M und H11L3M sind nicht geeignet, da diese erst bei einem höheren LED-Strom schalten aber der Adapter möglichst geringe BUS-Belastung erzeugen soll.
D1	BAT46 oder 1N4148	1	
D2	ZPD9.1 oder BZX55/C9V1	1	Zenerdiode, Spannung 9,1 Volt
D3/4	LED (grün/rot), kleine Ausführung	2	
Q1	BC547 oder ähnlich	1	
R1/R3/R4	1 kOhm Widerstand	3	
R2	4,7 kOhm Widerstand	1	
C1	100 nF Keramik Kondensator	1	
HT3 Port	Printklemmenblock	1	Conrad, Bestell-Nr.: 731986
J1	Buchsenleiste RM2,54	2*13	Conrad, Bestell-Nr.: 733779 oder 733755
-	Lochraster-Platine	1	52*33 mm -> 20*13 Lötaugen

Tabelle 1: HT3 Adapterstückliste

1.1.2 Funktionsbeschreibung

Zur galvanische Trennung wird der Optokoppler H11L1M von Fairchild (oder PC900V) zwischen HT3-Bus und Raspberry Pi verwendet.

Dieser hat ein Schmitt-Trigger Ausgangssignal mit Hysteresis und ist schnell genug für das Signalprotokoll mit: 9600 Baud.

Der Optokoppler schaltet bei Eingangsströmen größer 1.6 mA den Ausgang auf logisch Null. Unterhalb von 1.0 mA ist das Ausgangssignal auf logisch Eins. Durch diese Hysterese werden Störungen auf dem Eingangssignal unterdrückt.

Der Transistor am Ausgang des Optokopplers invertiert das Signal, sodass der UART einer CPU (z.B. Raspberry Pi) dieses korrekt empfangen kann.

Die Diode D1 dient als Verpolschutz, auf eine Eingangs-Brückenschaltung wurde hier verzichtet.

Zenerdiode D2 passt das Signal an den Optokoppler an.

LED D3 dient als Signalindikator für den richtigen Anschluss (richtige Polarität) des HT-Bus.

Die LED D4 ist am TX-UART Ausgang des Raspberry Pi-Ports angeschlossen und wird hier nur als Betriebsindikator genutzt.

1.1.3 Inbetriebnahme

1. Eingangs-Widerstand am HT3-Anschluss (Printklemmenblock) messen.
Es darf kein Kurzschluss vorhanden und der Widerstand muss größer als 1kOhm sein.
2. Galvanische Trennung zwischen HT3-Bus und uC Anschluss messen.
Der Widerstand zwischen HT3-Bus und Ausgangsseite des Optokopplers (U1) muss sehr groß sein (>> 1 MOhm). Es darf keine Verbindung zwischen HT3-Bus und UART-RX Eingang vorhanden sein.
3. Funktionsprüfung ohne Heizungs-Anschluss mit Prüfspannung.
Eine externe Spannung von ca. 14-15 Volt an den HT3-Bus Printklemmenblock anschliessen.
Die LED D3 muss aufleuchten, wenn die Polarität der Eingangsspannung korrekt ist. Die Stromaufnahme ist dabei größer als 1.6 mA jedoch weniger als 5 mA.

Den gleicher Test mit reduzierter Eingangsspannung von weniger als 11 Volt durchführen.
Die LED D3 darf nicht oder nur wenig leuchten. Die Stromaufnahme muss weniger als 1 mA sein. Falls dies nicht passt, den Einbau der Zenerdiode D2 prüfen.
4. Anschluss an die Heizungsanlage.
Vor Anschluss des Adapters die Heizung ausschalten! Die Hinweise des Herstellers beachten. Dabei Leitungslängen und Kabelquerschnitte berücksichtigen. Die Klemmen für den HT3-Bus sind in der Regel mit 'B' bezeichnet.
Die Klemmen des neueren EMS2-Bus sind in der Regel mit 'EMS' bezeichnet.

1.1.4 Realisierungsbeispiel

In den folgenden Bildern ist die Realisierung des HT3-Adapters mit dem Raspberry Pi gezeigt:

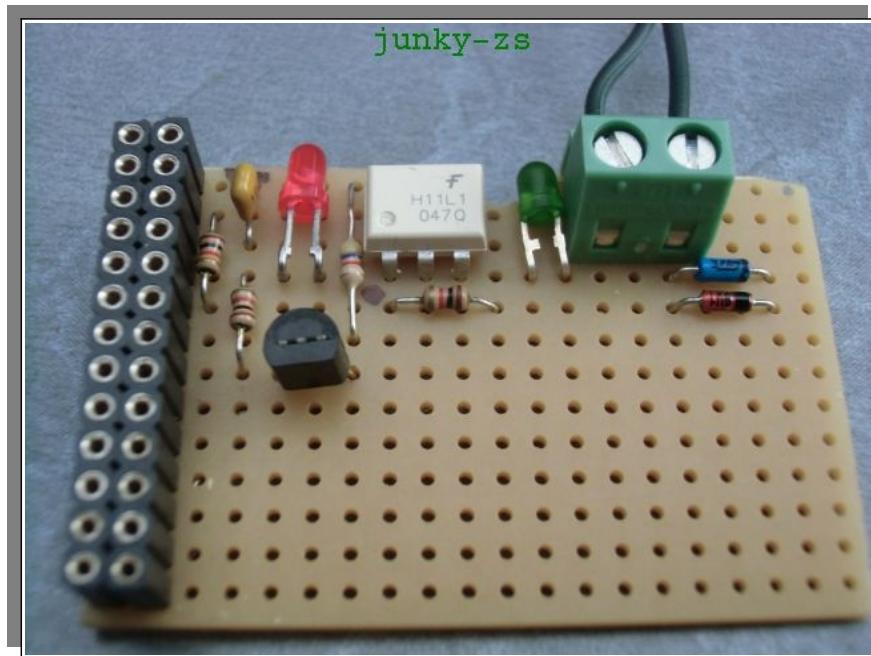


Abbildung 2: HT3 Mini-Adapter für Raspberry Pi

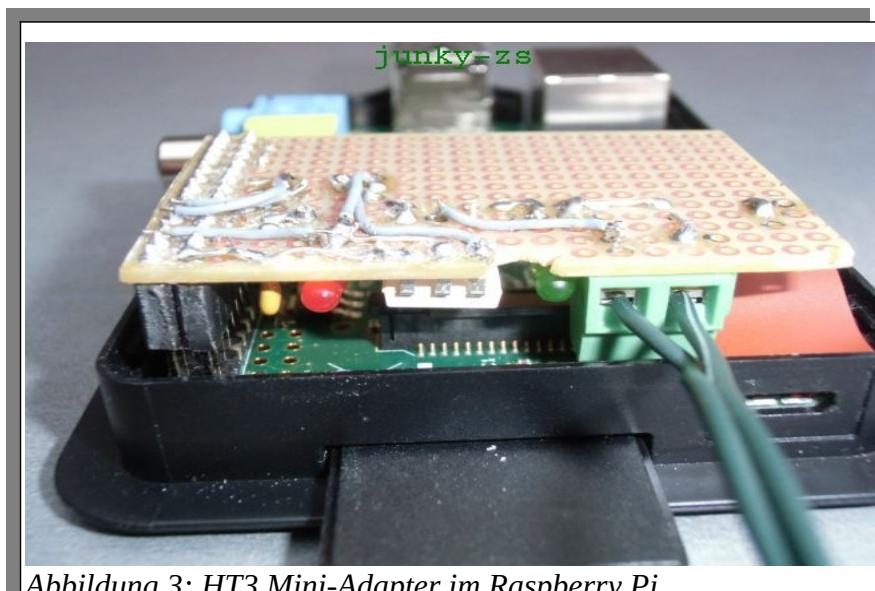


Abbildung 3: HT3 Mini-Adapter im Raspberry Pi

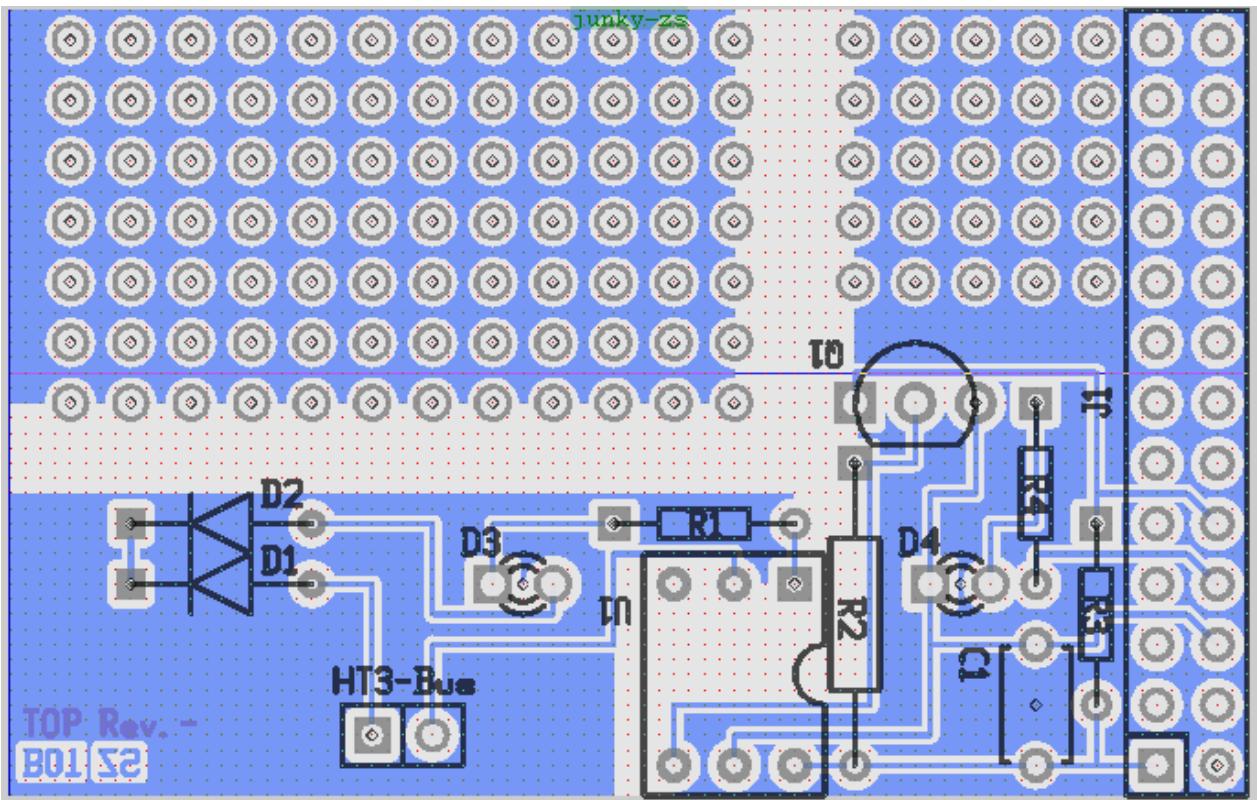


Abbildung 4: HT3 Adapter Layout

Bei der Bestückung ist zu beachten, dass die Bauteile so platziert werden wie hier gezeigt. Andernfalls passt der HT3 Adapter nicht auf den Raspberry Pi.

Auch muss der Transistor Q1 tief genug eingelötet werden, damit er keinen Kontakt mit dem darunter liegenden Spannungsregler hat. Tief genug meint hier, nicht höher als die Buchsenleisten-Oberkante.

Auch sollte der Abstand zwischen HT3-Bus und Raspberry Pi Ports möglichst groß sein um eine gewisse Spannungsfestigkeit zu gewährleisten. Daher auch die getrennten Vollflächen zwischen dem Ein- und Ausgang des Optokopplers U1. Die Vollflächen sollten keinesfalls verbunden werden, dann besser weglassen.

Die zusätzlichen Lötaugen lassen Platz für Erweiterungen.

1.2 USB-Adapter für PCs und Laptops (HT3-Microadapter)

Die Bilder zeigen wie der HT3-Microadapter aussehen kann. Die Anpassung zum HT3-Bus ist als eigenständige Platine ausgelegt. Diese wird unter den USB/RS232 Wandler gesteckt.

Als Gehäuse habe ich eine Streichholzschachtel gewählt. Ich dachte mir, das es eine zündende Idee ist es einmal so zu machen. Geräteschutzklasse: IP null

(bitte die Aufschrift 'VON KINDERN FERNHALTEN' nicht beachten, ist alles ist für den Nachbau geeignet)

Hinweis: Die rote und die orangene LED dienen nur zu Testzwecken und sind für den HT3-Datenempfang nicht erforderlich.



Abbildung 5:
HT3-Microadapter
(Hightech Gehäuse)



Abbildung 6: HT3-Microadapter
(Basisplatine)

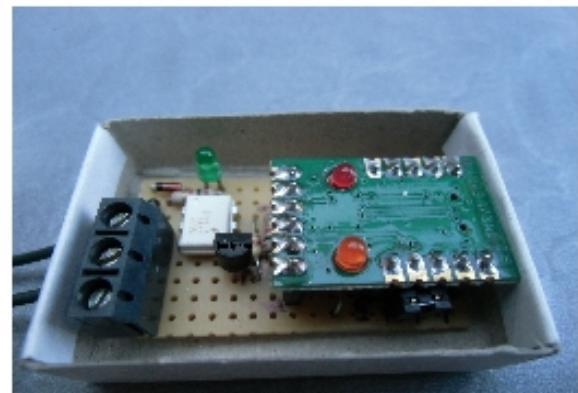


Abbildung 7: HT3 Microadapter (komplett)

1.2.1 Schaltplan und Stückliste

Im folgenden sind der Schaltplan und die zugehörige Stückliste aufgeführt:

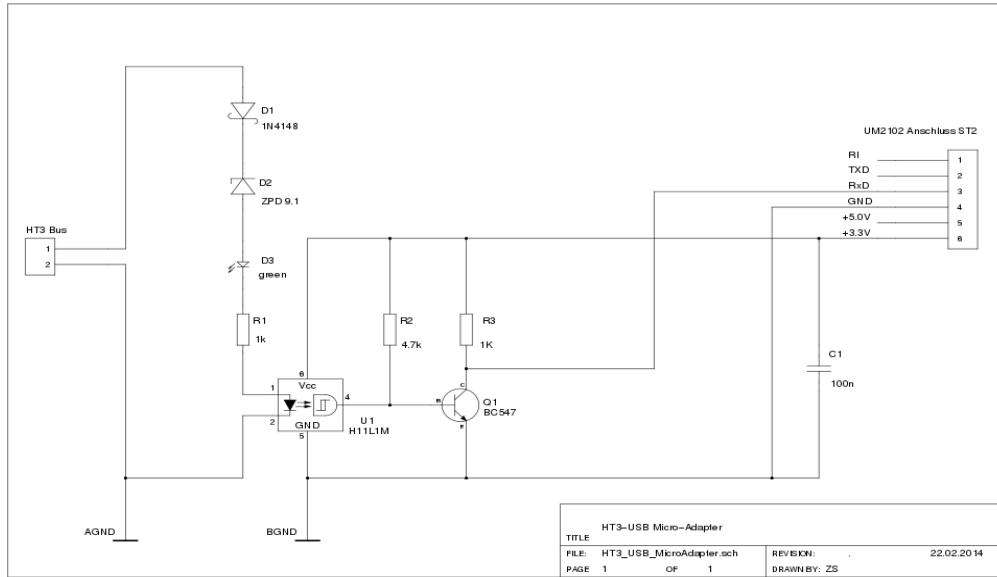


Abbildung 8: Schaltplan HT3 Microadapter (RS232->USB)

Name	Bezeichnung / Type	Anzahl	Bemerkung
--	Bauteile siehe Stückliste für den HT3-Miniadapter (RaspberryPi)	1 Satz	Siehe Tabelle 1: HT3 Adapterstückliste
UM2102N	Mini USB-Modul Bausatz	1	ELV: Artikel-Nr.: 150952
ST1-ST3	Stiftleiste RM 2.54 mm 20-polig	1	Conrad: Bestell-Nr.: 741105-62
BU1-BU3	Buchsenleiste 2.54 mm 20-polig (für das Mini USB-Modul)	1	Conrad: Bestell-Nr.: 733755-62

Tabelle 2: Stückliste HT3-Microadapter

1.3 Transceiver Frontend (*ht-transceiver*) für Raspberry Pi® / USB

Das 'ht_transceiver' Frontend ist ein Adapter für den RaspberryPi, der aber auch mit einem USB-Interface genutzt werden kann.

Er erlaubt die ('artgerechte') Kommunikation (RX/TX) des Host mit dem Heizungsbus.

Im Gegensatz zu den bisher vorgestellten Adapters ist dieser Adapter mit einer Atmel-CPU bestückt. Dieser hat die Aufgabe die Heizungs-Busprotokolle zu erfassen / senden, Endekennungen der Protokolle (Break-Signale) zu erkennen (RX) / senden (TX) und die Daten an die Auswertung weiterzureichen.

Der 'ht_transceiver' hat zwei serielle Schnittstellen, eine für den Heizungsbus und eine für die Kommunikation mit dem Host. Der Adapter wird signaltechnisch zwischen Heizungs-Bus und die Host-Schnittstelle geschaltet und dies erlaubt die weitere Nutzung der schon vorhandenen Analyse- / Logger-Software ohne größere Anpassungen.

Einzig die Baudrate zum Host-Analyse/Logger-Programm muss von 9600Baud auf 19200Baud erhöht werden.

Erstmals vorgestellt worden ist der 'ht_transceiver' auf der Forum-Seite [7]:

<https://www.mikrocontroller.net/topic/317004#3925213>.

Dort sind auch weitergehende Informationen (Grund warum aktiver Adapter etc.) vorhanden.

In den folgenden Kapiteln sind zwei verschiedene Versionen des 'ht_transceiver'-Adapters beschrieben, funktionell unterscheiden sie sich jedoch nicht.

Alle HW-Schnittstellen sind PIN-kompatibel, dies gilt auch für die SPI-Verbindung zum RaspberryPi. Somit ist die Programmierung der 'ht_transceiver' auch im eingebauten Zustand mit dem RaspberryPi möglich (avrdude).

Die Software ist mit Atmel Studio 6.2 erstellt und steht auf [github](#) [9] zur Verfügung.

Die Erfassung der Heizungsdaten ist mit allen hier beschriebenen Adapter-Versionen möglich.

Die Steuerung ist jedoch bei den Fxyz-Reglern vom Fertigungsdatum abhängig.

Die Cxyz-Regler erlauben alle die Steuerung der Heizung.

Die folgende Tabelle zeigt die Abhängigkeit der FD-Kennzahl vom Fertigungsdatum.

Das Fertigungsdatum der Fxyz-Regler kann man wie folgt bestimmen:

1. Menue-geführt (Fachmannebene; Menue-Taste länger als 3 Sekunden drücken)
FACHMANN EBENE: System Info
Fertigungsdatum des Reglers

Diese Angaben sind der FW100 Bedienungsanleitung entnommen, werden aber für andere Fx-Regler in ähnlicher Form vorhanden sein.

2. FD-Kennzahl (Fertigungsdatum) auf der Innenseite der Reglerabdeckung
Dazu den abgelesenen Wert: (FDxyz) mit folgender Tabelle vergleichen
und das Datum bestimmen.
Bei den System-Modulen ist am jeweiligen Typenschild das Fertigungsdatum zu entnehmen.

Dreistellige Fertigungsdatumskennzahlen:												
Jahr/Monat	Jan.	Feb.	März	April	Mai	Juni	Juli	Aug.	Sep.	Okt.	Nov.	Dez.
1980	041	042	043	044	045	046	047	048	049	050	051	052
1981	141	142	143	144	145	146	147	148	149	150	151	152
1982	241	242	243	244	245	246	247	248	249	250	251	252
1983	341	342	343	344	345	346	347	348	349	350	351	352
1984	441	442	443	444	445	446	447	448	449	450	451	452
1985	541	542	543	544	545	546	547	548	549	550	551	552
1986	641	642	643	644	645	646	647	648	649	650	651	652
1987	741	742	743	744	745	746	747	748	749	750	751	752
1988	841	842	843	844	845	846	847	848	849	850	851	852
1989	941	942	943	944	945	946	947	948	949	950	951	952
1990	061	062	063	064	065	066	067	068	069	070	071	072
1991	161	162	163	164	165	166	167	168	169	170	171	172
1992	261	262	263	264	265	266	267	268	269	270	271	272
1993	361	362	363	364	365	366	367	368	369	370	371	372
1994	461	462	463	464	465	466	467	468	469	470	471	472
1995	561	562	563	564	565	566	567	568	569	570	571	572
1996	661	662	663	664	665	666	667	668	669	670	671	672
1997	761	762	763	764	765	766	767	768	769	770	771	772
1998	861	862	863	864	865	866	867	868	869	870	871	872
1999	961	962	963	964	965	966	967	968	969	970	971	972
2000	081	082	083	084	085	086	087	088	089	090	091	092
2001	181	182	183	184	185	186	187	188	189	190	191	192
2002	281	282	283	284	285	286	287	288	289	290	291	292
2003	381	382	383	384	385	386	387	388	389	390	391	392
2004	481	482	483	484	485	486	487	488	489	490	491	492
2005	581	582	583	584	585	586	587	588	589	590	591	592
2006	681	682	683	684	685	686	687	688	689	690	691	692
2007	781	782	783	784	785	786	787	788	789	790	791	792
2008	881	882	883	884	885	886	887	888	889	890	891	892
2009	981	982	983	984	985	986	987	988	989	990	991	992
2010	001	002	003	004	005	006	007	008	009	010	011	012
2011	101	102	103	104	105	106	107	108	109	110	111	112
2012	201	202	203	204	205	206	207	208	209	210	211	212
2013	301	302	303	304	305	306	307	308	309	310	311	312
2014	417	418	419	420	453	454	455	456	457	458	459	460
2015	517	518	519	520	553	554	555	556	557	558	559	560
2016	617	618	619	620	653	654	655	656	657	658	659	660
2017	717	718	719	720	753	754	755	756	757	758	759	760
2018	817	818	819	820	853	854	855	856	857	858	859	860
2019	917	918	919	920	953	954	955	956	957	958	959	960
2020	037	038	039	040	073	074	075	076	077	078	079	080

Vier-/Fünfstellige Fertigungsdatumskennzahlen:
(J) JTTT (z.B. 9238 = 238. Tag 2009)

Tabelle 3: FertigungsDatum Kennzahlen (FD)

Für die Steuerung der Heizung mit einem Fxyz-Regler ist minimal das Fertigungsdatum:
'September 2008' := FD889 (dreistellig) bzw.
,1. September 2008':= 8383 (vierstellig)
erforderlich.

1.3.1 Adapter 'ht_piduino'

Der erste 'ht_transceiver'-Adapter (Name: '**ht_piduino**') ist mit einem ATmega328P-PU realisiert worden. Hinweise zur Hardware-Realisierung gibt es viele im www, besonders gute vom Arduino(TM) Uno Rev3 -Referenzboard und von der Internetseite [6].

Da der ATmega328 nur eine UART-Schnittstelle hat (Interface zum Heizungsbust), ist die zweite Schnittstelle als Software-UART realisiert worden (als Interface zum Host).

Der Adapter hat zwei Optokoppler und ist damit galvanisch getrennt vom Heizungsbust.

Obwohl der Atmega328P-PU ein relativ großes Gehäuse hat, passt dieser auf eine Platine für den RaspberryPi.

Die Platine ist sehr dicht bestückt (siehe Bild) und daher ist auch kein Platz mehr für Erweiterungen vorhanden. Als Nachteil hat sich der geringe Freiraum über der RaspberryPi CPU (SoC) herausgestellt. Es kann kein Kühlkörper mehr verwendet werden und bei einem geschlossenen RaspberryPi-Gehäuse sind thermische Probleme nicht ausgeschlossen.

Dies führte zum Entschluss einen zweiten Adapter ('ht_piduino') zu entwickeln.

Nutzt man den ht_piduino-Adapter mit einem USB<->UART Wandler (ohne RaspberryPi), so ist dieser u.U. die bessere Wahl, da er mehr Platz im Flash für Programmerweiterungen hat

(Bemerkung: 32kByte ATmega328 gegenüber 8kByte ATTiny841,
z.Zeit sind ca. 4 bis 5 kByte Programmcode für den 'ht_transceiver' benutzt)

1.3.1.1 'ht_piduino' Bild, Schaltplan und Stückliste



Abbildung 9: ht_piduino Adapter (bestückt)

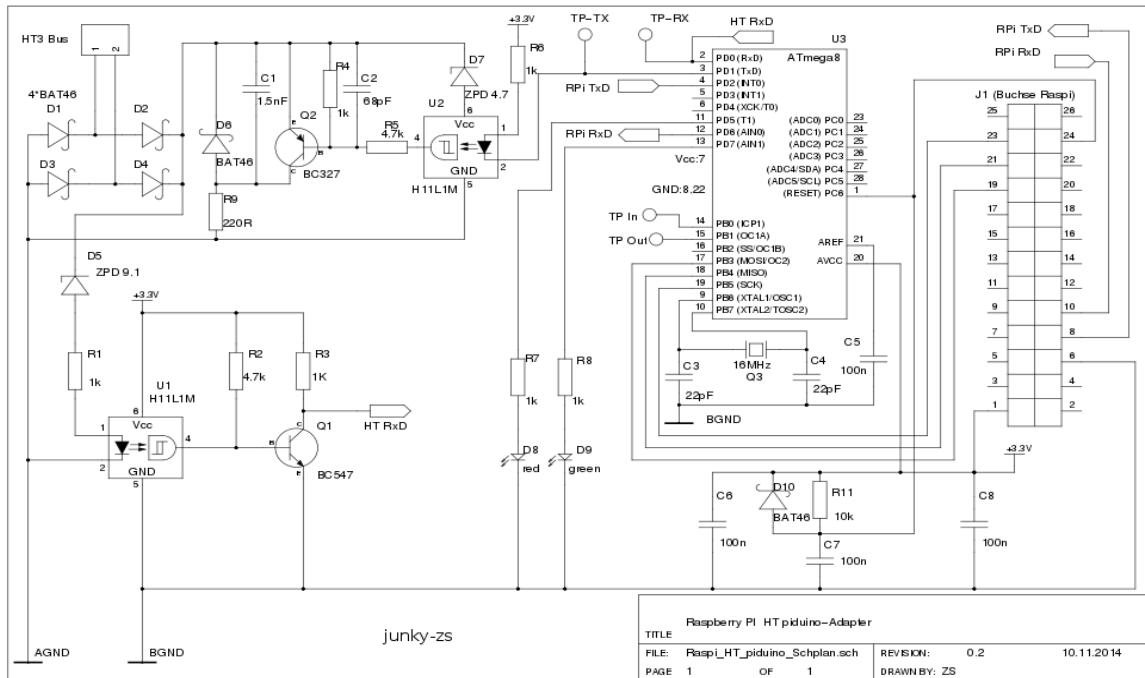


Abbildung 10: 'ht_piduino' Schaltplan (mit ATmega328P !)

Name	Bezeichnung / Type	Anzahl	Bemerkung
--	Bauteile siehe Stückliste für den ht_piduino (RaspberryPi) unter dem nebenstehenden Link auf 'raspi_ht_piduino'.	1 Satz	Siehe Stückliste unter: < http://www.reichelt.de/index.html? &ACTION=20&AWKID=998298&PROVID=2084 >
U1/U2	H11L1M (oder PC900V)	1	Den hat 'reichelt' nicht im Programm. (Will aber auch hier keine Werbung für den Lieferanten machen, das macht er schon selber mit seinem schnellen Lieferservice)
U3	ATmega328P-PU	1	Den hat 'reichelt' im Programm. Achtung: Schaltplan zeigt den pinkompatiblen ATmega8, die Software ist jedoch für den ATmega328P erstellt !

Tabelle 4: Stückliste 'ht_piduino'-Adapter

1.3.2 Adapter 'ht_pitiny'

Der zweite 'ht_transceiver'-Adapter (Name: '**ht_pitiny**') ist mit einem ATtiny841 realisiert worden. Diese CPU hat u.A. zwei serielle Schnittstellen (UART's) zur Verfügung und benötigt daher keinen SW-UART.

Zusätzlich ist die CPU im SOT14 Gehäuse sehr klein und durch die SMD-Bestückung des Adapters ist dieser auch recht kompakt geworden (siehe Bild).

Auch dieser Adapter hat zwei Optokoppler und ist somit galvanisch getrennt vom Heizungsbust. Der Programmcode belegt z.Zeit ca. 4kByte und lässt somit Platz für Erweiterungen.

1.3.2.1 'ht_pitiny' Bild, Schaltplan und Stückliste

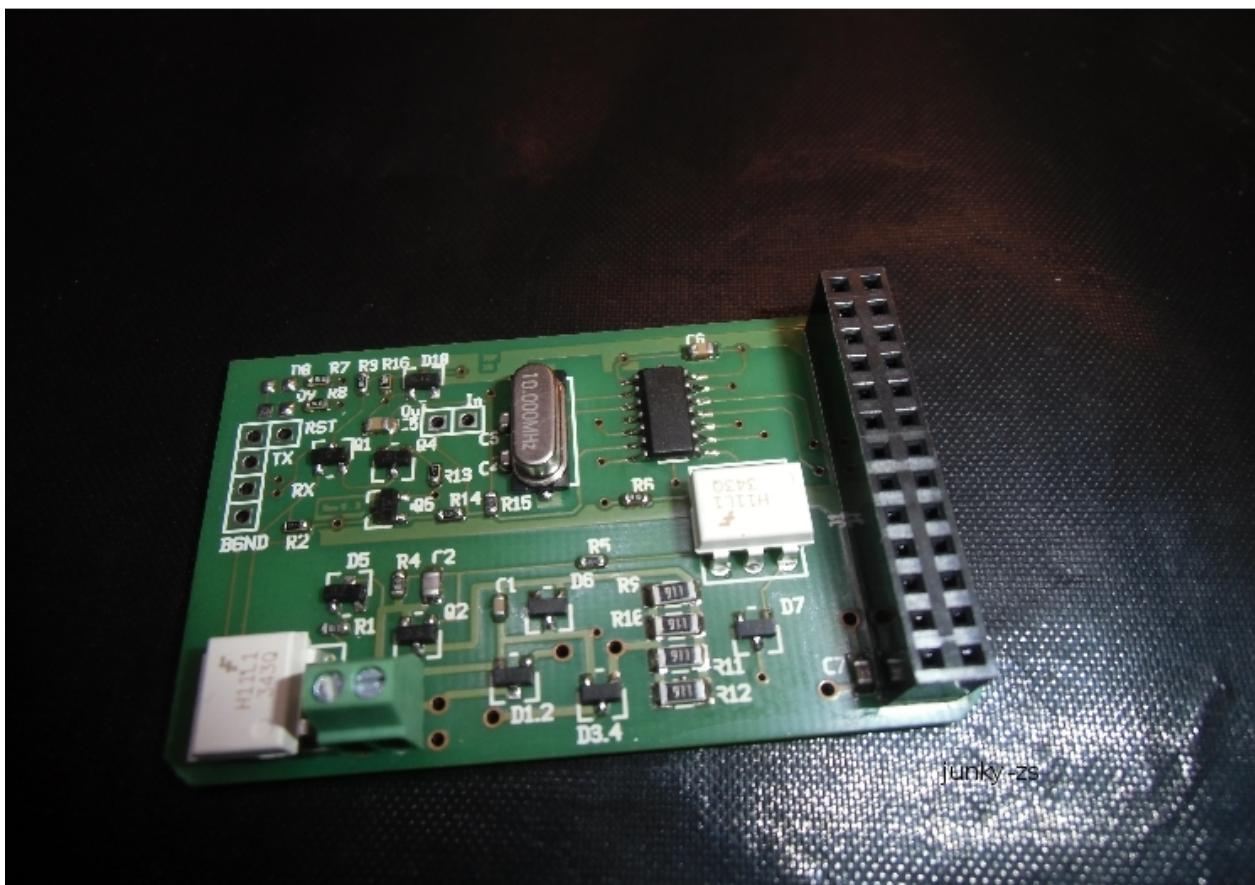


Abbildung 11: ht_pitiny Adpater (bestückt)

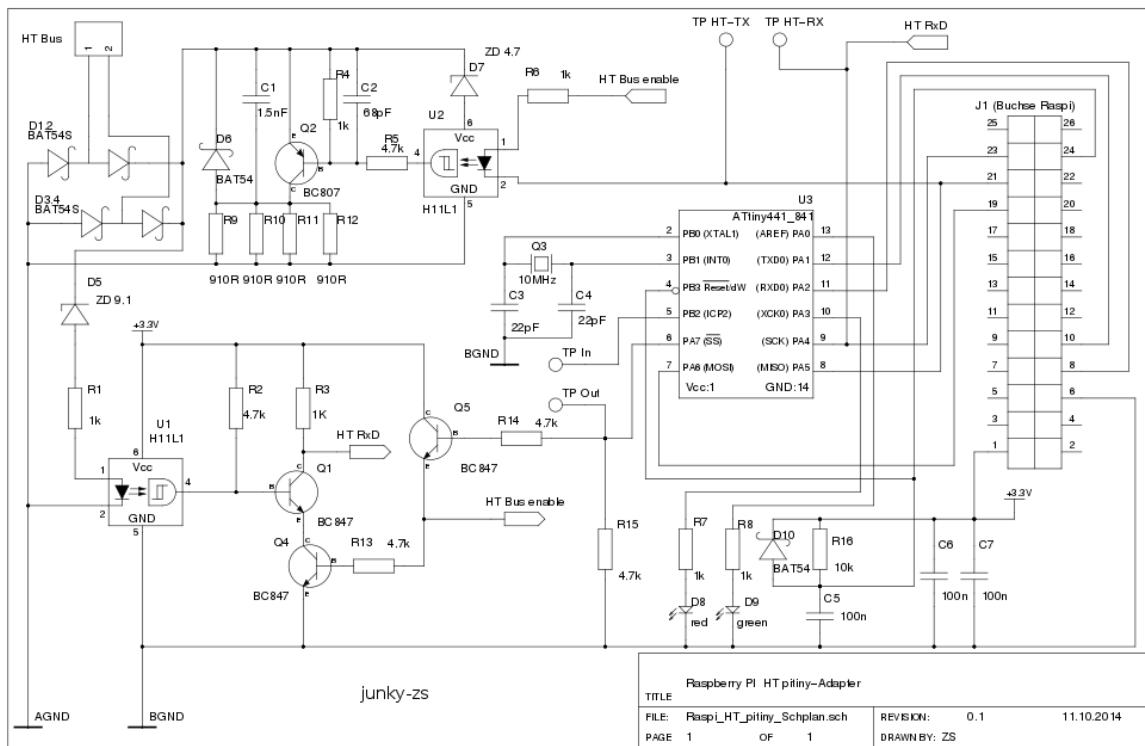


Abbildung 12: ht_pitiny Schaltplan (Bestückung nur mit ATtiny841 !)

Name	Bezeichnung / Type	Anzahl	Bemerkung
--	Bauteile siehe Stückliste für den ht_pitiny (RaspberryPi) unter dem nebenstehenden Link auf 'raspi_ht_pitiny'.	1 Satz	Siehe Stückliste unter: http://www.reichelt.de/?ACTION=20;AWKID=1189494;PROVID=2084
U1/U2	H11L1M (oder PC900V)	1	Den hat 'reichelt' nicht im Programm. (Will aber auch hier keine Werbung für den Lieferanten machen, das macht er schon selber mit seinem schnellen Lieferservice)
U3	ATtiny841	1	Den hat 'reichelt' auch nicht im Programm, ja Schitt can happen! Da sind andere Lieferanten am Zug der Zeit wie: 'mouse', 'Farnell (HBE)' und andere Ali- Express-Züge etc. Achtung: Schaltplan zeigt auch den pinkompatiblen ATtiny441, dieser hat jedoch zu wenig RAM/Flash-Speicher! Daher unbedingt den ATtiny841 verwenden!

Tabelle 5: Stückliste 'ht_pitiny'-Adapter

1.3.3 USB-MotherBoard für Adapter 'ht_transceiver' und UM2102N

Die 'ht_transceiver'-Adapter ('ht_piduino' und 'ht_pitiny') können auch ohne den RaspberryPi mit einem USB-ADAPTER (z.B.:ELV UM2102N) betrieben werden. Dazu ist ein passives Motherboard realisiert, welches die Verbindungen zwischen 'ht_transceiver', UM2102N und einem optionalen ISP-Anschluss zur Verfügung stellt. Der folgende Schaltplan zeigt, wie die Verbindungen zu realisieren sind.

Der UM2102N stellt die Betriebsspannung von 3.3Volt für den 'ht_transceiver' zur Verfügung. Die UART-Anschlüsse RX/TX sind mit zugehörigen Pins am USB-Adapter verbunden.

Über den optionalen Anschluss ST3 (ISP) kann die Atmega CPU programmiert werden. Die Brücke J1 sorgt für 3.3 Volt ODER 5.0 Volt Spannung für einen passiven ISP-Programmer (für den 'ht_transceiver' ist J1 immer auf 3.3Volt gesteckt).

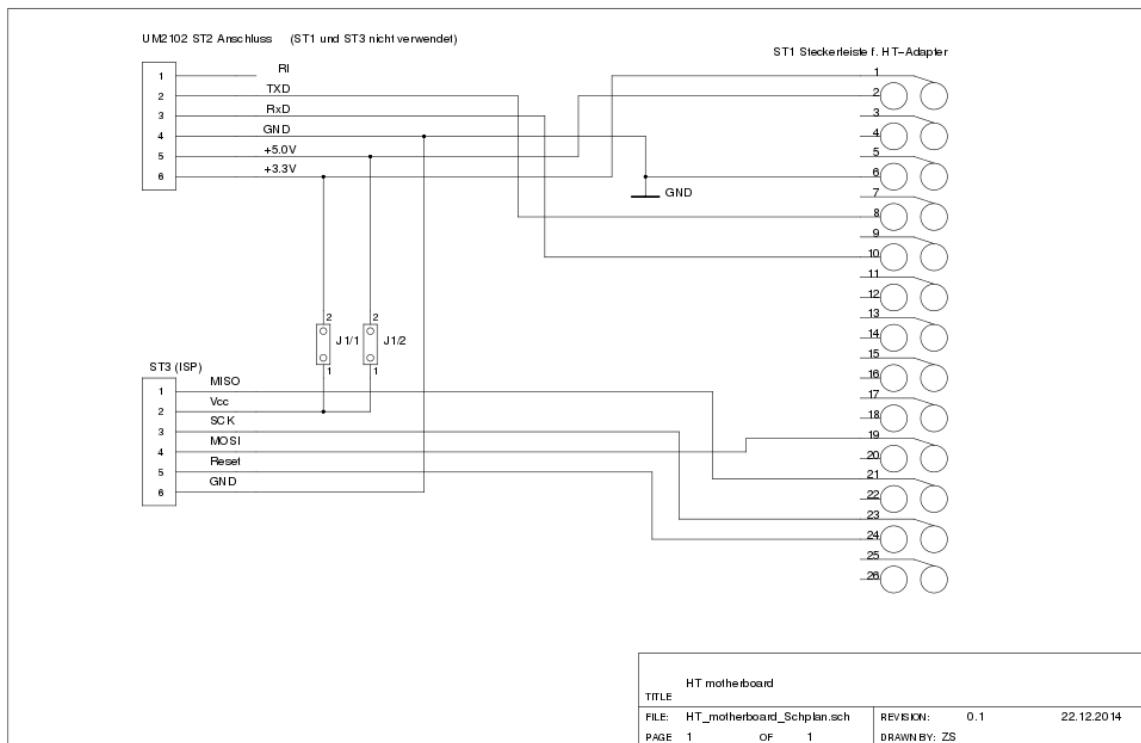


Abbildung 13: Motherboard für ht_transceiver

2 Software

Die Software ist in Python geschrieben. Der Grund ist die leichtere Portierbarkeit auf verschiedene Betriebssysteme. Es wird die Version 3.x (Python3) verwendet, Versionen darunter werden nicht unterstützt (siehe SW-Quellen unter Link [8]).

Es sind unterschiedliche Schnittstellen realisiert die je nach Bedarf aktiviert werden können bzw. aktiviert sind.

Folgende Schnittstellen sind z. Zeit realisiert:

Bezeichnung	Funktion	Default aktiv	Bemerkung
Sqlite - IF	SQLite - Datenbank	Nein	Daten werden in eine sqlite-Datenbank beschrieben.
Rrdtool - IF	Rrdtool - Datenbank	Ja	Round Robbin Datenbank mit grafischer Ausgabe der Daten.
MQTT - IF	MQTT Client Interface	Nein	Client Interface zur Verteilung der dekodierten Daten (publish) und zur Steuerung der Heizung mit Befehlen (subscribe). Für den Betrieb ist ein MQTT-Broker erforderlich.
SPS - IF	<u>Speicher</u> <u>Programmierbare</u> <u>Steuerungs</u> <u>Schnittstelle</u>	Nein	SPS Client Schnittstelle die dekodierte Daten auf Abfrage hin bereitstellt.
Hassio - IF	HomeAssistant MQTT-Schnittstelle	Nein	MQTT Subscribe-/Publish-Schnittstelle für die HomeAssistant App.

Tabelle 6: Aktivierbare Schnittstellen

Für die Schnittstelle zur rrdtool-Datenbank wird z. Zeit noch die Programmiersprache 'Perl' genutzt, da die Python3-Module nicht verfügbar waren.

Inzwischen (> 2015) gibt es eine Realisierung (import rrdtool) [10] die einen Ersatz der Perl-Scripte denkbar macht. Eine Realisierung steht noch aus.

Durch Konfigurationsänderungen lässt sich die rrdtool-Datenbank abschalten und der Betrieb nur mit der SQL-Datenbank durchführen. Allerdings entfällt dann auch die grafische Ausgabe der Daten.

Die Konfigurationsdaten sind in XML-Dateien abgelegt und dienen u.A. zur Erzeugung der Datenstrukturen der 'SQLite'- und 'rrdtool'-Datenbanken.

2.1 Verzeichnis-Struktur

Die Verzeichnis-Struktur sieht wie folgt aus:

Persönlicher Ordner workspace HT3 software var databases				Suchen
Name	Größe	Typ	Änderungsdatum	
+ etc	1 Objekt	Ordner	Di 04 Feb 2014 21:00:11 CET	
+ lib	10 Objekte	Ordner	Do 06 Feb 2014 18:09:08 CET	
+ var	2 Objekte	Ordner	Di 04 Feb 2014 21:00:11 CET	
create_databases.py	2,0 kB	Python-Skript	Mi 05 Feb 2014 19:08:57 CET	
HT3_Analyser.py	1,4 kB	Python-Skript	Mi 05 Feb 2014 18:18:33 CET	
HT3_LOGGER.py	1,4 kB	Python-Skript	Mi 05 Feb 2014 18:59:25 CET	
HT3_Systemstatus.py	1,4 kB	Python-Skript	Mi 05 Feb 2014 19:01:16 CET	

Abbildung 14: Datei-Verzeichnisstruktur

Hinweis:

Das Tool: HT3_LOGGER.py ist ersetzt durch den ,ht_collgate.py‘.

Die Namensgebung: HT3_.... Wurde (2014) wegen des Heizungs-Bus: Heatronic3 gewählt. Es werden jedoch auch die neuen Telegramme der EMS-Gattung dekodiert und angezeigt.

Deshalb werden die neueren Software-Module jetzt mit ,ht_... bezeichnet. Eine Umbenennung aller Modulnamen ist jedoch noch nicht erfolgt.

Modulname	Funktion	Bemerkung
create_databases.py	Erzeugt die Datenbank SQLite und rrdtool falls diese noch nicht vorhanden sind. Genutzt wird die Konfiguration unter ./etc/config.	Vorhandene Datenbanken werden nicht überschrieben.
ht_collgate.py	Konfigurierbarer Daemon, der aktivierte Schnittstellen mit dekodierten Daten versorgt.	Es werden die Datenbanken sqlite und rrdtool sowie die Schnittstellen: mqtt und SPS unterstützt. Das Tool: HT3_LOGGER.py wird durch diesen daemon ersetzt.
HT3_Analyser.py	HT3 Bus Analyser mit grafischer Datenausgabe in Plain-Text und Hexadezimal mit zugehörigen Protokoll-Beschreibungen.	Daten werden in die vorhandenen Datenbanken geschrieben.
HT3_Systemstatus.py	HT3 Systemstatus mit grafischer Datenausgabe in Plain-Text und Schreiben der Daten in die Datenbanken.	Daten werden in die vorhandenen Datenbanken geschrieben.
ht_proxy.py	Proxy-server für die Kommunikation zwischen Comport und Socket-Schnittstellen.	Comport Proxy-Server, der Socket-verbindungen für Clients bereitstellt.

ht_netclient.py	Socket-Client für die Heizungssteuerung.	Über diesen Client kann die Heizung gesteuert werden. Es werden dazu NetCom ähnliche Telegramme verwendet.
ht_binlogclient.py	Socket-Client für das Mitschreiben der RAW-Heizungsdaten in ein Logfile.	Erfassen und schreiben der RAW-Heizungdaten in ein Logfile unter: ~/HT3/sw/var/log.
ht_2hassio.py	MQTT - daemon als Schnittstelle zur HomeAssistant Applikation.	Die empfangenen MQTT-Daten werden für den HA aufbereitet und zum MQTT-Broker gesendet.

Tabelle 7: Software-Modulinformationen

2.2 Konfiguration

Im Verzeichnis 'etc/config' ist die Konfiguration zur Erzeugung und Nutzung der Datenbanken und Softwaremodule abgelegt.

Die XML-Datei 'HT3_db_cfg.xml' enthält alle Details für den 'Normalbetrieb'. Die XML-Dateien unter dem '4test'-Verzeichnis sind für den Testbetrieb der Python-Module vorgesehen.

Name	Größe	Typ	Anderungsdatum
config	2 Objekte	Ordner	Di 04 Feb 2014 21:00:11 CET
4test	3 Objekte	Ordner	Di 04 Feb 2014 21:00:11 CET
HT3_db_cfg.xml	11,8 kB	XML-Dokument	Di 04 Feb 2014 21:00:11 CET

Abbildung 15: Konfigurations-Verzeichnis

Die Einträge in der HT3_db_cfg.xml Datei werden für die Aktivierung (rrdtool), Namensgebung der Tabellen und Tabellen-Spalten (SQLite) und Startzeiten / Schrittweiten (rrdtool) benutzt.

Das File: collgate_cfg.xml ist für die Konfiguration des ht_collgate daemon erforderlich.

Die einzelnen Schnittstellen können je nach Bedarf aktiviert / deaktiviert werden.

Das File: ht_proxy_cfg.xml ist für die Konfiguration des ht_proxy - Server und - Client erforderlich. Darin enthalten sind u. A. die Infos zur Host-Adresse und Port-Nummer sowie auch die Daten der seriellen Schnittstelle.

Das File: mqtt_client_cfg.xml ist für die Konfiguration des MQTT - Clients vorgesehen.

Darin enthalten sind u. A. die Host-Adresse und Port-Nummer des MQTT - Broker.

Das File: SPS_cfg.xml ist für die Konfiguration des SPS - Client / Server vorgesehen.

Darin enthalten sind u.A. die Host-Adresse und Port-Nummer für den Zugriff auf den SPS-server.

Abbildung 16: Konfigurationsfile-Inhalt (Beispiel)

```
....<dbname_sqlite>./var/databases/HT3_db.sqlite</dbname_sqlite>
....<sql-db>
.....<enable>on</enable>
....</sql-db>
....<!-- rrdtool-database -->
....<dbname_rrd>./var/databases/HT3_db_rrd</dbname_rrd>
....<!-- 'dbname_rrd' without suffix, is used only as leading path & name
.....and to create rrdtool db-files for any 'systempart' with there
.....own name and suffix
....-->
....<rrdtool-db>
.....<enable>on</enable>
.....<step_seconds>60</step_seconds>
.....<starttime_utc>1344000000</starttime_utc>
....</rrdtool-db>

....<!-- global configuration-values -->
....<anzahl_heizkreise>1</anzahl_heizkreise>

....<systempart name="heizgeraet">
.....<shortname name="HG"/>
.....<hardwaretype>CSW</hardwaretype>...<!-- Wert wird nur in GUI angezeigt.
.....<logitem name="T_vorlauf_soll">
.....<datatype>INT</datatype>
.....<datause>GAUGE</datause>
.....<maxvalue>100</maxvalue>
.....<default>0</default>
.....<unit>Grad</unit>
.....<displayname>T-Soll (Regelung)</displayname>
....</logitem>
....<logitem name="T_vorlauf_ist">
.....<datatype>REAL</datatype>
.....<datause>GAUGE</datause>
.....<maxvalue>100.0</maxvalue>
.....<default>0.0</default>
.....<unit>Grad</unit>
.....<displayname>T-Ist (Vorlauf)</displayname>
....</logitem>
```

Auszug aus dem Konfigurations-File.

Die Bedeutung der einzelnen Parameter ist auf den folgenden Seiten beschrieben.

Parameter	Werte	Funktion
<dbname_sqlite>	Pfad und Name wählbar. Das Verzeichnis muss vorhanden sein.	Pfad und Name der SQL-Datenbank. Die Datenbank wird erzeugt, sofern das Verzeichnis vorhanden ist. Das Verzeichnis wird <u>nicht</u> angelegt.
<sql-db> <enable>	on oder 1 ->> Enable off oder 0 ->> Disable (Gross/Kleinschreibung erlaubt)	Aktiviert die Datenbank 'sqlite'. Es wird die Datenbank erzeugt, falls diese noch nicht vorhanden ist. Jeder andere Wert als 'on' /1 deaktiviert die Datenbank.
<sql-db> <autoerase_olddata>	Default: 30 Tage 0 ->> Disable	Automatisches Löschen der sqlite-Datenbank Einträge die älter sind als 30 Tage. Der Wert:0 deaktiviert diese Funktion.
<dbname_rrdtool>	Pfad und Name wählbar. Das Verzeichnis muss vorhanden sein.	Pfad und Name der rrdtool-Datenbank. Die Datenbank wird erzeugt, sofern das Verzeichnis vorhanden ist. Das Verzeichnis wird <u>nicht</u> angelegt.
<rrdtool-db> <enable>	on oder 1 ->> Enable off oder 0 ->> Disable (Gross/Kleinschreibung erlaubt)	Aktiviert die Datenbank 'rrdtool'. Es wird die Datenbank erzeugt, falls diese noch nicht vorhanden ist. Jeder andere Wert als 'on' /1 deaktiviert die Datenbank.
<rrdtool-db> <step_seconds>	Default: 60 Sekunden	Der Wert bestimmt das Aktualisierungs-Intervall der rrdtool-Datenbank. Mit diesem Intervall werden die Daten der SQLite-Datenbank in die rrdtool-Datenbank eingetragen. Kleinere Werte als 60 Sekunden sind nicht möglich (und auch nicht sinnvoll).
<rrdtool-db> <starttime_utc>	Default: 1344000000 (entspricht: 13:30:00 03.08.2012)	Frühestmöglicher Zeitstempel für rrdtool-Datenbankeinträge.
<rrdtool-db> <autocreate_draw>	Default: 2 Minuten 0 ->> Disable	Die Grafik der rrdtool-Datenbank Einträge wird alle 2 Minuten erzeugt. Der Wert:0 deaktiviert diese Funktion. Nach dem Start der Applikation wird dies nach 4 Minuten zum ersten mal durchgeführt.
<data_interface> <comm_type>	ASYNC oder SOCKET	Übertragungs-Eigenschaft des Dateninterfaces. ASYNC := seriell, asynchron; SOCKET:= IP mit Port (noch in Entwicklung)
<data_interface> <proto_type>	RAW oder TRX	Protokoll-Art des Dateninterfaces. RAW := transparente Datenübertragung. TRX := Daten mit Protokoll-Header (noch in Entwicklung)
<data_interface> <parameter name = „ASYNC“>	Parameter für Dataif: ASYNC und SOCKET	Parametern für Seriell, asynchron- und Socket-Verbindungen.
<data_interface> <parameter name = „ASYNC“> <serialdevice>	Device-Name: /dev/ttyAMA0 oder /dev/ttyUSB0	Device-Name der Schnittstelle unter Linux.

Parameter	Werte	Funktion
<data_interface> <parameter name = „ASYNC“> <inputtestfilepath>	Defaultwert := leer.	Wird an dieser Stelle ein Pfad mit Filename eines Binären Logfiles angegeben, so wird anstelle der Daten des DataInterface die Informationen des Binären Files ausgewertet. (Dies jedoch nicht in Echtzeit sondern schneller)
<data_interface> <parameter name = „ASYNC“> <baudrate>	Defaultwert := 9600 oder mit 'ht_transceiver' := 19200	Baudrate der seriellen Datenschnittstelle. Ist der Erfassungsadapter 'ht_transceiver' angeschlossen, so ist die Baudrate auf := 19200 einzustellen.
<data_interface> <parameter name = „ASYNC“> <config>	Defaultwert:=“8N1“	Konfigurations-Parameter der seriellen Schnittstelle. Dieser wird z.Z. nicht ausgewertet.
<data_interface> <parameter name = „SOCKET“> <client_config_file>	Konfig-File für proxy-Server und Client.	Konfigurations-Filename für Socket-Verbindung.
<logging> <path>	./var/log	Relativer Pfad zum Log-Verzeichnis. Das Verzeichnis muss vorhanden und beschreibbar sein.
<logging> <default_filename>	ht_default.log	Verwendeter Logfilename falls von der Applikation kein anderer Name übergeben wird.
<logging> <loglevel>	DEBUG, INFO, WARNING, ERROR, CRITICAL	Loglevel, der je nach Wert die aufgezeichneten Informationen / Meldungen beeinflusst.
<anzahl_heizkreise>	1...4	Gibt die Anzahl der Heizkreise des Systems an. Maximale Wert ist z.Zeit := 4 (0 ist nicht erlaubt). Dies ist ein <u>Startwert</u> für den Betrieb der Applikationen. Dieser Wert wird dynamisch angepasst durch den Empfang der zugehörigen Heizkreis-Telegramme.
<systempart name="">	heizgeraet heizkreis1 heizkreis2 heizkreis3 heizkreis4 warmwasser solar sysdatetime	Heizsystemanteile, für die Daten auf dem HT3-Bus gesendet werden. Die Namensgebung entspricht dabei denen der FWxyz - Reglerserie. Mit diesem Namen werden die Tabellen in der SQL-Datenbank erzeugt. Bei der rrdtool-Datenbank wird dieser Name auch als File-Namenserweiterung verwendet. (Hinweis: Die 'sysdatetime'-Daten werden zwar als Systemzeit angezeigt und in die SQL-Datenbank eingetragen, jedoch nicht in die 'rrdtool'-Datenbank übernommen)

Parameter	Werte	Funktion
<systempart ...> <shortname name="">	HG HK1 HK2 HK3 HK4 WW SO DT	Kurzname des 'systempart'-Namen ->> 'Nickname'. Dieser wird in der Software für das interne Datenhandling verwendet. Es werden nur maximal die ersten drei Charakter benutzt, die somit eindeutig sein müssen. Groß/Kleinschreibung ist erlaubt.
<systempart ...> <hardwaretype>	CSW, KUB, ISM1, ISM2, IPM1, IPM2 etc.	Type der Hardware, welcher diese Systempart-Daten bereitstellt. Dieser Type-Name wird in der GUI im Systempartteil dargestellt und kann auch leer bleiben. Aus diesem Namen werden <u>keine</u> Systemkonfigurationen abgeleitet.
<systempart ...> <logitem name="">	T_vorlauf_soll T_vorlauf_ist T_ruecklauf hexdump	Mit diesen Namen wird die SQL-Datenbank (Columns) und die rrdtool Datenitems erzeugt. Innerhalb eines <systempart>-Bereichs muss dieser Name eindeutig sein. Im 'hexdump' wird das zugehörige Datentelegramm in Hexform abgespeichert. Eine nachträgliche Veränderung des Namens ist nach der Erzeugung und Nutzung der Datenbanken zu vermeiden und führt u.U. zu Datenverlust bzw. aufwendigen Anpassungen und Korrekturen.
<systempart name="heizkreis(x)" "> <unmixed>	Flag (True/False) wobei: (x):= 1...4	Dieses Flag bestimmt, ob der Heizkreis keinen (True) oder einen Mischer (False) hat. Der Wert wird dynamisch durch den Empfang des zugehörigen Telegramms gesetzt und die GUI-Anzeige damit gesteuert.
<systempart name="warmwasser" "> <load_pump>	Flag (True/False)	Dieses Flag bestimmt, ob die Warmwasser-Erzeugung Teil des Heizgerätes (False) oder als externer Wasserspeicher mit separater Ladepumpe (True) vorhanden ist. Es wird nur die GUI-Anzeige damit gesteuert, die Erfassung wird dadurch nicht beeinflusst.
<systempart name="solar"> <second_heater>	Flag (True/False)	Dieses Flag bestimmt, ob es ein zweites Heizsystem (True) gibt oder nicht (False). Dieses zweite Heizsystem (Feststoff-Kessel etc.) wird z.Z. dem Solar-System als „Hybrid-Anteil“ zugeordnet. In der Regel gibt es dann einen separaten Puffer-Speicher, dessen Werte mit Systemmodulen (z.B. ISM2) überwacht werden. Es wird nur die GUI-Anzeige damit gesteuert, die Erfassung wird dadurch nicht beeinflusst.
<systempart name="solar"> <second_buffer>	Flag (True/False)	Dieses Flag bestimmt, ob es einen separaten Pufferspeicher im System gibt (True) oder nicht (False). Dieser wird in der Regel durch Systemmodule (z.B. ISM2) überwacht. Es wird nur die GUI-Anzeige damit gesteuert, die Erfassung wird dadurch nicht beeinflusst.

Parameter	Werte	Funktion
<logitem name=""> <datatype>	INT REAL TEXT	Wird bei der Erzeugung der SQL-Datenbank als Datentyp genutzt.
<logitem name=""> <datause>	GAUGE COUNTER ... weitere siehe rrdtool	Wird bei der Erzeugung der rrdtool-Datenbank als Logitem-type genutzt. Zur Zeit wird nur der Type: GAUGE verwendet.
<logitem name=""> <maxvalue>	Logitem abhängig	Maximale Wert des 'Logitems'. Wird für die Überprüfung der Messwerte auf den maximal zulässigen Wert verwendet. Wird dieser Maximalwert überschritten, so wird dieser Messwert auf den 'Defaultwert' eingestellt.
<logitem name=""> <default>	Logitem abhängig	Default Wert des 'Logitems'. Wird für die Initialisierung und Rücksetzung der Messwerte verwendet (siehe auch 'maxvalue').
<logitem name=""> <unit>	Grad Status % Minuten Zähler Wh kWh	Dieser Wert wird für die grafische Anzeige (GUI) verwendet und ist frei wählbar.
<logitem name=""> <displayname>	Text ist Logitem abhängig und frei wählbar	Dieser Wert wird für die grafische Anzeige (GUI) verwendet und ist frei wählbar. Ist der Wert leer, wird das Logitem <u>nicht in der GUI angezeigt</u> .
<logitem name=""> <accessname>	Frei wählbarer Name für den Datenzugriff auf den Logitem-Wert.	Über diesen Namen kann auf den Wert des Logitems zugegriffen werden. Diese Funktion wird von den Schnittstellen 'MQTT - Client IF', 'SPS IF' und ',ht_2hassio' - IF genutzt.
<logitem name=""> <set_param>	Eine Zeichenkette, die die möglichen Kommando-Parameter festlegt. Beispiele: Tniveau,2,1,auto manual oder Tdesired,3,4,T,heizen sparen temporary comfort1...	Trennzeichen der Parameter ist ','. Bedeutung: 1. Parameter = Kurzname der Aktion (Tniveau) 2. Parameter = Anzahl der folgenden Parameter 3. Parameter = Nummer des Heizkreises 3. Parameter = Auswahlliste oder 'T' (hängt vom Befehl (accessname) ab). 4. Parameter = Optionale Auswahlliste (je nach Befehl) Trennzeichen: (Trennung zw. Heatronic- und EMS- Parametern) 5. Parameter = 1. EMS Parameter 6. Parameter = 2. EMS Parameter

Tabelle 8: Konfigurations-Parameterbeschreibung

Kommando	Set-Parameter	Bus-/Controller-Typ		Bemerkung
		Heatronic Fxyz	EMS Cxyz	
Tdesired	Temperatur, heizen	x	---	Es wird die Temperatur für das Temperatur-Niveau: heizen gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: heizen gerade aktiv ist.
Tdesired	Temperatur, sparen	x	---	Es wird die Temperatur für das Temperatur-Niveau: sparen gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: sparen gerade aktiv ist.
Tdesired	Temperatur, frost	x	---	Es wird die Temperatur für das Temperatur-Niveau: frost gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: frost gerade aktiv ist.
Tniveau	auto	x	---	Es wird das <u>aktive</u> Temperatur-Niveau von der Automatik (Programm) gesetzt. Die jeweilige zum Temperatur-Niveau zugehörige Temperatur wird aktiviert.
Tniveau	heizen	x	---	Es wird das <u>aktive</u> Temperatur-Niveau auf: heizen gesetzt.
Tniveau	sparen	x	---	Es wird das <u>aktive</u> Temperatur-Niveau auf: sparen gesetzt.
Tniveau	frost	x	---	Es wird das <u>aktive</u> Temperatur-Niveau auf: frost gesetzt.
Tdesired	Temperatur, temporary	---	x	Es wird die Temperatur für das Temperatur-Niveau: temporary gesetzt. Nach dem nächsten Programmwechsel wird wieder das dann aktuelle Temperatur-Niveau eingestellt.

Kommando	Set-Parameter	Bus-/Controller-Typ		Bemerkung
		Heatronic Fxyz	EMS Cxyz	
Tdesired	Temperatur, comfort1	---	x	Es wird die Temperatur für das Temperatur-Niveau: comfort1 gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: comfort1 gerade aktiv ist.
Tdesired	Temperatur, comfort2	---	x	Es wird die Temperatur für das Temperatur-Niveau: comfort2 gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: comfort2 gerade aktiv ist.
Tdesired	Temperatur, comfort3	---	x	Es wird die Temperatur für das Temperatur-Niveau: comfort3 gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: comfort3 gerade aktiv ist.
Tdesired	Temperatur, eco	---	x	Es wird die Temperatur für das Temperatur-Niveau: eco gesetzt. Diese Änderung ist nur dann sichtbar wenn das Niveau: eco gerade aktiv ist.
Tdesired	Temperatur, manual	---	x	Es wird die Temperatur für das Temperatur-Niveau: manuell gesetzt. Diese Änderung ist nur dann sichtbar wenn der Status: manuell gerade aktiv ist.
Tniveau	auto	---	x	Die aktive Temperatur wird Programmgesteuert je nach aktivem Temperatur-Niveau ausgewählt.
Tniveau	manual	---	x	Die aktive Temperatur wird fest auf den Wert des Temperatur-Niveaus: manuell eingestellt.

Tabelle 9: Set-Parameter je nach Regler-Typ

Die Set-Parameter Tabelle beschreibt die möglichen Parameter für Set-Kommandos, die zur Steuerung der Heizung dienen.

Dabei ist es wichtig den jeweiligen Typ des Reglers (Controllers) zu berücksichtigen. Daher ist es unbedingt erforderlich den Reglertyp und auch das Fertigungsdatum (FD) des Reglers festzustellen damit eine Steuerung (mit dem ht_transceiver) möglich ist.

Hinweis: Auch ein Reglertyp: Cxyz kann an einem Heatronic-**Bus** betrieben werden, trotzdem sind die Parameter für '**EMS**' erforderlich.

Die Zuordnung ist:

Regler-Name	Controller-Typ	Bemerkung
Fxyz - Regler	Heatronic	Alle Fxyz Regler, wie z.B. FW100, FR120 etc. Der FR50 allerdings ist nur mit aktueller Software für eine Steuerung geeignet. Dabei ist das Fertigungs-Datum (FD) wichtig.
CT100	EMS	CT100 Regler (Test steht noch aus)
Cxyz - Regler	EMS	Alle Cxyz Regler. (Test für CW100/400 durchgeführt)

Tabelle 10: Zuordnung des Regler-Namen zum Controller-Typ

rrdtool-Archivwerte	Archiv-Details	Bemerkung
LAST	saved every 5 minutes, kept for 10years back	Letzter Wert wird alle 5 Minuten gespeichert und über 10 Jahre gehalten. Danach werden die ältesten Daten überschrieben.
AVERAGE	saved every 1 minute, kept for 1year back	Der Durchschnittswert wird jede Minute gespeichert und ein Jahr gehalten. Danach werden die ältesten Daten überschrieben.
MAX	saved every 5 minutes, kept for 1year back	Der Maximalwert wird alle 5 Minuten gespeichert und ein Jahr gehalten. Danach werden die ältesten Daten überschrieben.
MIN	saved every 5 minutes, kept for 1year back	Der Minimalwert wird alle 5 Minuten gespeichert und ein Jahr gehalten. Danach werden die ältesten Daten überschrieben.

Tabelle 11: RRDTool Archiv-Details

(Details sind der rrdtool - Beschreibung zu entnehmen [2])

2.3 Schnittstellen

Im folgenden werden die unterschiedlichen Schnittstellen beschrieben, die von den einzelnen Applikationen genutzt werden.

2.3.1 Datenbanken

Im Verzeichnis 'sw/var/databases' sind die Datenbank-Dateien für 'SQLite' und 'rrdtool' abgelegt.

Diese werden erstmalig beim Aufruf: 'create_databases.py' mit den Informationen der Konfiguration erzeugt

(Hinweis: Die Heizkreise werden jetzt namentlich anders als in der Abbildung mit '...rrd_heizkreis1.rrd' bis '...rrd_heizkreis4.rrd' erzeugt).

Die Verzeichnis-Struktur ist wie folgt:

Persönlicher Ordner workspace HT3 software var databases Suchen				
Name	Größe	Typ	Änderungsdatum	
HT3_db.sqlite	2,5 MB	SQLite3-Datenbank	Fr 07 Feb 2014 09:45:47 CET	
HT3_db_rrd_heizgeraet.rrd	249,8 MB	Unbekannt	Fr 07 Feb 2014 09:44:48 CET	
HT3_db_rrd_heizkreis.rrd	204,4 MB	Unbekannt	Fr 07 Feb 2014 09:45:47 CET	
HT3_db_rrd_solar.rrd	181,7 MB	Unbekannt	Fr 07 Feb 2014 09:44:48 CET	
HT3_db_rrd_sysdatetime.rrd	45,4 MB	Unbekannt	Do 06 Feb 2014 18:18:43 CET	
HT3_db_rrd_warmwasser.rrd	90,8 MB	Unbekannt	Fr 07 Feb 2014 09:45:47 CET	junky-as

Abbildung 17: Datenbanken Verzeichnis

Die Größe der erzeugten Datei wächst bei der SQLite-Datenbank pro Tag um ca. 4 MByte an.

Nach einem Jahr ist mit ca. 1.5 GByte Daten zu rechnen. Eine Löschroutine ist realisiert. Dazu ist im Konfigurations-File: HT3_db_cfg.xml der Wert:

<autoerase_olddata>30</autoerase_olddata>
zu aktivieren.

Bei der rrdtool-Datenbank bleibt die Dateigröße fest bestehen und wird durch die gewählte Archiv-Speichertiefe und Anzahl der 'Logitems' bei der Erzeugung der Datenbank bestimmt.

Eine nachträgliche Veränderung der Datenbank-Strukturen bei vorhandenen Datenbanken ist schwierig und kann zu Datenverlusten führen. Deshalb sollte dies vermieden werden.

2.3.1.1 Datenbank 'SQLite'

Die Datenbank 'sqlite' wird erzeugt und genutzt, sobald im Konfigurationsfile der Parameter <sql-db><enable>on aktiviert ist (Details siehe: 2.2 Konfiguration). Eine schon vorhandene Datenbank wird nicht überschrieben.

Die SQLite-Datenbank wird für die Speicherung der dekodierten Datentelegramme benutzt.

Eine Auswertung der Daten wird in dieser Datenbank nicht gemacht.

Die folgenden Bilder zeigen die Tabellen und Spalten der SQL-Datenbank.
 (SQLite Plugin des Firefox)

The screenshot shows the SQLite Manager interface with the following details:

- File Edit View History Bookmarks Tools Help**
- SQLite Manager - /home/zs/workspace/HT3/software/var/databases/HT3_db.sqlite - Mozilla Firefox**
- Database Table Index View Trigger Tools Help**
- HT3_db.sqlite** is selected in the left sidebar.
- Structure** tab is active.
- heizgeraet** table is selected in the tree view.
- Local_date_time**, **UTC**, **T_vorlauf_soll**, **T_vorlauf_ist**, **T_ruecklauf**, **T_mischer**, **V_modus**, **V_brenner_motor**, **V_h**, and **V_h2** are the columns of the table.
- The table contains approximately 4193 rows of data.
- The last row (row 42) is highlighted in green, showing values: 2014.02.06 18:27:30, 1391707627, 37, 23.5, 23.2, 38.9, 1, 0, 1, and 1.
- Pagination at the bottom indicates "1 to 100 of 4193".

Abbildung 18: SQL-Datenbanktabelle 'heizgeraet'

Die Tabellen-Namen entsprechen dabei dem 'systempart'-Namen aus der Konfiguration.

Die Tabelle 'rddtool_infos' ist ab der Sw-Revision: 0.2.0 entfallen.

Die Daten für die rrdtool-Datenbank werden zyklisch alle 60 Sekunden von der Applikation automatisch eingetragen.

Die Spalten: 'Local_date_time' (lokale Zeit) und 'UTC' (UTC-Zeit) enthalten die Zeitstempel, in denen der Datenbank-Eintrag erfolgt ist. Diese Einträge sind in jeder Tabelle enthalten und werden automatisch beim SQL-'insert'-Aufruf erzeugt.

Der UTC-Zeitstempel wird für die interne Bestimmung von Zeitintervallen genutzt. Dabei ist dieser Wert von der Sommer-/Winterzeit-Umschaltung unabhängig und ist somit immer eine inkrementelle Referenz.

Zu beachten ist die korrekte Zeiteinstellung von CPU / Laptop, auf denen die Applikation läuft.

Bei CPU's ohne RTC (RealTimeClock) kann die Zeitsynchronisation durch den Anschluss an ein Gateway/Router (z.B. Fritzbox) erreicht werden.

Die Tabelle 'rddtool_infos' entfällt seit der SW-Version: 0.2.0.

Die Datenbank Einträge für das rrdtool werden jetzt zyklisch alle 60 Sekunden durch die Applikation gemacht.

2.3.1.2 Datenbank 'rrdtool'

Die Datenbank 'rrdtool' wird erzeugt, sobald im Konfigurationsfile der Parameter <rrdtool-db><enable>on aktiviert ist (Details siehe: 2.2 Konfiguration). Eine schon vorhandene Datenbank wird nicht überschrieben.

Für die Erzeugung der Datenbank und für die Daten-Aktualisierung werden perl-scripte dynamisch erzeugt und ausgeführt. Ebenso wird die Grafikerzeugung durch 'rrdgraph' mit einem perl-script realisiert.

Die dekodierten Daten werden in 60 Sekunden Intervallen von der Applikation: ht_collgate in die rrdtool-Datenbank geschrieben.

Die Applikationen: HT3_Analyser und HT3_Systemstatus schreiben nicht in die Datenbanken (separates Konfigurationsfile).

Es werden alle Werte 'logitems' der einzelnen Systemparts 'syspart' in die rrdtool-Datenbank übertragen mit Ausnahme von:

'sysdatetime' und 'hexdump'

Für jeden 'syspart' des Heizungssystems ('heizgeraet', 'heizkreis(n:=1...4)', 'warmwasser', 'solar' und 'sysdatetime') gibt es ein eigenes 'rrdtool'-Datenbankfile mit dem Datei-Suffix: .rrd (siehe Bild).

Persönlicher Ordner					workspace	HT3	software	var	databases	Suchen
Name	Größe	Typ	Änderungsdatum							
HT3_db.sqlite	2,5 MB	SQLite3-Datenbank	Fr 07 Feb 2014 09:45:47 CET							
HT3_db_rrd_heizgeraet.rrd	249,8 MB	Unbekannt	Fr 07 Feb 2014 09:44:48 CET							
HT3_db_rrd_heizkreis.rrd	204,4 MB	Unbekannt	Fr 07 Feb 2014 09:45:47 CET							
HT3_db_rrd_solar.rrd	181,7 MB	Unbekannt	Fr 07 Feb 2014 09:44:48 CET							
HT3_db_rrd_sysdatetime.rrd	45,4 MB	Unbekannt	Do 06 Feb 2014 18:18:43 CET							
HT3_db_rrd_warmwasser.rrd	90,8 MB	Unbekannt	Fr 07 Feb 2014 09:45:47 CET							

Einträge in die rrdtool-db dürfen für ein 'Logitem' nur einmal für einen bestimmten Zeitstempel gemacht werden. Eine Wiederholung mit gleichem Zeitstempel führt zu einer Fehlermeldung der Datenbank. Die Applikation sorgt für die richtige zeitliche Reihenfolge.

Es muss jedoch beachten werden, dass nicht mehrere Applikationen mit aktiviertem Datenbankzugriff betrieben werden da diese dann parallel die gleichen Daten in die gleiche Datenbank schreiben wollen. Dies kann mit unterschiedlichen Konfigurationsfiles (andere Datenbank oder Datenbankzugriff deaktiviert) vermieden werden.

Siehe [2]

2.3.2 'MQTT' - Client

Diese Schnittstelle nutzt die Eigenschaften des MQTT - Protokolls (Message Queue Telemetry Transport).

Es werden die dekodierten Daten zu einem MQTT - Broker gesendet der separat installiert werden muss, siehe Installation.

Die einzelnen dekodierten Daten werden gemäß MQTT in der Form:
topic, payload

übertragen. Der MQTT-Client macht in diesem Fall ein: Publish.

Es werden nur Payload - Datenänderungen übertragen um die erforderliche Kommunikation gering zu halten. Diese Eigenschaft kann im Konfigurationsfile deaktiviert werden mit:

```
<Publish_OnlyNewValues>False</Publish_OnlyNewValues>
```

Die Topic-Name werden aus dem Topic-Rootnamen des Konfigurationsfile: mqtt_client_cfg.xml und dem <accessname> des Konfigurationsfile: HT3_db_cfg.xml gebildet.

Beispiel für einen **Publish** Topic-Namen:

hometop/ht/ch_Treturn

Beide Teil-Namen (Topic-rootname und accessname) können frei festgelegt werden.

Zusätzlich zum Publish (senden) von dekodierten Heizungsdaten registriert (Subscribe) sich der MQTT-Client auch beim MQTT - Broker.

Mit dieser Eigenschaft können Kommandos eines anderen MQTT-Clients (über den Broker) empfangen und ausgewertet werden. Diese Kommunikation erfolgt ebenfalls in der Struktur: topic, payload.

Das nötige Subscribe-Topic unterscheidet sich jedoch vom Publish-Topic. Es wird vor dem Topic noch ein 'set/' String erwartet.

Beispiel für einen **Subscribe** Topic-Namen:

set/hometop/ht/hc1_Tdesired

Die MQTT - Client Schnittstelle ist mit mehreren Klassen im Modul: 'mqtt_client_if.py' realisiert. Detaillierte Beschreibung der Eigenschaften sind in diesem Modul zu finden.

Das Konfigurationsfile: mqtt_client_cfg.xml legt die Eigenschaften der Schnittstelle fest.

Per default ist 'localhost' für den MQTT - Broker eingetragen. Diese muss angepasst werden wenn der Broker auf einer anderen Hardware installiert ist.

Inhalt des mqtt_client_cfg.xml Konfigurationsfiles:

```
<mqtt_client_cfg>
<item>mqtt_client_configuration</item>

<logging>
  <path>./var/log</path>
  <filename>mqtt_client.log</filename>
  <loglevel>INFO</loglevel>  <!-- loglevel-types are (details see library: logging.py):
    DEBUG  := Debug-Level
    INFO   := Information-Level
    WARNING := Warning-Level
    ERROR   := Error-Level
    CRITICAL := Critical-Level - >
</logging>

<!-- broker-server configuration-values - >
<mqtt_broker>
  <serveraddress>localhost</serveraddress>
  <portnumber>1883</portnumber>  <!-- Default portnumber for MQTT - >
  <user></user>
  <password></password>
</mqtt_broker>

<!-- mqtt-client configuration-values - >
<mqtt_client>
  <topic_root_name>hometop/ht</topic_root_name>
  <QoS>1</QoS>          <!-- Quality of Service: 0,1 or 2 - >
  <CleanSession>False</CleanSession>  <!-- True / False - >
  <Retain>True</Retain>        <!-- True / False - >
  <LastWillRetain>True</LastWillRetain>  <!-- True / False - >
  <Publish_OnlyNewValues>True</Publish_OnlyNewValues> <!-- True / False - >
  <client_id>myHeater_MQTT_Client</client_id> <!-- unique ID for your purposes - >
</mqtt_client>
</mqtt_client_cfg>
```

MQTT Details siehe auch [11]

Um die Kommunikation mit dem Broker und der Applikation testen zu können kann man die MQTT-Client Tools 'mosquitto_sub' und 'mosquitto_pub' verwenden.

1. Subscribing auf alle dekodierten Daten die zum Broker gesendet wurden:

```
mosquitto_sub -d -h <host IP-Adr.> -t hometop/ht/# -q 1
```

2. Publishing von Steuerdaten zu Steuerung der Heizung (nur mit ht_transceiver möglich):

2.1 Setzen des aktuellen Temperatur-Niveaus auf „heizen“ mit:

```
mosquitto_pub -d -h <host IP-Adr.> -t set/hometop/ht/hc1_Tniveau -m "heizen"
```

2.2 Setzen der Soll-Temperatur auf 21.5 Grad des Niveaus „heizen“ mit:

```
mosquitto_pub -d -h <host IP-Adr.> -t set/hometop/ht/hc1_Tdesired -m "21.5,heizen"
```

Die möglichen 'Set-Parameter' sind in der Tabelle: Tabelle 9: Set-Parameter je nach Regler-Typ beschrieben.

2.3.3 'SPS' – ht_Server

Der SPS ht_server (Speicher Programmierbare Steuerung) stellt die dekodierten Daten einem anfragenden SPS-Client zur Verfügung.
Die Abfrage auf Heizungsdaten kann mit den <**accessname**> aus dem Konfigurationsfile: HT3_db_cfg.xml erfolgen.

Beispiel:

```
SPS client send      -> :ch_Tflow_measured  
SPS server response <- :ch_Tflow_measured=43.9;
```

Ebenso ist eine Abfrage in Kurzform möglich. Diese Befehls-Kurzform wird nach dem Start des SPS ht_servers generiert und sieht wie folgt aus:

```
sps_cmd ; nickname ; accessname  
A00    ;HG        ;ch_Tflow_desired  
A01    ;HG        ;ch_Tflow_measured  
A02    ;HG        ;ch_Treturn  
A03    ;HG        ;ch_T3waymixer  
A04    ;HG        ;ch_mode  
A05    ;HG        ;ch_burner_fan  
A06    ;HG        ;ch_burner_operation  
....  
B00    ;HK1       ;hc1_Tdesired  
B01    ;HK1       ;hc1_Tmeasured  
B02    ;HK1       ;hc1_Troom  
....
```

Für jeden Systemanteil (nickname) wird ein neuer Buchstabe in aufsteigender Reihenfolge beginnend mit 'A' gewählt. Es wird ein Mapping zwischen den 'acccsesname's und den SPS Kommandos gebildet.

Eine Besonderheit bei der Namesgebung sind dabei die Special-Commands:

sps_cmd ; nickname ; accessname	Funktion
S00 ;special ;hostname	Hostname des Target.
S01 ;special ;os_sys	Betr.System des Target.
S09 ;special ;map_dump	Kommando-Map als CSV-File.

Das Kommando-Mapping wird als CSV-File erzeugt sobald man den Kurzbefehl: **S09** sendet.

Das CSV-File wird im Verzeichnis: ~/HT3/sw/var/log im Target-Rechner erzeugt.

Beispiel für eine Abfrage mit Kurzform:

```
SPS client send      -> :A01  
SPS server response <- :A01=43.9;
```

Die SPS – ht_Server Schnittstelle ist mehreren Klassen im Modul: 'SPS_if.py' realisiert.

Das Konfigurationsfile: SPS_cfg.xml legt die Eigenschaften der Schnittstelle fest.

Wenn der SPS - Client nicht dort installiert ist wo der ht_colgate als SPS - Server läuft, so ist <serveraddress>**IP-Adresse ht_colgate - server**</serveraddress> des SPS-Client anzupassen.

Gleiches gilt für die Port-Nummer falls diese anderweitig schon vergeben wurde.

Inhalt des SPS-Konfigurationsfiles:

```
</SPS_cfg>
<item>SPS_configuration</item>
<!-- sps configuration-values →
<SPS_server>
  <serveraddress></serveraddress>
  <portnumber>10001</portnumber>
</SPS_server>
<!-- sps-client configuration-values →
<SPS_client>
  <serveraddress>localhost</serveraddress>
  <portnumber>10001</portnumber>
</SPS_client>
</SPS_cfg>
```

2.3.4 'HomeAssistant'-IF (mqtt client-daemon)

Die HomeAssistant Software (hier kurz: HA genannt) nutzt MQTT Telegramme für die Erfassung von Daten.

Die nötigen Strukturen für MQTT- „topic“ und - „payload“ unterscheiden sich jedoch von den im diesem Projekt per Default eingestellten Ausgaben.

Zusätzlich kann die HA-Software von den Datenerfassungs-Sensoren automatisch Konfigurationsdaten übernehmen.

Die Anpassung der default MQTT-Datenausgaben wird durch ein zusätzliches Software-Modul erreicht, welche als „daemon“ im Hintergrund betrieben und beim RPi-Boot gestartet wird.
Folgende zusätzlichen Software-Module werden benutzt:

Modul	Funktion
~/HT3/sw/ht_2hassio.py	Mqtt-daemon, generiert angepasste mqtt-topic und -payload (json.format) für die Homeassistant Software.
~/HT3/sw/systemconfig/ht_2hass	Startscript für ht_2hassio.py

Der daemon nutzt das für dieses Projekt per default eingestellte Konfigurationsfile:

~/HT3/sw/etc/config/mqtt_client_cfg.xml

und den darin enthaltene „topic“-Namen:

```
<!-- mqtt-client configuration-values -->
<mqtt_client>
    <topic_root_name>hometop/ht</topic_root_name>
```

Der daemon „subscribed“ mit diesem topic-Namen und erhält die zugehörigen Heizungs-Daten vom MQTT-Broker.

Diese Informationen werden für den HA angepasst und als neue „topic“- / „payload“-Daten per „publish“ wieder zum MQTT-Broker übertragen.

Die HA-Software erhält dann die aufbereiteten Daten vom MQTT-Broker.

Alle Daten erhalten zur Unterscheidung von anderen MQTT-Daten einen Kenner für die HA-Software:

'**homeassistant/sensor/heatersystem/state**'
→ aktuelle Sensor-Stati (als json.stream in der payload).

'**homeassistant/sensor/heatersystem/<sensor-name>/config**'
→ Konfigurationsinformation des Sensors.

Beispiel für State-Daten:

```
Topic: "homeassistant/sensor/heatersystem/state"
Payload: "{{ value_json.dhw_generating }}",
"name": "dhw_generating",
"uniq_id": "dhw_generating",
"unit_of_measurement": ""
}
```

Damit der ht_2hassio.py-daemon erhält, muss die MQTT-Schnittstelle im ht_collgate Konfigurations-File aktiviert sein.

Bei der ersten Inbetriebnahme müssen alle MQTT-Daten für die korrekte automatische HA-Konfiguration ausgegeben werden.

Dazu ist das Flag: Publish_OnlyNewValues auf 'False' zu setzen.

Konfigurationsfile:

~/HT3/sw/etc/config/mqtt_client_cfg.xml

1. Erster Start und Initialisierung

Wert ändern auf 'False'

<Publish_OnlyNewValues>**False**</Publish_OnlyNewValues>

Danach ist der daemon: ht_collgate.py neu zu starten.

2. Abgeschlossene automatische Konfiguration

Nach erfolgreicher automatischer Konfigurations-Datenübernahme in der HA-Software kann das Flag wieder auf 'True' gesetzt werden.

<Publish_OnlyNewValues>**True**</Publish_OnlyNewValues>

Danach ist der daemon: ht_collgate.py neu zu starten.

Damit der daemon beim Booten gestartet wird, ist das Start-Script: ht_2hass zu aktivieren. Details sind im Kapitel: Installation zu finden.

Wie die erfassten Heizungsdaten im HomeAssistant Dashboard anzeigen werden können zeigt folgendes Beispiel:

HAus			
Heizgerät	Heizkreis 1	Heizung_Solar	Heizung_WW
ch_Tflow_measured 25,7 °C	hc1_Tmeasured 21,8 °C	sol_Tcollector 21 °C	dhw_Tcylinder 41,9 °C
ch_Treturn 25,3 °C	hc1_Tflow_desired 38 °C	sol_Tcylinder_bottom 29,3 °C	dhw_Tmeasured 43,8 °C
ch_T3waymixer 43,8 °C	hc1_Tdesired 21,5 °C	sol_yieldd_last_hour 0	dhw_Tok 0
ch_Toutside 0,0 °C	hc1_mixerposition 0 %	sol_yield_sum 0,0	dhw_Tdesired 49 °C
ch_Tflow_desired 38 °C	hc1_Tflow_mixer 0,0 °C	sol_pump 0	dhw_runtime_ch 1024.8666666666666666
ch_burner_power 0 %	hc1_Thiveau 3	sol_runtime 0	dhw_starts_ch 5255
ch_pump_heating.power 40 %	hc1_pump 1	sol_yieldd_last_day 0,0	dhw_charge_once 0
ch_Thdrylic_switch 0 °C	hc1_operationstatus 3	sol_buffer_full 0	dhw_thermal_desinfection 0
ch_mode 1	Binärsensor	sol_collector_deactive 0	dhw_generating 0
ch_burner_fan 0	Updater	Person	dhw_boost_charge 0
ch_burner_operation 0	Aus	bertha Unbekannt	
ch_pump_heating 1	Sonne	Regnerisch HAus 5,3 °C G> 2,3 mm	
ch_pump_circulation 0	Sun Über dem Horizont		
ch_runtime_tot 10949.2833333333333333			

Homeassistant Konfigurations-Beispiel:

```
- title: Heizung und Solar
  path: heizung-und-solar
  badges: []
  cards:
    - entities:
        - entity: sensor.ch_toutside
        - entity: sensor.ch_tflow_desired
        - entity: sensor.ch_tflow_measured
        - entity: sensor.ch_treturn
        - entity: sensor.ch_t3waymixer
        - entity: sensor.ch_pump_heating
        - entity: sensor.ch_burner_power
        - entity: sensor.ch_pump_heating_power
        - entity: sensor.ch_mode
        - entity: sensor.ch_burner_fan
        - entity: sensor.ch_burner_operation
        - entity: sensor.ch_thdrylic_switch
        - entity: sensor.ch_causecode
        - entity: sensor.ch_errorcode
        - entity: sensor.ch_pump_circulation
        - entity: sensor.ch_pump_cylinder
        - entity: sensor.ch_runtime_ch
        - entity: sensor.ch_runtime_tot
        - entity: sensor.ch_starts_ch
        - entity: sensor.ch_runtime_tot
      title: Heizgerät
      type: entities
    entities:
      - entity: sensor.hc1_tflow_desired
      - entity: sensor.hc1_tdesired
      - entity: sensor.hc1_tmeasured
      - entity: sensor.hc1_tniveau
      - entity: sensor.hc1_operationstatus
      - entity: sensor.hc1_tflow_mixer
      - entity: sensor.hc1.mixerposition
      - entity: sensor.hc1_pump
      title: Heizkreis
    - type: entities
      entities:
        - entity: sensor.dhw_tdesired
        - entity: sensor.dhw_tcylinder
        - entity: sensor.dhw_tmeasured
        - entity: sensor.dhw_tok
        - entity: sensor.dhw_runtime_ch
        - entity: sensor.dhw_starts_ch
        - entity: sensor.dhw_charge_once
        - entity: sensor.dhw_thermal_desinfection
        - entity: sensor.dhw_generating
        - entity: sensor.dhw_boost_charge
      title: Warmwasser
    - type: entities
      entities:
        - entity: sensor.sol_tcollector
        - entity: sensor.sol_tcylinder_bottom
        - entity: sensor.sol_pump
        - entity: sensor.sol_runtime
        - entity: sensor.sol_yield_last_hour
        - entity: sensor.sol_yield_sum
        - entity: sensor.sol_yield_last_day
        - entity: sensor.sol_buffer_full
        - entity: sensor.sol_collector_deactive
      title: Solar
```

2.4 Applikationen

Alle realisierten Applikationen nutzen die Heaterbus Protokoll-Daten und tragen diese in die SQLite- und rrdtool-Datenbank ein bzw. senden diese zum MQTT-Broker (wenn die Konfiguration dafür aktiviert ist).

Unterschiede sind nur bei der grafischen Ausgabe vorhanden. Details im folgenden.

Hinweis:

Die Applikationen: HT3_Systemstatus.py, HT3_Analyser.py haben ein eigenes Konfigurationsfile, in dem der Datenbank-Zugriff (sqlite und rrdtool) deaktiviert ist.

Damit wird ein Mehrfacheintrag der zeitgleichen ‚Logitems‘ in die Datenbanken vermieden.

2.4.1 HT3-Analyser

Der HT3 Bus Analyser dient zur Analyse der empfangenen Datenprotokolle. Es werden die Daten grafisch in Plain-Text als Systemstatusanzeige und als hexadezimaler Protokollstring im Hexdump-Fenster angezeigt.

Die hexadezimalen Ausgaben beginnen mit der Message-Id 'MsgID' des Telegramms gefolgt vom 'Nickname' der Systemkomponenten (HG, HK1, HK2, HK3, HK4, WW, SO, DT).

Die Zeilen sind je nach 'Nickname' farblich unterschieden.

Die einzelnen Systemkomponenten (Heizgeraet, Heizkreis(e), Warmwasser und Solar) können separat ausgewählt werden. Es werden dann nur noch die Protokolle der jeweiligen Systemkomponente angezeigt. Das rechte Statusfenster zeigt in diesem Fall die aktuellen Daten der Systemkomponente an.

Eine Beschreibung der Protokolle kann über den Auswahlknopf 'Info' erreicht werden. Die Darstellung wird als Html-File mit dem auf dem System vorhandenen Browser gemacht.

Nach Auswahl 'System' werden alle Systemkomponenten gleichzeitig angezeigt.

Der Auswahlknopf 'Hexdump clear' löscht den Inhalt des Hexdump-Fenster, der Auswahlknopf 'Ende' beendet die Applikation.

Die folgende Abbildung zeigt ein System mit einem Heizkreis.

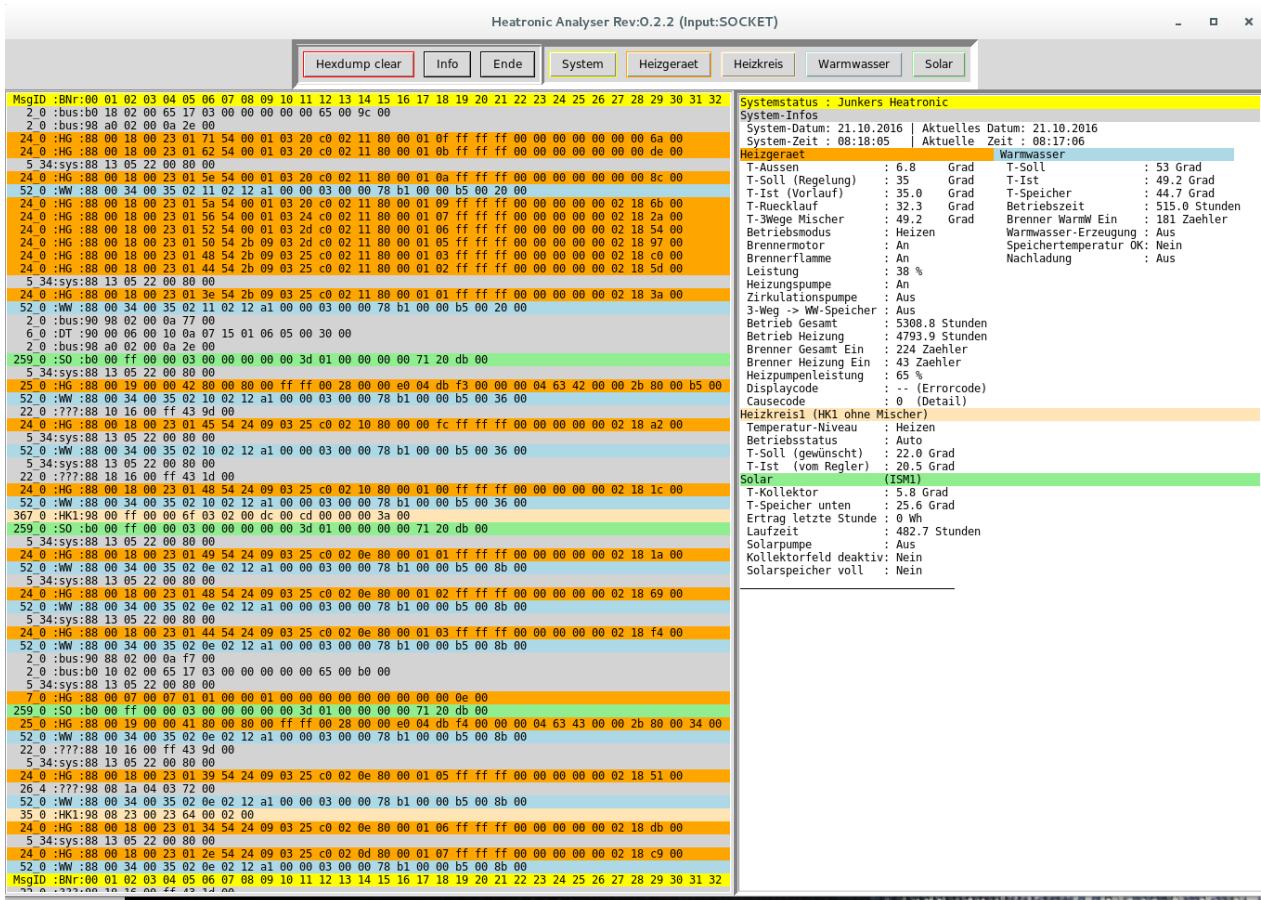


Abbildung 19: HT3 Analyser GUI

Wie die HT3-Analyser-GUI mit 3 Heizkreisen aussehen kann, zeigt das folgende Bild:

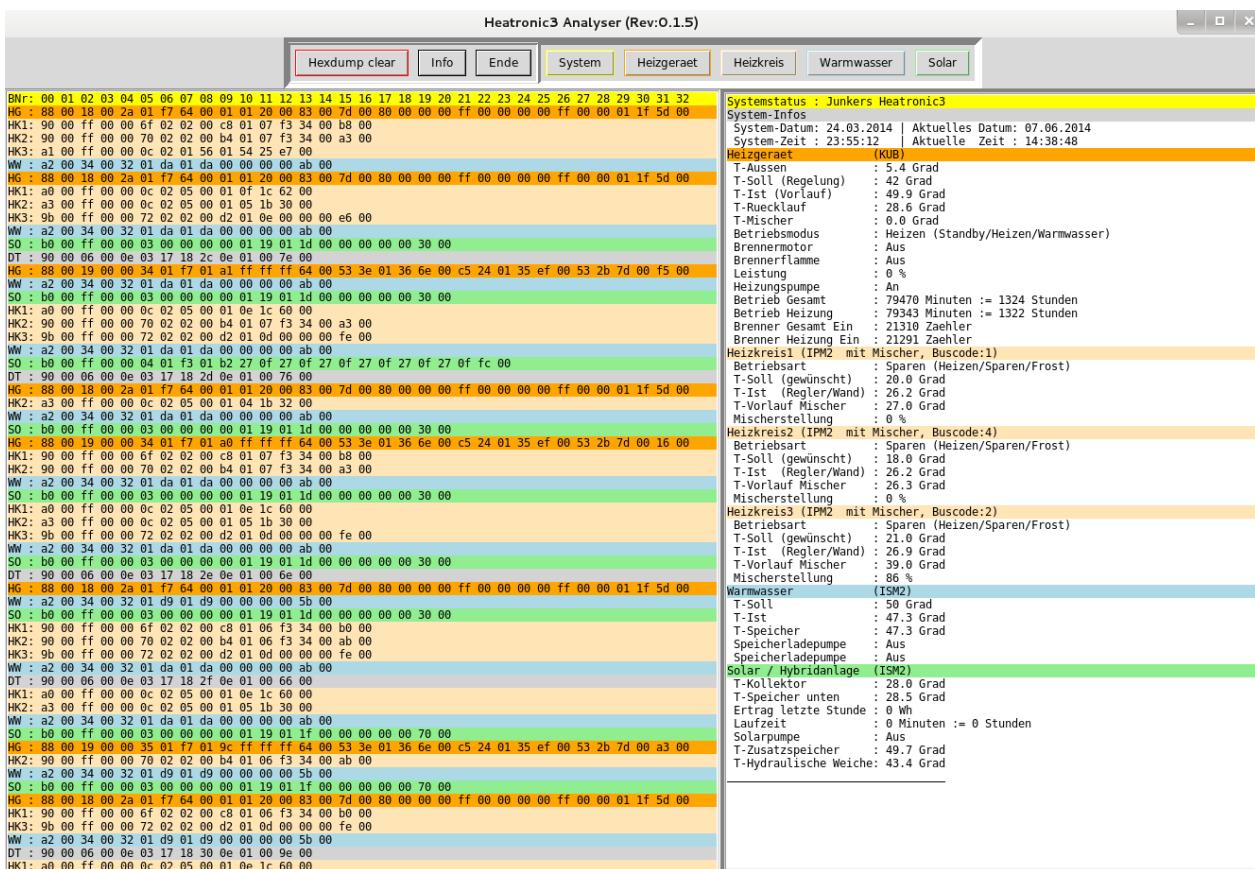


Abbildung 20: HT3 Analyser GUI (System mit 3 Heizkreisen)

2.4.2 HT3_Systemstatus

Der HT3 Systemstatus zeigt als Übersicht den aktuellen Heizungssystemstatus grafisch an. Die einzelnen Systemkomponenten (Heizgeräet, Heizkreis, Warmwasser und Solar) können wie beim HT3_Analyser separat ausgewählt werden.

Heatronic Systemstatus Rev:0.2.2 (Input:SOCKET)			
System	Heizgeräet	Heizkreis	Warmwasser
Solar	Ende		
Systemstatus : Junkers Heatronic			
System-Infos			
System-Datum:	21.10.2016	Aktuelles Datum:	21.10.2016
System-Zeit :	08:36:05	Aktuelle Zeit :	08:35:35
Heizgeräet		Warmwasser	
T-Aussen	: 6.8 Grad	T-Soll	: 53 Grad
T-Soll (Regelung)	: 35 Grad	T-Ist	: 48.0 Grad
T-Ist (Vorlauf)	: 35.0 Grad	T-Speicher	: 44.5 Grad
T-Rücklauf	: 32.6 Grad	Betriebszeit	: 515.0 Stunden
T-3Wege Mischer	: 48.0 Grad	Brenner WarmW Ein	: 181 Zähler
Betriebsmodus	: Heizen	Warmwasser-Erzeugung	: Aus
Brennermotor	: An	Speichertemperatur OK:	Nein
Brennerflamme	: An	Nachladung	: Aus
Leistung	: 37 %		
Heizungspumpe	: An		
Zirkulationspumpe	: Aus		
3-Weg -> WW-Speicher	: Aus		
Betrieb Gesamt	: 5309.1 Stunden		
Betrieb Heizung	: 4794.1 Stunden		
Brenner Gesamt Ein	: 224 Zähler		
Brenner Heizung Ein	: 43 Zähler		
Heizpumpenleistung	: 65 %		
Displaycode	: -- (Errorcode)		
Causecode	: 0 (Detail)		
Heizkreis1 (HK1 ohne Mischer)			
Temperatur-Niveau	: Heizen		
Betriebsstatus	: Auto		
T-Soll (gewünscht)	: 22.0 Grad		
T-Ist (vom Regler)	: 20.5 Grad		
Solar (ISM1)			
T-Kollektor	: 6.6 Grad		
T-Speicher unten	: 25.6 Grad		
Ertrag letzte Stunde	: 0 Wh		
Laufzeit	: 482.7 Stunden		
Solarpumpe	: Aus		
Kollektorfeld deaktiv:	Nein		
Solarspeicher voll	: Nein		

2.4.3 ht_collgate

Der ht_collgate verbindet sich mit dem ht_proxy.server und erhält über diesen die RAW-Daten der Heizung. Diese RAW-Daten werden dekodiert und im RAM für Abfragen bereitgestellt.

Der ht_collgate wird als daemon betrieben und nutzt je nach Konfiguration die Schnittstellen zu:

1. sqlite – Datenbank,
2. rrdtool – Datenbank,
3. MQTT – Client Schnittstelle,
4. SPS – Server Schnittstelle.

Zu 1. & 2.

Die Daten werden in die Datenbanken sqlite und rrdtool geschrieben. Eine grafische Ausgabe der Daten wird nur mit dem 'rrdtool' gemacht.

Die Schnittstellen **MQTT** und **SPS** sind per default ausgeschaltet, können jedoch im Konfigurations-File 'collgate_cfg.xml' aktiviert werden.

Der Daemon: ht_collgate ist dann erneut zu starten.

Zu 3.

Die MQTT Schnittstelle sendet die dekodierten Heizungsdaten zu einem MQTT - Broker (publish) und kann ebenso Heizungs-Kommandos von anderen MQTT - Clients empfangen (gesendet über den Broker und subscribe). Das 'publish' erfolgt immer dann wenn neue Daten für einen Heizungs-Systemanteil vorhanden sind.

Weitere Details siehe: 'MQTT' - Client

Zu 4.

Die SPS Schnittstelle stellt anfragenden SPS-Clients die dekodierten Heizungsdaten zur Verfügung. Der Zugriff durch den SPS-Client auf die Heizungsdaten erfolgt mit den 'accessname's aus dem Konfigurations-File oder ebenso durch Kurzkommandos. Eine Steuerung der Heizung mit dieser Schnittstelle ist z. Zeit nicht realisiert.

Die dekodierten Daten für die SPS-Clients stammen aus dem RAM des ht_collgate und sind jederzeit (ca. 2 Minuten nach dem ht_collgate Start) verfügbar.

Weitere Details siehe: 'SPS' - ht_Server

Inhalt des collgate_cfg.xml Konfigurationsfiles:

```
<collgate_cfg>
<item>collgate_configuration</item>
<interfaces>
  <ht_if>
    <!-- this IF is forced to be on -->
    <cfg_file>./etc/config/HT3_db_cfg.xml</cfg_file>
  </ht_if>
  <MQTT_client_if>
    <enable>Off</enable>
    <cfg_file>./etc/config/mqtt_client_cfg.xml</cfg_file>
  </MQTT_client_if>
  <SPS_if>
    <enable>Off</enable>
    <cfg_file>./etc/config/SPS_cfg.xml</cfg_file>
  </SPS_if>
</interfaces>
</collgate_cfg>
```

Die MQTT- und SPS-Schnittstellen sind per default ausgeschaltet.
Bei Bedarf diese mit '**On**' aktivieren.

Die HT_IF Schnittstelle kann nicht deaktiviert werden da diese für den Empfang und die Dekodierung der Daten nötig ist.

Innerhalb des Konfigurations-Files: HT3_db_cfg.xml können jedoch die jeweiligen Datenbanken sqlite-db und rrdtool-db aktiviert bzw. deaktiviert werden.

2.4.4 ht_2hassio

Das Modul: ,ht_2hassio.py‘ ist die Schnittstelle zwischen den MQTT-Ausgaben von diesem Projekt und der HomeAssistant Applikation (Hassio).

Voraussetzung für eine korrekte Funktion ist die Aktivierung der MQTT-Schnittstelle mit den Ausgaben der erfassten Heizungsdaten.

Dazu muss im ht_colgate Konfigurations-File die MQTT-Schnittstelle auf ,ON‘ eingestellt werden, siehe ht_colgate.

Das Modul ,ht_2hassio.py‘ läuft als ,daemon‘, gestartet beim Booten des RPi und verwendet das für MQTT vorgesehene Konfigurationsfile:

`~/HT3/sw/etc/config/mqtt_client_cfg.xml`

Der daemon nutzt u.A. den darin enthaltenen ,topic‘-Namen um sich mit dem MQTT-Broker zu verbinden (Subscribe).

`<topic_root_name>hometop/ht</topic_root_name>`

Es wird für die Verbindung zum MQTT-Broker eine eindeutige Kennung
`<client_id>` verwendet:

`„hometop2HA_if“`

Die empfangenen Daten (topic, payload) werden aufbereitet und als neuer topic/payload-Stream im JSON-Format an den MQTT-Broker für die HomeAssistant-Applikation gesendet (Publish).

Zur Unterscheidung zu anderen MQTT-Datenströmen wird folgender Prefix verwendet:

`"homeassistant/sensor/heatersystem/state" bzw.`

`„homeassistant/sensor/heatersystem/<sensor-name>/config“`

Ist keine MQTT-Ausgabe/-Schnittstelle aktiviert und kein MQTT-Broker vorhanden, werden Fehlermeldungen generiert.

Logfile:

`„~/HT3/sw/var/log/hometop2HA_if.log“`

Falls diese Schnittstelle nicht benötigt wird, sollte das Startscript des ht_2hassio.py nicht aktiviert werden.

2.5 Comport <=> Socket proxy (Server & Client)

Der 'comport <=> socket' - proxy ist ein Software-Anteil zwischen der Erfassungs-Hardware und der Telegramm-Auswertung / Steuerung.

Die proxy-Software besteht aus den Anteilen:

1. proxy-server und
2. proxy-client(s).

Dies erlaubt den Zugriff auf die Adapter-Hardware über Socket-Verbindungen und Telegramme.

Der proxy-server stellt für N Socket-Clients den Zugriff auf die Hardware zur Verfügung. Die maximale Anzahl der Clients (N) ist nur durch die verwendete CPU-Hardware (RaspberryPi) begrenzt. N <= 3 Clients ist ein getester Standardwert.

2.5.1 ht_proxy (proxy-server)

Der proxy-server 'ht_proxy' stellt die Verbindung zwischen dem Comport: /dev/ttyAMA0 bzw. /dev/serial0 und der TCP/IP (Socket-Verbindung) her.

Da der 'ht_proxy' direkt mit der comport-Schnittstelle verbunden ist, muss der daemon auch auf der Hardware installiert sein, die diese Schnittstelle bereitstellt. Dies ist in der Regel der RaspberryPi.

Die Installation wird im Kapitel: Installation beschrieben.

Folgende Software-Anteile sind Bestandteil des proxy-servers:

Software-Anteile	Funktion	Bemerkung
./HT3/sw/ht_proxy.py	Proxy-Server. Wird gestartet durch Aufruf Klasse: <i>cht_proxy_daemon</i>	Übergabe des Konfigurations-Files an die aufgerufene Klasse.
./HT3/sw/lib/ht_proxy_if.py	Library-Klasse: <i>cht_proxy_daemon</i> und zugehörige Methoden.	Proxy-server daemon der library
./HT3/sw/etc/config/ht_proxy_cfg.xml	Konfigurations-File für proxy-server und proxy-client.	Konfig-File für Server und Client.
./HT3/sw/etc/sysconfig/ht_proxy	Init-Script für das Starten des proxy-servers.	Script muss mit 'systemctl' in /etc/init.d installiert werden.
./HT3/sw/var/log/ht_proxy.log	Logfile des proxy-servers	Verzeichnis wird angelegt, falls nicht vorhanden. Der Name stammt aus dem Konfigurations-File.

Der proxy-server wird automatisch in den zugehörigen Run-Level gestartet. Dies wird durch ein Init-Script erreicht, welches auch für die korrekte Start-Reihenfolge sorgt (Start des proxy-servers vor 'HT'-Clients).

Die Nutzung des proxy-servers ist unabhängig von der verwendeten Erfassungs-Hardware. Er kann mit 'HT3-miniaadapter', 'HT3-microadapter' aber auch mit einem der 'ht_transceiver' betrieben werden.

Einzig die erforderliche Baudrate und der Anschlussport muss korrekt eingestellt sein. Dies erfolgt im Konfigurationsfile: ht_proxy_cfg.xml

Adapter-Name	Schnittstelle zum proxy-server	Baudrate
HT3-MiniAdapter	/dev/ttyAMA0 (RaspberryPi) oder /dev/serial0	9600
HT3-MicroAdatper	/dev/ttyUSBx (PC / Laptop etc.)	9600
ht_transceiver (ht_piduino & ht_pitiny)	/dev/ttyAMA0 (RaspberryPi) oder /dev/serial0	19200

Durch den proxy-server werden die von der Adapter-Hardware erfassten Heizungsbus-Signale an jeden verbundenen proxy-client gesendet. Ebenso werden Befehle von jedem verbundenen proxy-client an den 'ht_transceiver' weitergeleitet. Gleichzeitige Kommandos von mehreren Clients an den 'ht_transceiver'-Adapter müssen jedoch vermieden werden. Eine Kollisions-Erkennung bzw. Vermeidung ist nicht realisiert.

Der ht_proxy schreibt Informationen und Debug-Ausgaben in ein zugehöriges Log-File.

Das Log-File und das konfigurierte Verzeichnis werden automatisch angelegt, falls diese nicht vorhanden sind.

2.5.2 ht_client_example (proxy-client Beispiel)

Der proxy-client 'ht_client_example' ist ein Beispiel für einen proxy-client.

Die Klasse: *cht_socket_client* des Moduls: *ht_proxy_if* erhält das Konfiguration-File als Übergabe-Parameter.

Die für den Client relevanten Informationen werden aus dem Client-Anteil des Konfiguration-Files entnommen.

In diesem Client-Beispiel wird die client.run()-Methode aufgerufen, welche die erfassten Daten als Byte-Hexwerte anzeigt.

Folgende Software-Anteile sind Bestandteil eines proxy-clients:

Software-Anteile	Funktion	Bemerkung
./HT3/sw/ht_client_example.py	Aufruf und Start der Klasse: <i>cht_socket_client</i>	Übergabe des Konfigurations-Files an die aufgerufene Klasse.
./HT3/sw/lib/ht_proxy_if.py	Klasse: <i>cht_socket_client</i> .	Proxy-client der library
./HT3/sw/etc/config/ht_proxy_cfg.xml	Konfigurations-File für proxy-server und proxy-client.	Konfig-File für Server und Client.
./HT3/sw/var/log/ht_client_modem.log bzw. ./HT3/sw/var/log/ht_client_rx.log	Logfile des proxy-clients für devicetype:MODEM bzw. RX	Verzeichnis und Namen stammen aus dem Konfigurations-File.

2.5.3 ht_netclient (Heizungssteuer-Client)

Der 'ht_netclient.py' dient zur Steuerung der Heizungsanlage mit dem 'ht_transceiver'-Adapter.

Dazu verbindet sich der 'ht_netclient' mit dem 'ht_proxy' und sendet die erforderlichen Befehle.

Danach trennt der Client die Verbindung wieder und beendet sich.

Eine Steuerung der Heizungsanlage mit den anderen Adapter-Typen ist nicht möglich.

Folgende Software-Anteile sind Bestandteil des ht_netclient:

Software-Anteile	Funktion	Bemerkung
./HT3/sw/ht_netclient.py	Aufruf und Start der Klasse: <i>cht_socket_client</i>	Übergabe des Konfigurations-Files an die aufgerufene Klasse.
./HT3/sw/lib/ht_proxy_if.py	Klasse: <i>cht_socket_client</i> .	Proxy-client der library
./HT3/sw/etc/config/ht_proxy_cfg.xml	Konfigurations-File für proxy-server und proxy-client.	Konfig-File für Server und Client.
./HT3/sw/var/log/ht_client_modem.log bzw. ./HT3/sw/var/log/ht_client_rx.log	Logfile des proxy-clients für devicetype:MODEM bzw. RX	Verzeichnis und Namen stammen aus dem Konfigurations-File.
./HT3/sw/lib/ht_transceiver.py	Library mit Methoden für 'ht_transceiver' - Befehle	Transceiver-Konfiguration und Reset damit möglich.
./HT3/sw/lib/ht_yanetcom.py	Library für NetCom ähnliche Befehle.	Einstellung der Heizung-Temperaturwerte und Betriebsarten ist damit möglich.

2.5.3.1 ht_netclient Steuerbefehle

Die im folgenden beschriebenen Steuerbefehle sind für die Heizungsregler-Typen: Fxyz und Cxyz unterschiedlich. Direkt nach dem Aufruf des Programms wird der Regler-Typ abgefragt.

Folgende Steuer-Befehle sind für die Fxyz Regler (z.B.: FW100) realisiert:

Befehl und Parameter	Parameter	Funktion	Bemerkung
ht_netclient.py -t xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Heizen (default)	Temperatur-Niveau bleibt auch nach einem Betriebsstatus-Wechsel erhalten.
ht_netclient.py -tmod WERT	WERTe: heizen sparen frost	Festlegung des Betriebsstatus für das ausgewählte Temperatur-Niveau.	Betriebsstatus für das jeweilige Temperatur-Niveau. (Wird zusammen mit: -t xy.z verwendet).
ht_netclient.py -b WERT	WERTe: auto heizen sparen frost	Es wird der Betriebs-Status der Heizung fest auf den angegebenen Parameter eingestellt. Der Wert des zugehörigen Temperatur-Niveaus verändert sich nicht.	Der jeweilige Betriebs-Status wird eingestellt und als NetCom (NC) Information am Bedienteil (Fxyz) angezeigt. Siehe Bilder.
ht_netclient.py -hc CIRCUIT	CIRCUIT: 1 ... 4	Heizkreis Nummer. Default: 1	Heizkreis-Nummer für das gewählte Temperatur-Niveau. (Wird zusammen mit: -t xy.z verwendet).

Beispiel 1:

ht_netclient.py -t 15.5 -tmod frost -hc 2

Einstellung Temperatur-Niveau auf 15.5 Grad für Betriebs-Status 'frost' und Heizkreis: 2

Beispiel 2:

ht_netclient.py -t 21.5 -tmod heizen -hc 1

oder

ht_netclient.py -t 21.5 -tmod heizen

oder

ht_netclient.py -t 21.5

Einstellung Temperatur-Niveau auf 21.5 Grad für Betriebs-Status 'heizen' und Heizkreis: 1

Alle 3 Befehle sind gleichwertig, da Heizkreis 1 und Betriebs-Status 'heizen' Defaultwerte sind.

Folgende Steuer-Befehle sind für die Cxyz Regler (z.B.: CW100) realisiert:

Befehl und Parameter	Parameter	Funktion	Bemerkung
ht_netclient.py -t xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus.	Temperatur-Niveau (temporär). Wert bleibt nur bis zum nächsten Programmwechsel gesetzt.
ht_netclient.py -tc1 xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Comfort1.	Temperatur für den Betriebsstatus: Comfort1.
ht_netclient.py -tc2 xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Comfort2.	Temperatur für den Betriebsstatus: Comfort2.
ht_netclient.py -tc3 xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Comfort3.	Temperatur für den Betriebsstatus: Comfort3.
ht_netclient.py -teco xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Eco.	Temperatur für den Betriebsstatus: Eco.
ht_netclient.py -tman xy.z	Temperatur (float)	Einstellung des Temperatur-Niveaus für den Betriebsstatus: Manuell.	Temperatur für den Betriebsstatus: Manuell. Im Regler-Display wird der Status: Manuell angezeigt. Der Temperatur-Wert bleibt konstant bestehen ohne Programmsteuerung.
ht_netclient.py -ecomode x	WERTe: 0 := Off 1 := Hold_OutD 2 := Hold_Room 3 := Reduced	Einstellung der Eco-Mode Betriebsart.	Betriebsart des Eco-Modes.
ht_netclient.py -b WERT	WERTe: auto manual	Es wird der Betriebs-Status der Heizung fest auf den angegebenen Parameter eingestellt. Die Werte der Temperatur -Niveaus verändern sich nicht.	Der jeweilige Betriebs-Status wird eingestellt und am Bedienteil (Cxyz) angezeigt.
ht_netclient.py -hc CIRCUIT	CIRCUIT: 1 ... 4	Heizkreis Nummer. Default: 1	Heizkreis-Nummer für das gewählte Temperatur-Niveau. (Wird zusammen mit: -tc1..3 xy.z verwendet).

Beispiel 1:

```
ht_netclient.py -tc2 21.5 -hc 2  
Einstellung Temperatur-Niveau:Comfort2 auf 21.5 Grad und Heizkreis: 2
```

Beispiel 2:

```
ht_netclient.py -tc1 21.5  
Einstellung Temperatur-Niveau:Comfort1 auf 21.5 Grad und Heizkreis: 1  
Heizkreis 1 ist Defaultwert und braucht nicht angegeben zu werden.
```

Regler-Typ unabhängige Befehle:

Befehl und Parameter	Parameter	Funktion	Bemerkung
ht_netclient.py -h	--	Hilfe-Funktion	Ausgabe der Hilfe-Funktion
ht_netclient.py -ht_adr ADR	ADR: (13)dez. (10)dez.	Geräte-Adresse des 'ht_transceiver'-Adapters.	Die kommandierte Adresse wird erst nach einem Reset-Kommando an den 'ht_transceiver' aktiv. MB-LAN und NetCom100 haben die Geräte-Adresse: 13
ht_netclient.py -ht_RST 1	1	Reset des 'ht_transceivers'.	Ein zuvor eingestellte Geräteadresse wird nach dem Reset aktiviert.

Bei allen Befehlen ist mit einer Wartezeit von mehr als 5 Sekunden vor einer Reaktion der Heizungsanlage zu rechnen.

Innerhalb dieser Zeit sollten keine weiteren Befehle an den Adapter gesendet werden.

Ausgenommen davon ist nur der Reset-Befehl, der jedoch nur einen Reset des 'ht_transceiver'-Boards ausführt.

Falls in einem Heizungs-System der 'ht_transceiver' und MB-Lan oder NetCom100 aktiv sind **muss** der 'ht_transceiver' auf die Geräte-Adresse: 10 eingestellt werden, damit Telegramm-Kollisionen auf dem HT-Bus vermieden werden.

Folgende Befehles-Sequenz ist für die Änderung der Geräte-Adresse (auf 10) nötig:

1. ht_netclient.py -ht_adr 10
2. ht_netclient.py -ht_RST 1

Folgende Bilder zeigen die Bedienteil-Anzeigen für die jeweilige Netcom-Betriebsart an einem FW100 Regler. Andere Regler-Typen haben diese Anzeige u. U. nicht.



Abbildung 22: NC "Auto"



Abbildung 23: NC
"Heizen"



Abbildung 21: NC
"Sparen"



Abbildung 25: NC "Frost"



Abbildung 24: Temperatur-
Niveaus

Der NetCom-Modus „NC“ kann nur durch manuelle Betätigung des Mode-Wähler am Bedienteil Fwxyz /Frxyz verlassen werden.

Das einmal eingestellte Temperatur-Niveau (hier 22.5 Grad für Heizen) bleibt konstant erhalten solange dies nicht erneut überschrieben wird. Somit wird bei jedem Wechsel in die Betriebsart: „Heizen“ dieses Temperatur-Niveau eingestellt.

Dieses Verhalten ist also anders als das temporäre manuelle Verstellen der gewünschten Soll-Temperatur am Einstellrad, welche nur bis zum nächsten Betriebsart-Wechsel erhalten bleibt (Details siehe Bedienungsanleitungen).

3 Installation

Vor der Installation der Applikation ist das Betriebssystem zu aktualisieren. Je nach Betriebssystem sind unterschiedliche Aktionen erforderlich.

Diese Beschreibung beschränkt sich auf das Betriebssystem: 'Linux-Debian Jessie und Buster'.

Je nach Linux-Version und Ausgabestand sind einige der folgenden Aktionen nicht erforderlich. Ebenso hängen die Aktionen vom RaspberryPi Typ ab.

3.1 Betriebssystem

Aktualisierung des Betriebssystem mit:

```
$ sudo apt-get update
```

Den letzten Ausgabestand aktivieren:

```
$ sudo apt-get upgrade
```

Python3 installieren (falls noch nicht vorhanden):

```
$ sudo apt-get install python3
```

Seriellen Treiber für Python3 laden:

```
$ sudo apt-get install python3-serial
```

setup tools und GPIO Treiber für Python3 laden:

```
$ sudo apt-get install python3-setuptools  
$ sudo apt-get install RPI.GPIO
```

TK (GUI) Treiber für Python3 laden:

```
$ sudo apt-get install python3-tk
```

Perl objekt-orientiertes RRDTool Interface installieren:

```
$ sudo apt-get install librrdtool-oo-perl  
  
anschliessend  
$ sudo apt-get autoremove
```

RRDTool Datenbank installieren:

```
$ sudo apt-get install rrdtool
```

User in Gruppe <dialout> aufnehmen:

```
$ sudo adduser 'username' dialout
```

Tool: git installieren:

```
$ sudo apt-get install git
```

3.2 **UART Schnittstelle**

Deaktivieren der default eingeschalteten TTY-Systemausgaben (RaspberryPi):

```
# sudo raspi-config
```

```
5 Interfacing Options Configure connections to peripherals
P6 Serial           Enable/Disable shell and kernel messages on the serial connection
Would you like a login shell to be accessible over serial?
→ <No>
Would you like the serial port hardware to be enabled?
→ <Yes>
# sudo reboot
```

Nach dem reboot folgende Aktionen ausführen:

Für RaspberryPi Versionen < 3

```
# sudo systemctl stop serial-getty@ttyAMA0.service
# sudo systemctl disable serial-getty@ttyAMA0.service
```

Für RaspberryPi Versionen >= 3

```
# sudo systemctl stop serial-getty@ttyS0.service
# sudo systemctl disable serial-getty@ttyS0.service
```

Aktion prüfen:

```
# sudo systemctl status serial-getty@ttyAMA0.service bzw.
# sudo systemctl status serial-getty@ttyS0.service
```

Ausgabe:

```
serial-getty@ttyAMA0.service - Serial Getty on ttyAMA0 bzw.
serial-getty@ttyAMA0.service - Serial Getty on ttyS0
Loaded: loaded (/lib/systemd/system/serial-getty@.service; disabled; vendor preset: enabled)
Active: inactive (dead)
Docs: man:agetty(8)
      man:systemd-getty-generator(8)
      http://0pointer.de/blog/projects/serial-console.html
```

Siehe auch [1]

Deaktivieren des default eingeschalteten Bluetooth-IF (nur RaspberryPi **B3/B4**):

```
# sudo systemctl disable hcuart
```

Hinweise zu diesem Thema unter:

```
https://www.raspberrypi.org/documentation/configuration/uart.md
```

Siehe auch [13]

Neustart des Raspberry Pi (falls keine MQTT-Schnittstelle erforderlich ist):

```
# sudo reboot
```

3.3 **Komponenten für MQTT**

Key für zukünftige Updates holen / installieren (je nach Debian-Version)

```
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
$ sudo apt-key add mosquitto-repo.gpg.key
$ cd /etc/apt/sources.list.d
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
      oder
```

```
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
oder
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

System aktualisieren

```
$ sudo apt-get update
```

Installation des MQTT-Broker

```
$ sudo apt-get install mosquitto
```

Installation der MQTT Clients (mosquitto_sub und mosquitto_pub)

```
$ sudo apt-get install mosquitto-clients
```

```
$ sudo apt-get autoremove
```

Test ob der Broker läuft

```
$ ps -aux |grep mosquitto
```

Resultat aus der ps - Anzeige:

```
... ... ... /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

Installation von weiteren python-Modulen

Tool 'pip' installieren

```
$ sudo apt-get install python3-pip
```

Holen von paho-mqtt

```
$ sudo pip3 install paho-mqtt
```

oder auch

```
$ sudo pip-3.2 install paho-mqtt (je nach pip-Version)
```

Neustart des Raspberry Pi:

```
# sudo reboot (Falls keine HomeAssistant-Schnittstelle erforderlich ist)
```

Details siehe auch [12]

3.3.1 HomeAssistant (Hassio) Schnittstelle

Alle folgenden Anpassungen des Start-Scripts sind NUR erforderlich, wenn folgendes NICHT passt:

User = 'pi'
Pfad = /home/pi/HT3/....

Ansonsten kann das Startscript ohne Anpassungen mit dem Tool: 'systemctl' aktiviert werden.

Installation des Startscripts 'ht_2hass':

Anpassung und Aktivieren des Startscripts 'ht_2hass' (als root für user:**pi**):

1. Username und Verzeichnisse sind gegebenenfalls anzupassen

(Dies ist **nur erforderlich**, wenn der user **NICHT 'pi'** ist und das Verzeichnis HT3 nicht unter /home/pi/ liegt.)
sudo nano ./HT3/sw/etc/sysconfig/ht_2hass
USER="pi" <<- auf erforderlichen Wert korrigieren
DAEMON=/home/\$USER/HT3/sw/\$NAME
PIDFILE=/home/\$USER/HT3/sw/var/run/\$NAME.pid
APPLICATION_FOLDER=/home/\$USER/HT3/sw/

Datei speichern

2. Datei kopieren:

sudo cp ./HT3/sw/etc/sysconfig/ht_2hass /etc/init.d

Script aktivieren:

cd /etc/init.d
sudo chmod +x /etc/init.d/ht_2hass
sudo systemctl enable ht_2hass

Damit wird beim nächsten Start des RPi der daemon: ht_2hassio.py automatisch gestartet.

Neustart des Raspberry Pi:

sudo reboot

3.4 Applikationen

(Vor der Installation der Applikation ist das Betriebssystem zu aktualisieren).

Die aktuelle Software mit Dokumentation von github.com holen (als user 'pi'):

```
$ cd  
$ git clone https://github.com/norberts1/hometop\_HT3.git  
$ Folder: HT3 zu ~/. verschieben  
$ mv ~/hometop_HT3/HT3 ~/.
```

Einrichten eines Datenbank-Verzeichnisses (als root):

(Dies ist **nur erforderlich**, falls die Datenbank **nicht** auf dem Default-Verzeichnis: **./HT3/sw/var/databases** sein soll)

Beispiel für einen Verzeichnis auf dem USB-Stick

```
# mkdir -p /media/usbstick/HT3/sw/var/databases  
# cd /media  
# chmod -R 775 ./usbstick/HT3/  
# chown -R 'username' ./usbstick/HT3/
```

Bemerkung:

Die **Verzeichnis-Struktur unterhalb** von '/media/usbstick' muss so angelegt werden, da das Perl-Grafikscript so auf diese Struktur zugreift.

Anpassen der Konfiguration an neues Datenbank-Verzeichnis (als user: pi):

(Dies ist **nur erforderlich**, falls die Datenbank **nicht** auf dem Default-Verzeichnis: **./HT3/sw/var/databases** sein soll)

Datei: ./etc/config/HT3_db_cf.xml anpassen

```
$ nano ./etc/config/HT3_db_cf.xml  
von  
$ <dbname_sqlite>./var/databases/HT3_db.sqlite</dbname_sqlite>  
in  
$ <dbname_sqlite>/media/usbstick/HT3/sw/var/databases/HT3_db.sqlite</dbname_sqlite>  
und von  
$ <dbname_rrd>./var/databases/HT3_db_rrd</dbname_rrd>  
in  
$ <dbname_rrd>/media/usbstick/HT3/sw/var/databases/HT3_db_rrd</dbname_rrd>
```

Danach Datei speichern

Erzeugen der Datenbanken (als user: pi):

```
$ cd ./HT3/sw  
$ ./create_databases.py
```

!! Achtung

Das Erzeugen der Datenbanken dauert einige Zeit (> 5 und < 15 Minuten) auf dem Raspberry Pi.

Hinweis:

Soll die sqlite-Datenbank genutzt werden, so ist vor dem Erzeugen der Datenbank die sqlite-db im Konfigurationsfile 'HT3_db_cfg.xml' zu aktivieren: ,on'.

```
...
<dbname_sqlite>./var/databases/HT3_db.sqlite</dbname_sqlite>
<sql-db>
  <enable>on</enable>
```

Alle folgenden Anpassungen der Start-Scripte sind NUR erforderlich, wenn folgendes **NICHT** passt:

User = 'pi'
Pfad = /home/pi/HT3/....
Ansonsten können die Startscripte ohne Anpassungen mit dem Tool: 'systemctl' aktiviert werden.

Anpassung und Aktivieren des Startscripts 'ht_collgate' (als root für user:pi):

1. Username und Verzeichnisse sind gegebenenfalls anzupassen
(Dies ist **nur erforderlich**, wenn der user **NICHT 'pi'** ist und das Verzeichnis HT3 nicht unter /home/pi/ liegt.)

```
# sudo nano ./HT3/sw/etc/sysconfig/ht_collgate
USER="pi"                                     <<<- auf erforderlichen Wert korrigieren
DAEMON=/home/$USER/HT3/sw/$NAME
PIDFILE=/home/$USER/HT3/sw/var/run/$NAME.pid
APPLICATION_FOLDER=/home/$USER/HT3/sw/
```

Datei speichern
2. Datei kopieren:

```
# sudo cp ./HT3/sw/etc/sysconfig/ht_collgate /etc/init.d
```

Script aktivieren:

```
# cd /etc/init.d
# sudo chmod +x /etc/init.d/ht_collgate
# sudo systemctl enable ht_collgate
```

Anpassung und Aktivieren des Startscripts 'ht_proxy' (als root für user:pi):

(Nur erforderlich, wenn man ht_proxy - SERVER/CLIENT(s) verwenden will)

1. Username und Verzeichnisse sind gegebenenfalls anzupassen
(Dies ist **nur erforderlich**, wenn der user **NICHT 'pi'** ist und das Verzeichnis HT3 nicht unter /home/pi/ liegt.)

```
# sudo nano ./HT3/sw/etc/sysconfig/ht_proxy
USER="pi"                                     <<<- auf erforderlichen Wert korrigieren
DAEMON=/home/$USER/HT3/sw/$NAME
PIDFILE=/home/$USER/HT3/sw/var/run/$NAME.pid
APPLICATION_FOLDER=/home/$USER/HT3/sw/
```

Datei speichern
2. Datei kopieren:

```
# sudo cp ./HT3/sw/etc/sysconfig/ht_proxy /etc/init.d
```

Script aktivieren:

```
# cd /etc/init.d
# sudo chmod +x /etc/init.d/ ht_proxy
# sudo systemctl enable ht_proxy
```

Anpassung und Aktivieren des Startscripts 'httpd' (als root für user:**pi**):

(Nur erforderlich, falls man keinen anderen Http-Server installieren will)

1. Username und Verzeichnisse sind gegebenenfalls anzupassen

(Dies ist **nur erforderlich**, wenn der user **NICHT 'pi'** ist und das Verzeichnis HT3 nicht unter /home/pi/ liegt.)

```
# sudo nano ./HT3/sw/etc/sysconfig/httpd
USER="pi"                                     <--- auf erforderlichen Wert korrigieren
DAEMON=/home/$USER/HT3/sw/$NAME
PIDFILE=/home/$USER/HT3/sw/var/run/$NAME.pid
APPLICATION_FOLDER=/home/$USER/HT3/sw/
```

Datei speichern

2. Datei kopieren:

```
# sudo cp ./HT3/sw/etc/sysconfig/httpd /etc/init.d
```

Script aktivieren:

```
# cd /etc/init.d
# sudo chmod +x /etc/init.d/ httpd
# sudo systemctl enable httpd
```

Anpassen der Applikationen an die Schnittstelle (Beispiel: ASYNC):

Je nach Schnittstellen-Typ folgende Devices verwenden:

```
# deviceport="/dev/ttyAMA0" <--- UART-Schnittstelle des Raspberry Pi
```

oder

```
# deviceport="/dev/serial0" <--- UART-Schnittstelle des Raspberry Pi (Pi4/buster)
```

oder

```
# deviceport="/dev/ttyUSB0" <--- 1. USB -Schnittstelle des Raspberry Pi / Laptop
```

oder

```
# deviceport="/dev/ttyUSB1" <--- 2. USB -Schnittstelle des Raspberry Pi / Laptop
```

Die Konfigurations-Anpassung erfolgt im File:

\$./HT3/sw/etc/config/HT3_db_cfg.xml bei den Parametern:

(**grün** := Wert muss auf 'ASYNC' gesetzt werden.

gelb := Werte müssen auf korrekte Schnittstelle und Baudrate eingestellt werden.)

```
<data_interface>
<comm_type>ASYNC</comm_type> <!-- communication-types are:
                                ASYNC:=tty/comport; SOCKET:= socket-interface -->
<proto_type>RAW</proto_type> <!-- protocoll-types   are:
                                RAW:=transparent ; TRX :=TBD (transceiver-messages with header) -->
<parameter name="ASYNC">
!-----!
<serialdevice>/dev/ttyAMA0</serialdevice>
!---- Hier Achtung: Besonderheit bei Raspberry4 und debian-buster -----!
<serialdevice>/dev/serial0</serialdevice>
!-----!
<inputtestfilepath></inputtestfilepath>
<baudrate>9600</baudrate> <---- Für HT3_miniAdapter und HT3_microAdapter
ODER
<baudrate>19200</baudrate> <---- Für ht_transceiver (ht_piduino und ht_pitiny)
<config>"8N1"</config> <-- only 8N1 available -->
</parameter>
<parameter name="SOCKET"> <---- Wird in diesem Modus nicht genutzt
    <client_config_file>./etc/config/ht_proxy_cfg.xml</client_config_file>
</parameter>
</data_interface>
```

Anpassen der Applikationen an die Schnittstelle (Beispiel: SOCKET):

Die Konfigurations-Anpassung erfolgen in den Files:

\$./HT3/sw/etc/config/HT3_db_cfg.xml und ./HT3/sw/etc/config/ht_proxy_cfg.xml:
(grün := Wert muss auf 'SOCKET' gesetzt werden.
gelb := Werte müssen auf korrekte Schnittstelle und Baudrate eingestellt,
gewünschte Portnummern und Logfile-Namen können angepasst werden.)

HT3_db_cfg.xml:

```
<data_interface>
  <comm_type>SOCKET</comm_type> <!-- communication-types are:
    ASYNC:=tty/comport; SOCKET:= socket-interface -->
  <proto_type>RAW</proto_type> <!-- protocoll-types are:
    RAW:=transparent ; TRX :=TBD (transceiver-messages with header) -->
  <parameter name="ASYNC"> <----- Wird in diesem Modus nicht genutzt
    <serialdevice>/dev/ttyAMA0</serialdevice>
    <inputtestfilepath></inputtestfilepath>
    <baudrate>19200</baudrate>
    <config>"8N1"</config> <!-- only 8N1 available -->
  </parameter>
  <parameter name="SOCKET">
    <client_config_file>/etc/config/ht_proxy_cfg.xml</client_config_file>
  </parameter>
</data_interface>
```

ht_proxy_cfg.xml :

(für den Server-Anteil)

```
<proxy_server>
  <serveraddress></serveraddress> <----- leer := Zugriff aller Clients auf Server erlaubt
  <servername></servername> <----- leer := Zugriff aller Clients auf Server erlaubt
  <portnumber>48088</portnumber>
  <logfilepath>/var/log/ht_proxy.log</logfilepath>
  <ht_transceiver_if devicename="RX">
    <parameter>
      <serialdevice>/dev/ttyAMA0</serialdevice>
      ----- Hier Achtung: Besonderheit bei Raspberry4 und debian-buster -----
      <serialdevice>/dev/serial0</serialdevice>
      -----
      <baudrate>19200</baudrate>
      <config>"8N1"</config> <!-- only 8N1 available -->
    </parameter>
  ....
```

(für den Client-Anteil)

```
<proxy_client devicename="RX">
  <serveraddress>localhost</serveraddress> <----- IP-Adresse des proxy-server-host: localhost/192.168.2.1/...
  <servername></servername> <----- oder proxy-server Hostname
  <portnumber>48088</portnumber>
  <logfilepath>/var/log/ht_client_rx.log</logfilepath>
  <devicetype>RX</devicetype>
</proxy_client>
<proxy_client devicename="MODEM">
  <serveraddress>localhost</serveraddress> <----- IP-Adresse des proxy-server-host: localhost/192.168.2.1/...
  <servername></servername> <----- oder proxy-server Hostname
  <portnumber>48088</portnumber>
  <logfilepath>/var/log/ht_client_modem.log</logfilepath>
  <devicetype>MODEM</devicetype>
</proxy_client>
```

Neustart des Rechners (als root):

```
# sudo reboot
```

Nach dem Neustart des Rechners müssen der 'ht_proxy.py'-Server, die
Applikation 'ht_coligate.py' und 'httpd.py' automatisch gestartet worden sein.
Prüfung mit: ps ax | grep python

3.4.1 Kurzbeschreibung der Softwaremodule

Der proxy-server wird zeitlich vor allen proxy-clients gestartet, damit eine Socket-Verbindung erstellt, die Daten erfasst und in die Datenbanken geschrieben werden können.

Es werden dann alle 2 Minuten die Grafikausgaben der rrdtool-Datenbank im Verzeichnis: <Zielverzeichnis>/HT3/sw/etc/html/ als *.png Files abgelegt. Alte PNG-Files werden überschrieben.

Ebenfalls muss der Http-Server 'httpd.py' automatisch gestartet worden sein. Dieser Server erwartet Anfragen auf dem Port:**48086** und sobald PNG-Dateien erzeugt wurden (alle 2 Minuten Intervall basiert), werden diese vom Browser angezeigt.

Eine funktionale Systemübersicht zeigen die folgenden Bilder. Dabei ermöglicht die unterschiedliche Konfiguration die Verwendung verschiedener Hardware (ht_transceiver, HT3_mini-Adapter oder HT3_micro-Adapter).

Jeder Adapter-Typ kann direkt an einem ComPort (/dev/ttyAMA0 bzw. /dev/serial0) betrieben werden.

Dazu ist die Konfiguration im File: ./HT3/sw/etc/config/HT3_db_cfg.cml) auf „**ASYNC**“ einzustellen und die richtige Baudrate (19200 / 9600) auszuwählen:

```
<data_interface>
<comm_type>ASYNC</comm_type>
```

Jeder Adapter-Typ kann aber auch mit dem 'ht_proxy' und dem SOCKET-Interface betrieben werden.

Dazu ist die Konfiguration im File: ./HT3/sw/etc/config/HT3_db_cfg.cml) auf „**SOCKET**“ einzustellen:

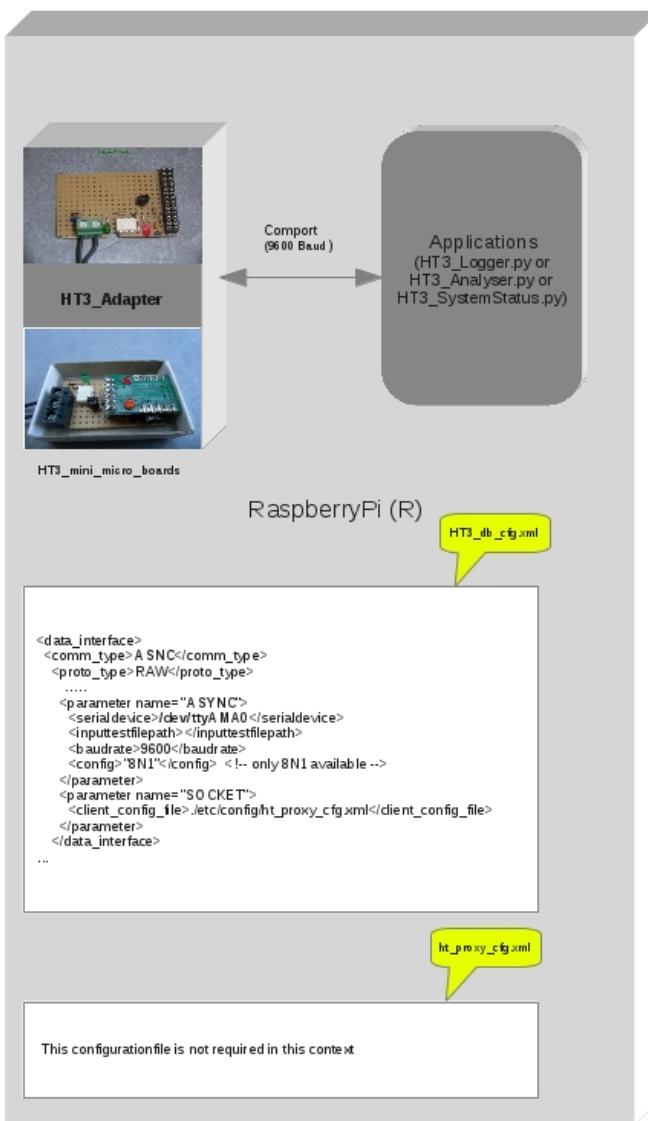
```
<data_interface>
<comm_type>SOCKET</comm_type>
```

Zusätzlich müssen der ht_proxy-server und der ht_proxy-client konfiguriert werden. Dies wird im File: ./HT3/sw/etc/config/ht_proxy_cfg.xml gemacht.

Achtung:

Per Default ist das Projekt bereits auf folgende Werte eingestellt:

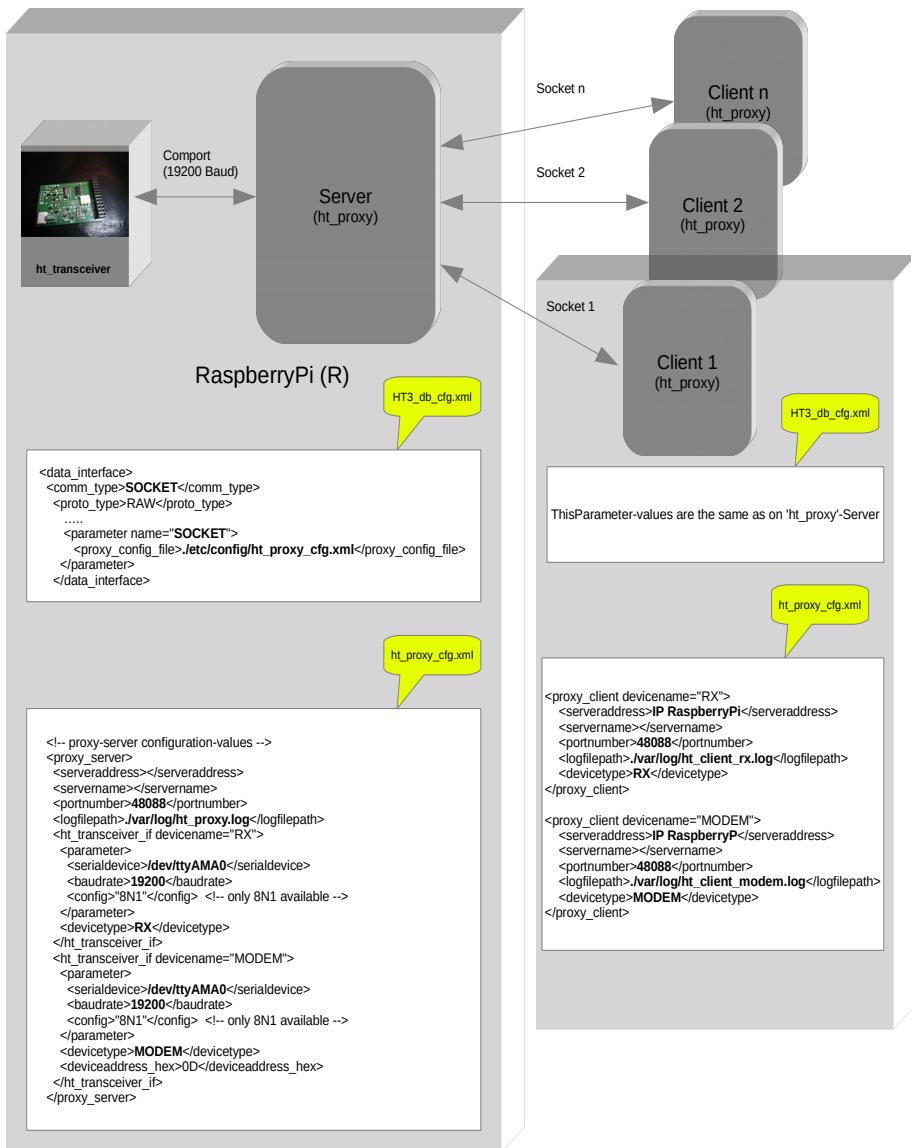
1. Socket-Schnittstelle ist aktiv.
2. ht_proxy-server läuft auf dem Host: ,localhost‘
3. Adapter werden vom ht_proxy-Server über die serielle Schnittstelle betrieben und die erfassten Daten werden über die Socket-Schnittstelle verteilt.
4. Die Steuerung der Heizung (mit aktiven Adapters) erfolgt über den ht_proxy-Server.



Raspberry Pi (R) with 'HT3_mini- / HT3_micro-Adapter' and the required configuration

junkys@gmx.de

Abbildung 26: HT3 mini- / micro- Adapter Konfiguration



RaspberryPi (R) with 'ht_transceiver'-board and 'ht_proxy' configured as server.

There are socket-connections possible for N clients,
where N should be ≤ 3 on Raspberry Pi Rev: A/A+/B and B+.

Attention: Portnumber changed from 8088 to 48088 (SW-Rev. ≥ 0.4)

Junkyzs at gmx dot de

Abbildung 27: ht_transceiver mit ht_proxy Konfiguration

3.4.2 Mögliche Konfigurationen (HW/SW)

Folgende Konfigurationen sind möglich:

Hardware am Raspberry Pi	Konfiguration für: ASYNC-Interface	Konfiguration für: SOCKET-Interface (proxy-server)	Proxy-Server 'ht_proxy' installiert und aktiv	Steuerung der Heizung möglich ?
HT3-miniAdapter	File: HT3_db_cfg.xml Werte: <comm_type> ASYNC <baudrate> 9600 <serialdevice>/dev/ttyAMA0	--	Nicht möglich, Konflikt mit dem ComPort	Nein
HT3-microAdapter	File: HT3_db_cfg.xml Werte: <comm_type> ASYNC <baudrate> 9600 <serialdevice>/dev/ttyUSB(x)	--	Nicht möglich, Konflikt mit dem ComPort	Nein
ht_piduino ht_pitiny	File: HT3_db_cfg.xml Werte: <comm_type> ASYNC <baudrate> 19200 <serialdevice>/dev/ttyAMA0	--	Nicht möglich, Konflikt mit dem ComPort	Nein
HT3-miniAdapter	--	File: HT3_db_cfg.xml Werte: <comm_type> SOCKET <proxy_config_file> Pfad auf File File: ht_proxy_cfg.xml <baudrate> 9600 <serialdevice>/dev/ttyAMA0	Ja, Proxy-Server muss installiert sein und laufen.	Nein
HT3-microAdapter	--	File: HT3_db_cfg.xml Werte: <comm_type> SOCKET <proxy_config_file> Pfad auf File File: ht_proxy_cfg.xml <baudrate> 9600 <serialdevice>/dev/ttyUSB(x)	Ja, Proxy-Server muss installiert sein und laufen.	Nein
ht_piduino ht_pitiny	--	File: HT3_db_cfg.xml Werte: <comm_type> SOCKET <proxy_config_file> Pfad auf File File: ht_proxy_cfg.xml <baudrate> 19200 <serialdevice>/dev/ttyAMA0	Ja, Proxy-Server muss installiert sein und laufen.	Ja

4 HT Applikation im Betrieb

Folgende Bilder zeigen grafische Ausgaben aus der rrdtool-Datenbank von erfassten Heizungssystemdaten. Die Grafiken werden als PNG-Dateien erzeugt und mit einem Browser dargestellt (das Darstellungs-intervall hier ist 2 Tage).

Der Aufruf erfolgt in einem Browser (firefox etc.) mit:

<http://<IP-Adr. des RaspberryPi>:48086>

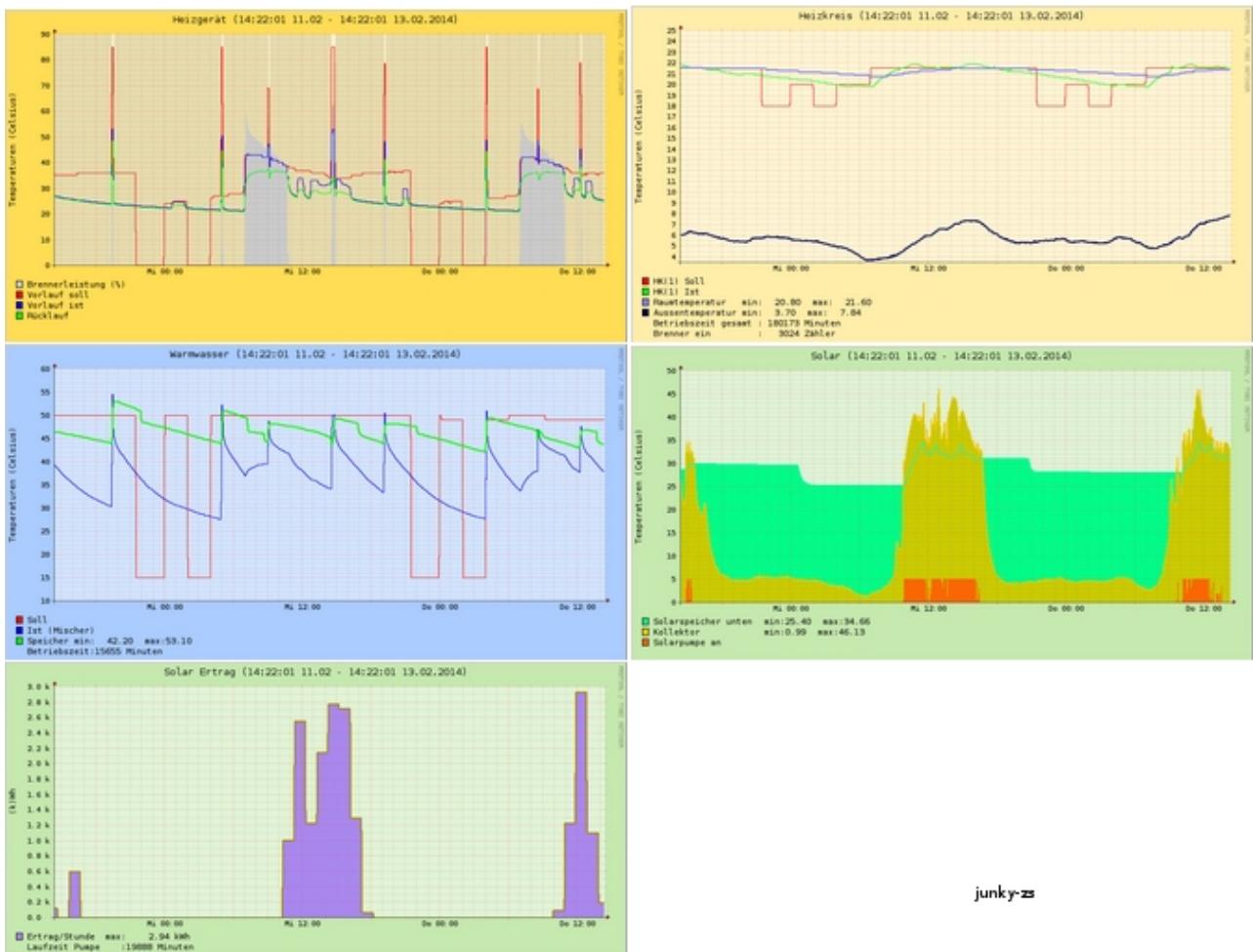


Abbildung 28: HT3 Systemhistorie im Browserfenster

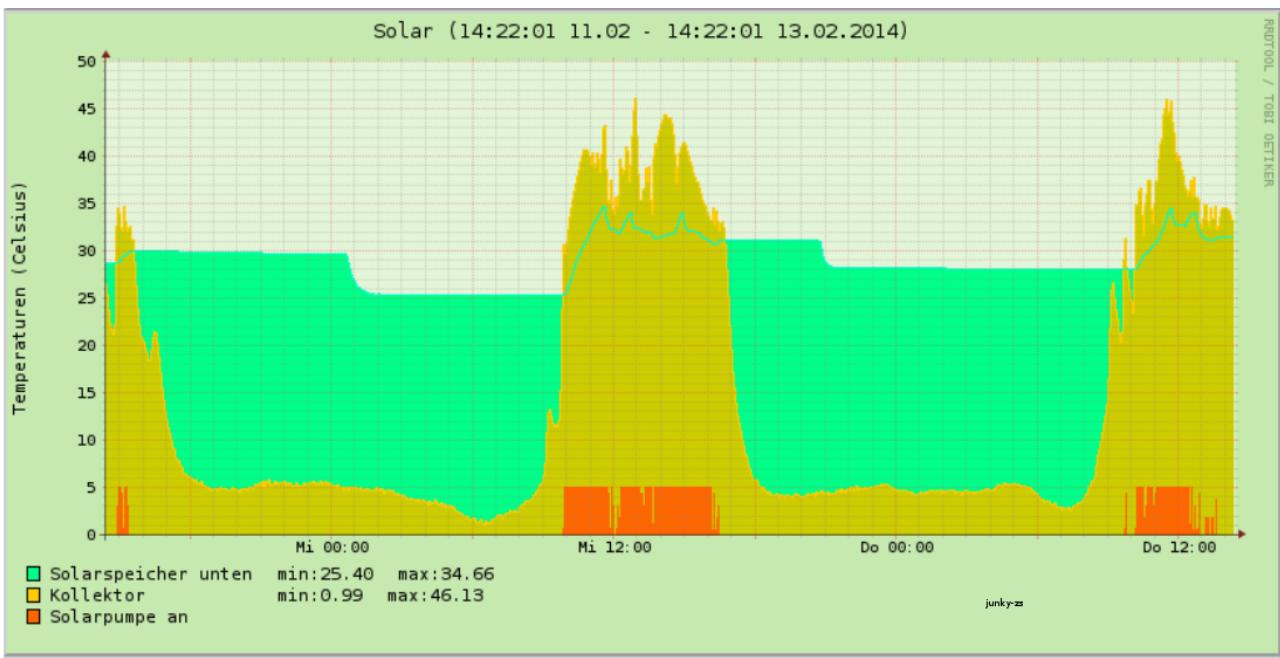


Abbildung 29: HT3 Solarhistorie im Browserfenster

Das HTML-Modul ist zur Zeit sehr einfach gehalten und nur für die Anzeige der Grafiken ausgelegt. Verschiedene Anpassungen sind denkbar, z.B. die Auswahl des Anzeigeintervalls. Dies ist aber noch nicht realisiert.

HTML-MODUL './HT3/sw/etc/index.html' :

```
<HTML>
<HEAD>
<TITLE>Heizungs Historie</TITLE></HEAD>
<BODY>
<IMG src="./HT3_Heizgeraet.png" alt="Heizgeraet">
<IMG src="./HT3_Warmwasser.png" alt="Warmwasser">
<BR>
<IMG src="./HT3_Heizkreis1.png" alt="Heizkreis1">
<IMG src="./HT3_Heizkreis2.png">
<BR>
<IMG src="./HT3_Heizkreis3.png">
<IMG src="./HT3_Heizkreis4.png">
<BR>
<IMG src="./HT3_Solar.png" alt="Solar">
<IMG src="./HT3_Solarertrag.png" alt="Solarertrag">
</BODY>
</HTML>
```

Die Grafiken werden mit einem Perl-Modul aus den Daten der rrdtool-Datenbank für das ausgewählte Intervall erzeugt und im '<Zielverzeichnis>/HT3/sw/etc/html'-Verzeichnis abgelegt. Das 'Zielverzeichnis' ist nach der Installation das Installations-Verzeichnis.

Falls man den 'Http-Daemon Lite' (./HT3/sw/etc/html/httpd.py) nutzen möchte, müssen die erzeugten Grafiken im Verzeichnis des Daemon's liegen.

5 Weiterführende Literatur und URL's

Python 3 Lernen und professionell anwenden Verlag: mitp
Michael Weigend

Python 3 Das umfassende Handbuch Verlag: Galileo Computing
Johannes Ernesti und Peter Kaiser

- [1] <http://kampis-elektroecke.de> Elektronik, Code und mehr
- [2] <http://oss.oetiker.ch/rrdtool> About RRDtool
- [3] <http://www.mrtg.org/rrdtool/gallery/index.en.html> RRDtool Gallery
- [4] <http://code.google.com/p/pyrrd> A Pure-Python OO wrapper for RRDTool
- [5] <http://www.mikrocontroller.net/forum/Haus & Smart Home> Forum: Haus & Smarthome
- [6] <http://pi.gadgetoid.com/pinout/atmega328-arduino>. Atmega328 over SPI
- [7] <https://www.mikrocontroller.net/topic/317004#3925213>. Forum zum Thema 'ht_transceiver'
- [8] https://github.com/norberts1/hometop_HT3. Hometop HT3 Soft-/Hard-Ware.
- [9] https://github.com/norberts1/hometop_ht_transceiver. Hometop HT-Transceiver Soft-/Hard-Ware.
- [10] <http://oss.oetiker.ch/rrdtool/prog/rrdpython.en.html> RRDPython Firma: Oetiker
- [11] <http://www.hivemq.com/blog/mqtt-client-library-paho-python> MQTT Client Encyclopedia Paho Python
- [12] <http://www.steves-internet-guide.com/install-mosquitto-linux/> Installationshilfe
- [13] <https://www.raspberrypi.org/documentation/configuration/uart.md> UART configuration

6 Historie

Docu-Revision	Software-Revision	Remark
0.6 / 2021-03-07	≥ 0.4	Portnumbers changed, Homeassistant-IF added. Homeassistant Installation-Procedure added.
0.5 / 2020-08-04	< 0.4	Installation-Procedure modified.