

**Heizungs-Datenerfassung**  
**mit**  
**Raspberry Pi ®**  
**und**  
**Laptop/PC**

# Inhaltsverzeichnis

Kurzbeschreibung.....	3
1Hardware.....	4
1.1Adapter für Raspberry Pi®.....	4
1.1.1Schaltplan und Stückliste.....	5
1.1.2Funktionsbeschreibung.....	6
1.1.3Inbetriebnahme.....	7
1.1.4Realisierungsbeispiel.....	8
1.2USB-Adapter für PCs und Laptops (HT3-Microadapter).....	10
1.2.1Schaltplan und Stückliste.....	11
1.3Transceiver Frontend (ht-transceiver) für Raspberry Pi® / USB.....	12
1.3.1Adapter 'ht_piduino'.....	12
1.3.1.1'ht_piduino' Bild, Schaltplan und Stückliste.....	13
1.3.2Adapter 'ht_pitiny'.....	15
1.3.2.1'ht_pitiny' Bild, Schaltplan und Stückliste.....	15
1.3.3USB-MotherBoard für Adapter 'ht_transceiver' und UM2102.....	17
2Software.....	18
2.1Verzeichnis-Struktur.....	18
2.2Konfiguration.....	19
2.3Datenbanken.....	25
2.3.1Datenbank 'SQLite'.....	25
2.3.2Datenbank 'rrdtool'.....	28
2.4Applikationen.....	29
2.4.1HT3-Analyser.....	29
2.4.2HT3_Systemstatus.....	31
2.4.3HT3_Logger.....	32
2.5Comport <=> Socket proxy (Server & Client).....	32
2.5.1ht_proxy (proxy-server).....	32
2.5.2ht_client_example (proxy-client Beispiel).....	33
2.5.3ht_netclient (Heizungssteuer-Client).....	34
2.5.3.1ht_netclient Steuerbefehle.....	35
3Installation.....	37
3.1Betriebssystem.....	37
3.2Applikation.....	38
4HT3 Applikation im Betrieb.....	46
5Weiterführende Literatur und URL's.....	48

## Kurzbeschreibung

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

Die Adaption beschränkt sich z.Zeit auf die Protokolle von Heizungsanlagen mit Heatronic3©-Bus der Firma Junkers.

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 einen HT3-Telegrammanalysator mit zugehöriger grafischen Anzeige.

Durch Konfigurationsänderungen lässt sich die Software zu einer reinen grafischen Statusanzeige oder zu einem Datenlogger (ohne Grafikausgabe) ändern.

Die Daten werden nach Erfassung eines gültigen Telegramms in eine SQLite – Datenbank geschrieben. Im Rhythmus von 60 Sekunden (default) werden diese SQL-Daten in eine weitere Datenbank (rrdtool) übertragen. Mit diesem 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:  
Norbert Scharf Email: junky-zs@gmx.de

# 1 Hardware

Die Hardware besteht aus einem Adapter, der die Pegel-Wandlung und Anpassung der Heatronic3-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 Heatronic3-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 dieser für das Interface zur Auswertung 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®

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 auf der CPU-Seite des Adapters 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 Heatronic3-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:

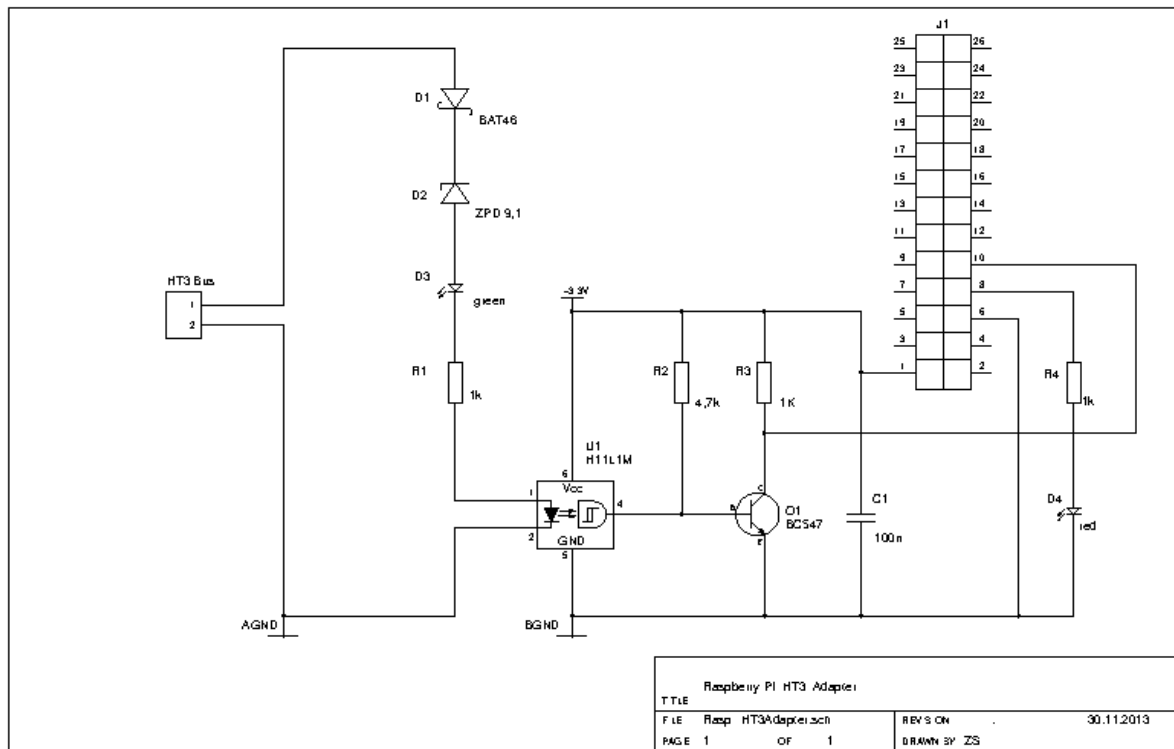


Abbildung 1: Schaltplan HT3 Adapter für Raspberry Pi

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-Lastung 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 Lötungen

*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 Hysterese und ist schnell genug für das Signalprotokoll mit: 9600 Baud.

Der Optokopplers 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 HT3-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 ( $>> 1 \text{ MOhm}$ ). 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 gleichen 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.

### 1.1.4 Realisierungsbeispiel

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

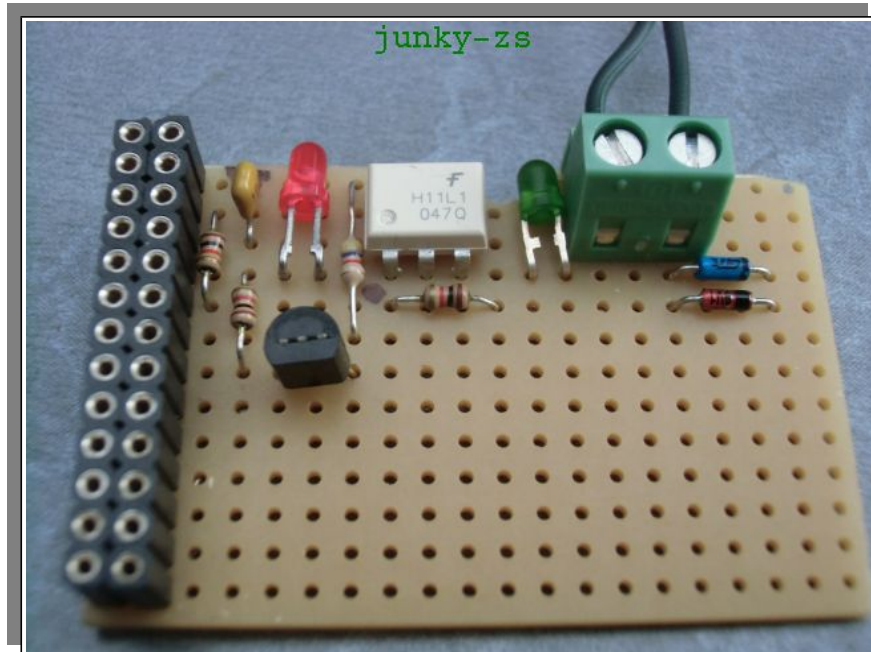


Abbildung 2: HT3 Adapter für Raspberry Pi

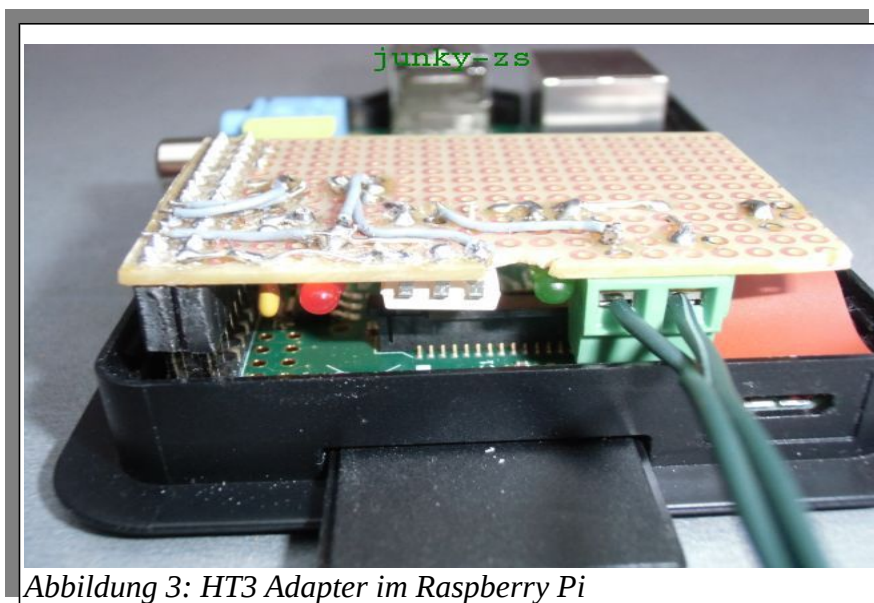


Abbildung 3: HT3 Adapter im Raspberry Pi



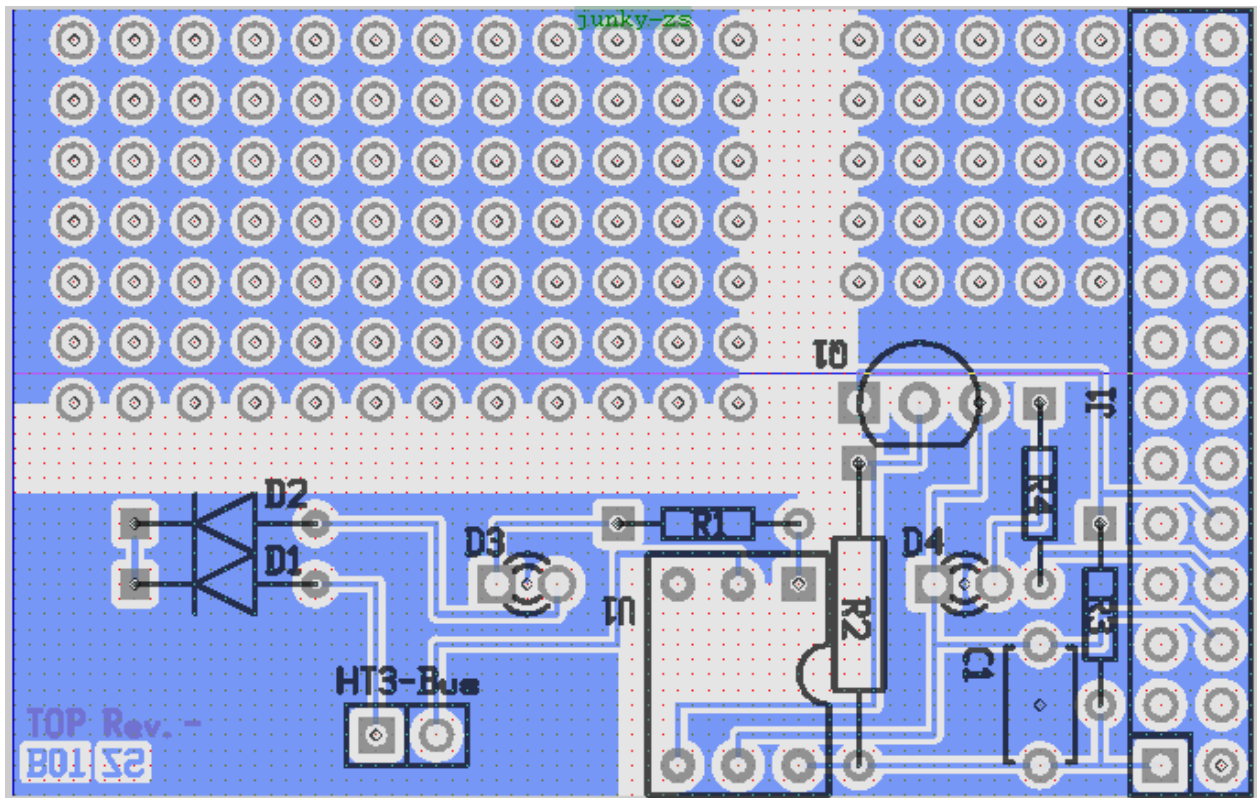


Abbildung 4: HT3 Adapter Layout

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 Lötungen 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)

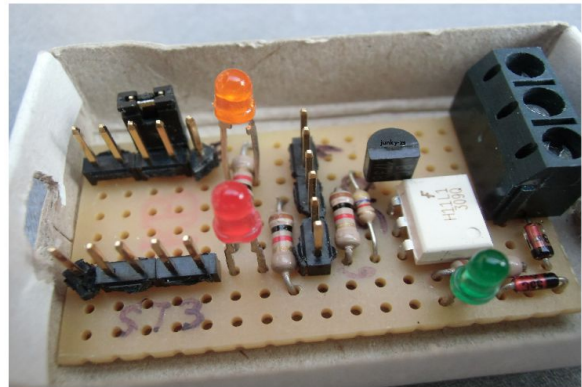


Abbildung 6: HT3-Microadapter  
(Basisplatine)



Abbildung 7: HT3 Microadapter (komplett)

### 1.2.1 Schaltplan und Stückliste

Im folgenden sind der Schaltplan und die zugehörige Stückliste aufgeführt:

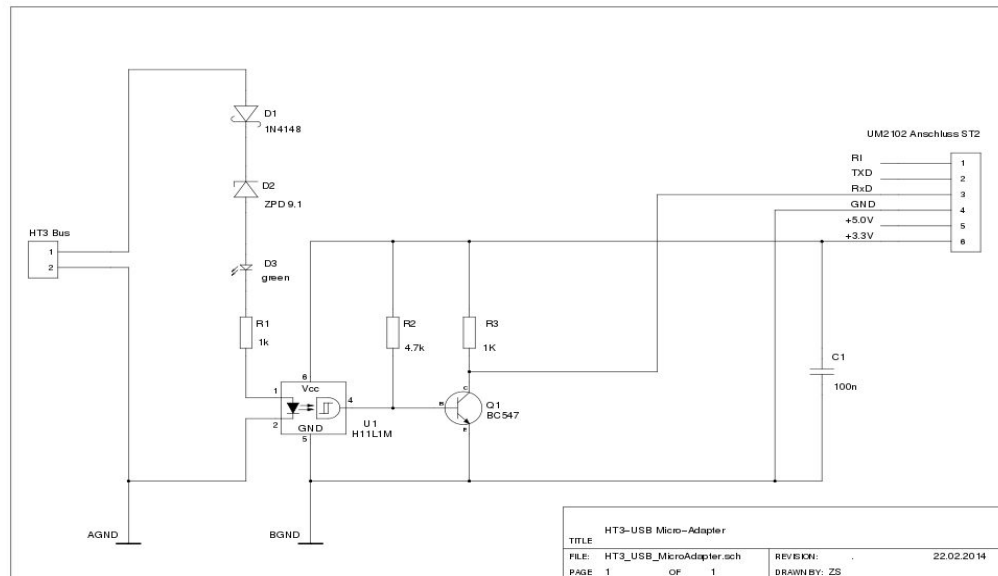


Abbildung 8: Schaltplan HT3 Microadapter (RS232->USB)

Name	Bezeichnung / Type	Anzahl	Bemerkung
--	Bauteile siehe Stückliste für den HT3-Adapter (RaspberryPi)	1 Satz	Siehe Tabelle 1: HT3 Adapterstückliste
UM2102	Mini USB-Modul Bausatz	1	ELV: Bestell-Nr.: 68-91859
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 Adaptern 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.

#### 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 Heizungsbus), ist die zweite Schnittstelle als Software-UART realisiert worden (als Interface zum Host).

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

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\_pitiny') 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)



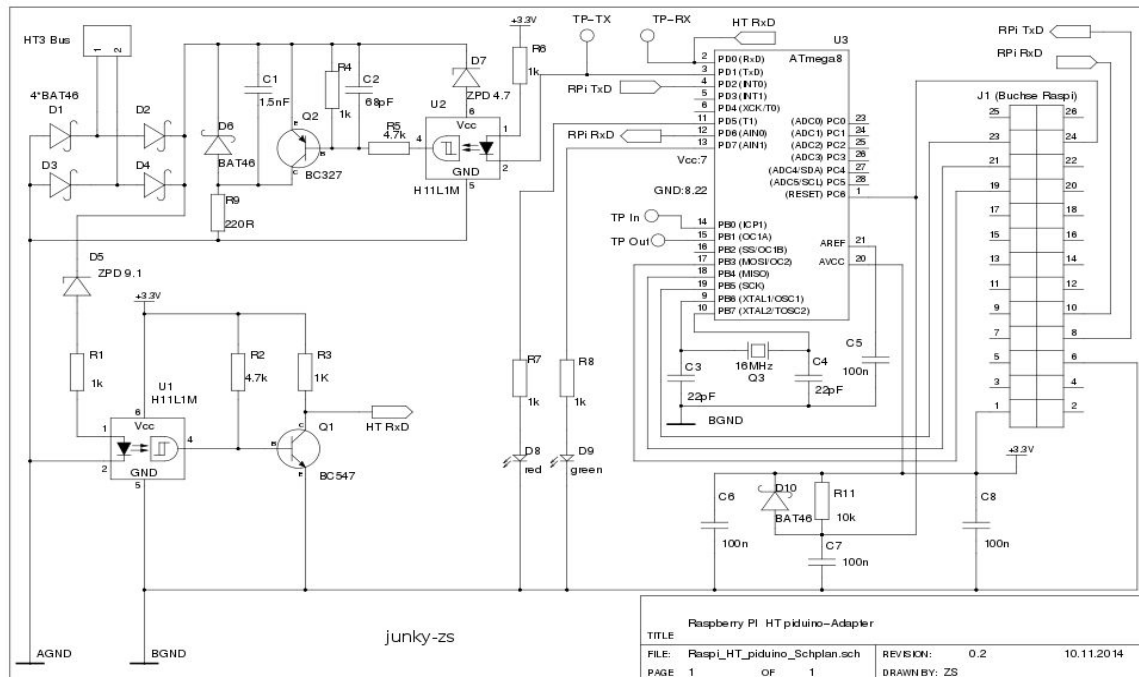


Abbildung 10: 'ht\_piduino' Schaltplan (mit ATmega328P !)

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: <a href="https://secure.reichelt.de/index.html?&amp;ACTION=20&amp;AWKID=998298&amp;PROVID=2084">https://secure.reichelt.de/index.html?&amp;ACTION=20&amp;AWKID=998298&amp;PROVID=2084</a>
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 3: 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 Heizungsbus. Der Programmcode belegt z.Zeit ca. 4kByte und lässt somit Platz für Erweiterungen.

#### 1.3.2.1 'ht\_pitiny' Bild, Schaltplan und Stückliste

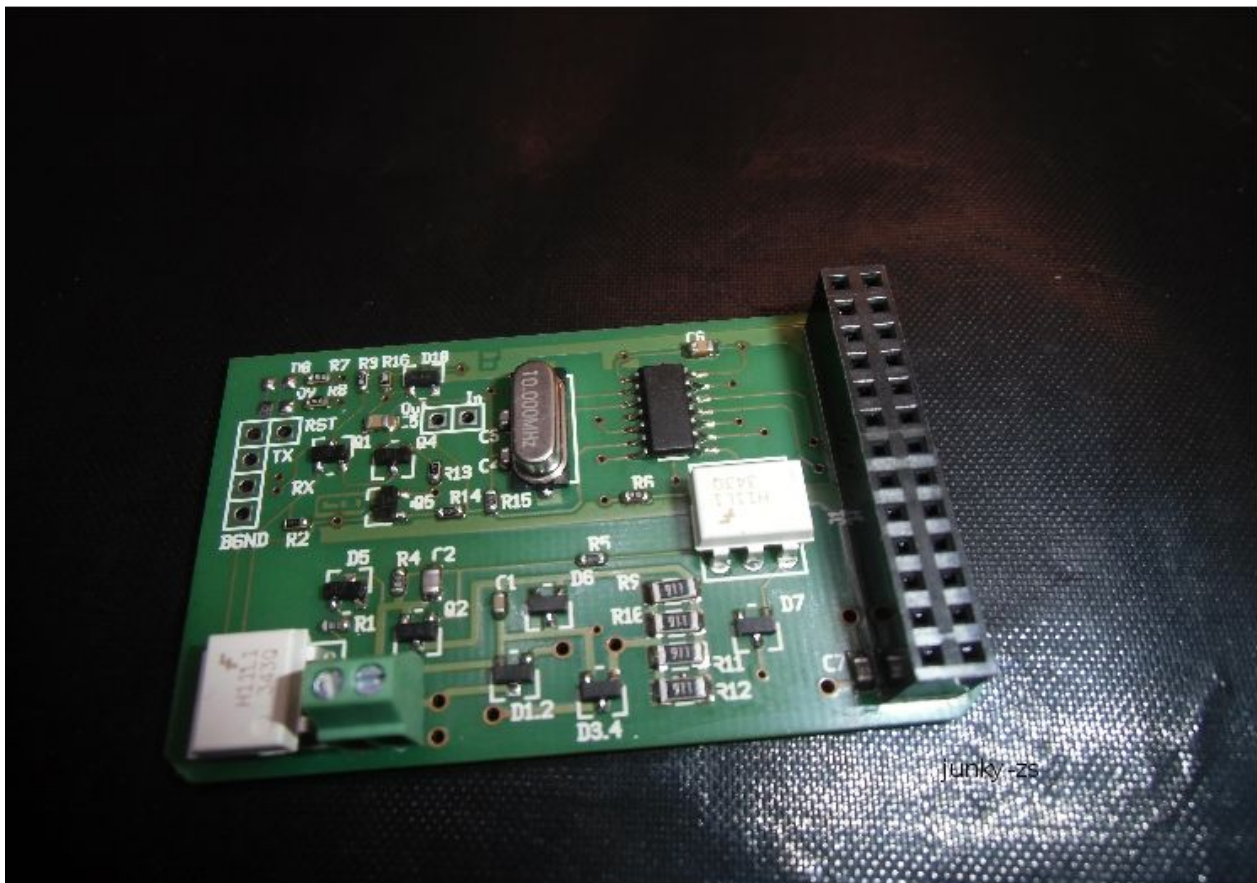


Abbildung 11: ht\_pitiny Adapter (bestückt)

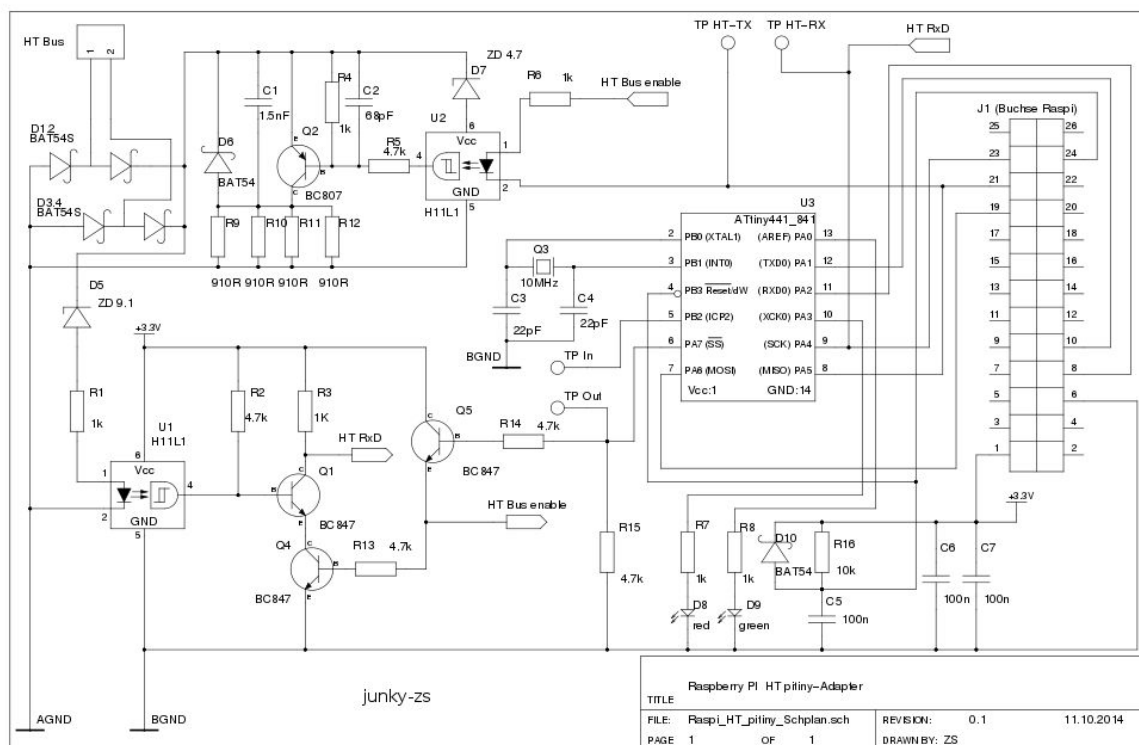


Abbildung 12: ht\_pitiny Schaltplan (Bestückung nur mit ATtiny841 !)

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: <a href="https://secure.reichelt.de/index.html?&amp;ACTION=20&amp;LA=5010&amp;AWKID=998314&amp;PROVID=2084">https://secure.reichelt.de/index.html?&amp;ACTION=20&amp;LA=5010&amp;AWKID=998314&amp;PROVID=2084</a>
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 4: Stückliste 'ht\_pitiny'-Adapter



### 1.3.3 USB-MotherBoard für Adapter 'ht\_transceiver' und UM2102

Die 'ht\_transceiver'-Adapter ('ht\_piduino' und 'ht\_pitiny') können auch ohne den RaspberryPi mit einem USB-ADAPTER (UM2102) betrieben werden. Dazu ist ein passives Motherboard realisiert, welches die Verbindungen zwischen 'ht\_transceiver', UM2102 und einem optionalen ISP-Anschluss zur Verfügung stellt. Der folgende Schaltplan zeigt, wie die Verbindungen zu realisieren sind.

Der UM2102 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).

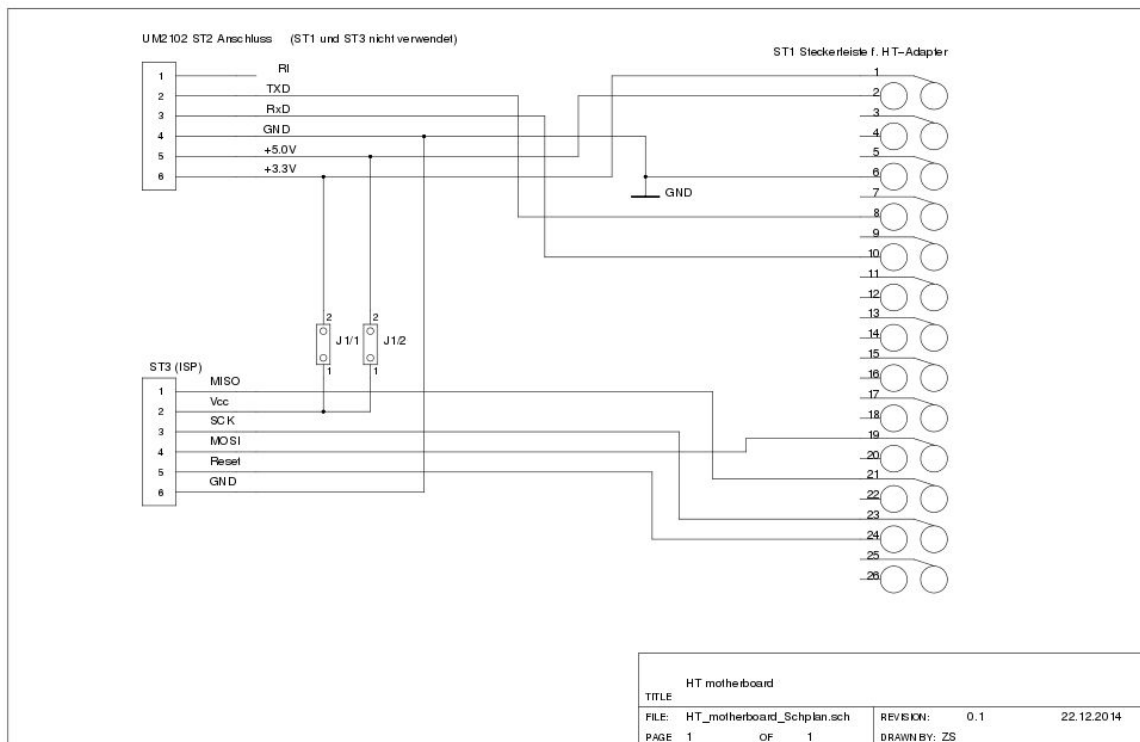


Abbildung 13: Motherboard für ht\_transceiver

## 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]).

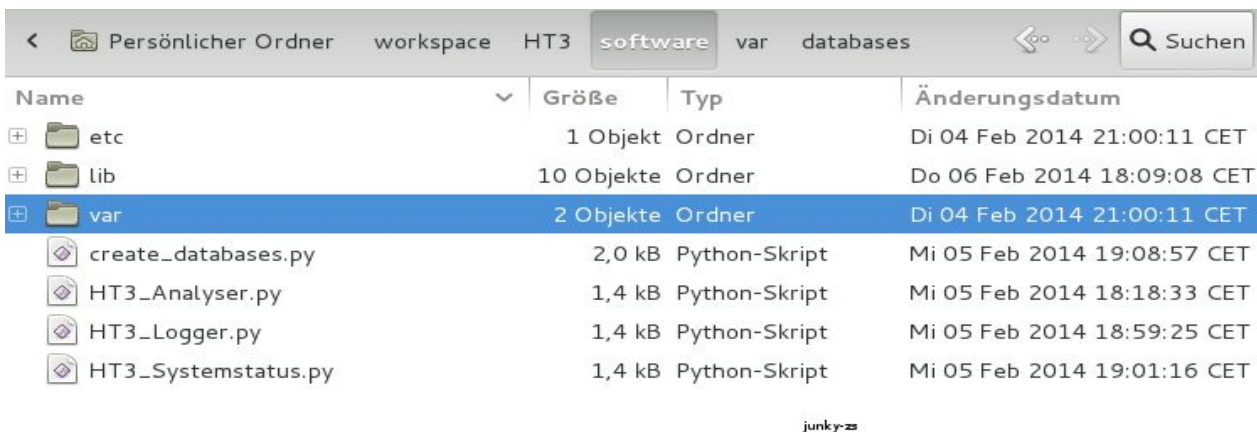
Für die Schnittstelle zur rrdtool-Datenbank wird die Programmiersprache 'Perl' genutzt, da die Python3-Module zur Zeit noch nicht verfügbar sind (Linux-Debian 'Wheezy') .

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:



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

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.
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_Logger.py	HT3 Logger schreibt die erfassten Daten in die Datenbanken, es werden diese nicht direkt grafisch angezeigt.	Der Logger dient zur Erfassung der Heizungsdaten ohne grafischer Datenausgabe. 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.

Tabelle 5: Software-Modulinformationen

## 2.2 Konfiguration

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



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.

Abbildung 16: Konfigurationsfile-Inhalt

```
.... <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.
<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.
<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.
<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.

Parameter	Werte	Funktion
<logging> <path>	./var/log	Relativer Pfad zum Log-Verzeichnis. Das Verzeichnis muss vorhanden und beschreibbar sein.
<logging> <default_filename>	ht_logger.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).
<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)
<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> System-konfigurationen 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 nach der Erzeugung und Nutzung der Datenbanken ist zu vermeiden und führt u.U. zu Datenverlust bzw. aufwendigen Anpassungen und Korrekturen.
<systempart name="heizkreisx"> <unmixed>	Flag (True/False)  wobei: x:= 1...4	Dieses Flag bestimmt, ob der Heizkreis keinen (True) oder einen Mischer (False) hat. Es wird nur die GUI-Anzeige damit gesteuert, die Erfassung wird dadurch nicht beeinflusst.
<systempart name="heizkreisx"> <buscodierung>	Wert je nach Heizkreis und Hersteller, Standardbereich 1...8.	Dieses Wert wird nur in der GUI-Anzeige verwendet, die Erfassung wird dadurch nicht beeinflusst. Dieser Wert ist vom System abhängig und kann der

Parameter	Werte	Funktion
	wobei: x:= 1...4	Anlagenkonfiguration entnommen werden.
<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.
<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 Zaehler 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> .

Tabelle 6: Konfigurations-Parameterbeschreibung

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 7: RRDTool Archiv-Details*

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



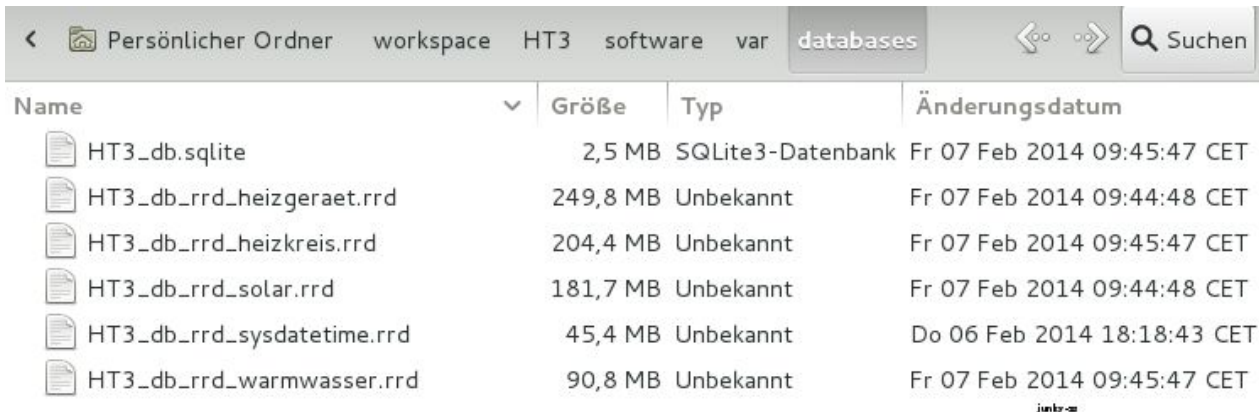
## 2.3 Datenbanken

Im Verzeichnis '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 anders als in der Abbildung mit '...rrd\_heizkreis1.rrd' bis '...rrd\_heizkreis4.rrd' erzeugt).

Die Verzeichnis-Struktur ist wie folgt:



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

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 z.Zeit nicht realisiert.

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.

### 2.3.1 Datenbank 'SQLite'

Die Datenbank 'sqlite' wird erzeugt, 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 'Zwischen'-Speicherung der dekodierten Datentelegramme benutzt. Intervall gesteuert werden die Daten von der SQL-Datenbank in die rrdtool-Datenbank übertragen.

Eine Auswertung der Daten wird z.Z. in dieser Datenbank nicht gemacht.

Geplant sind: Solarertrag pro Tag und Solarertrag-Gesamtsumme.

Beide Informationen sind nicht in den Datentelegrammen enthalten.

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

Die Tabellen-Namen entsprechen dabei dem 'systempart'-Namen aus der Konfiguration. Eine Ausnahme davon ist die Tabelle 'rrdtool\_infos', deren Aufgabe weiter unten beschrieben ist.

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.

The screenshot shows the SQLite Manager interface with the 'heizgeraet' table selected. The table structure is as follows:

ro...	Local_date_time	UTC	T_vorlauf_soll	T_vorlauf_ist	T_ruecklauf	T_mischer	V_modus	V_brenner_motor	V_h
26	2014.02.06 18:23:28	1391707408	37	23.7	23.2	39.2	1	0	1
27	2014.02.06 18:23:38	1391707418	37	23.5	23.2	39.2	1	0	1
28	2014.02.06 18:23:48	1391707428	37	23.7	23.2	39.2	1	0	1
29	2014.02.06 18:23:58	1391707438	37	23.5	23.2	39.2	1	0	1
30	2014.02.06 18:24:08	1391707448	37	23.7	23.2	39	1	0	1
31	2014.02.06 18:24:18	1391707458	37	23.7	23.2	39	1	0	1
32	2014.02.06 18:24:28	1391707468	37	23.5	23.2	39	1	0	1
33	2014.02.06 18:24:38	1391707478	37	23.5	23.2	39	1	0	1
34	2014.02.06 18:24:48	1391707488	37	23.7	23.2	39.2	1	0	1
35	2014.02.06 18:24:58	1391707498	37	23.5	23.2	39	1	0	1
36	2014.02.06 18:25:18	1391707518	37	23.5	23.2	39	1	0	1
37	2014.02.06 18:25:48	1391707548	37	23.5	23.2	39	1	0	1
38	2014.02.06 18:25:58	1391707558	37	23.5	23.2	39	1	0	1
39	2014.02.06 18:26:18	1391707578	37	23.5	23.2	39	1	0	1
40	2014.02.06 18:26:38	1391707598	37	23.5	23.2	38.9	1	0	1
41	2014.02.06 18:26:58	1391707618	37	23.5	23.2	38.9	1	0	1
42	2014.02.06 18:27:...	1391707627	37	23.5	23.2	38.9	1	0	1

Abbildung 18: SQL-Datenbanktabelle 'heizgeraet'

Der UTC-Zeitstempel wird für die interne Bestimmung von Zeitintervallen und als Zeit-Referenz für den intervall gesteuerten Eintrag in die rrdtool-Datenbank 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 'rrdtool\_infos' wird für die Synchronisation der SQLite-Datenbank mit der rrdtool-Datenbank verwendet.

Falls in der Konfiguration die rrdtool-Datenbank deaktiviert ist, wird diese Tabelle nicht erzeugt. Eine Kommunikation mit der rrdtool-Datenbank ist dann nicht möglich.

Das Bild zeigt die Tabelle 'rrdtool\_infos' und die zugehörigen Spalten.

rowid	Local_date_time	UTC	rrdtool_timestamp	comment	errors
3386	2014.02.07 08:27:48	1391758068	1391758065	SO:1391758030	None
3387	2014.02.07 08:27:48	1391758068	1391758065	HG:1391758010	None
3384	2014.02.07 08:27:47	1391758067	1391758065	WW:1391758010	None
3385	2014.02.07 08:27:47	1391758067	1391758065	HK:1391758061	None
3382	2014.02.07 08:26:48	1391758008	1391758005	SO:1391757968	None
3383	2014.02.07 08:26:48	1391758008	1391758005	HG:1391757950	None
3380	2014.02.07 08:26:47	1391758007	1391758005	WW:1391757951	None
3381	2014.02.07 08:26:47	1391758007	1391758005	HK:1391758001	None
3378	2014.02.07 08:25:48	1391757948	1391757945	SO:1391757907	None
3379	2014.02.07 08:25:48	1391757948	1391757945	HG:1391757901	None
3376	2014.02.07 08:25:47	1391757947	1391757945	WW:1391757901	None
3377	2014.02.07 08:25:47	1391757947	1391757945	HK:1391757940	None
3374	2014.02.07 08:24:48	1391757888	1391757885	SO:1391757846	None
3375	2014.02.07 08:24:48	1391757888	1391757885	HG:1391757830	None
3372	2014.02.07 08:24:47	1391757887	1391757885	WW:1391757841	None
3373	2014.02.07 08:24:47	1391757887	1391757885	HK:1391757879	None
3370	2014.02.07 08:23:48	1391757828	1391757825	SO:1391757784	None
3371	2014.02.07 08:23:48	1391757828	1391757825	HG:1391757771	None

Abbildung 19: Tabelle 'rrdtool\_infos'

Tabellen-Spalte	Bedeutung	Bemerkung
Local_date_time	Zeitpunkt (Local time) des Daten-'insert'.	Wert wird automatisch beim SQL-insert eingetragen.
UTC	Zeitpunkt (UTC-time) des Daten-'insert'.	Wert wird automatisch beim SQL-insert eingetragen.
rrdtool_timestamp	Oberer Zeitstempel (UTC-time) der Daten für die rrdtool-Datenbank. Diese Spalte sorgt dafür, das auch bei einem Neustart des Programms noch fehlende Daten-Übertragungen nachgeholt werden. Dazu wird der größte 'rrdtool_timestamp'-Wert verwendet um von dort an die Datensatz-Übertragung zu starten. Jede Übertragung zur rrdtool-Datenbank wird hier eingetragen, unabhängig davon ob ein Fehler aufgetreten ist oder nicht.	Es wird hier der obere Zeitwert (UTC) des gerade aktuellen Intervalls verwendet. Die an die rrdtool-db übertragenen Daten haben maximal diesen UTC-Zeitstempel. (Im Bild ist ein 60 Sekunden Intervall zu sehen)  Es wird nur der erste Eintrag eines <Systemparts> an die rrdtool-db übertragen, auch wenn mehrere gültige Einträge innerhalb eines Intervalls vorhanden sind.
comment	Text-String mit Informationen, die zu Debug-Zwecken dienen können.	Bedeutung der Informationen im Bild: <Nickname>:<UTC-Timestamp> Beispiel: SO:1391757784 SO := Systempart 'Solar' UTC:= Zeitwert an dem dieses Telegramm in die SQL-db eingetragen wurde.
errors	Fehlereintrag oder None	Falls bei der Datenübertragung in die rrdtool-db Fehler aufgetreten sind, wird dies hier eingetragen. Tritt kein Fehler auf, wird hier 'None' eingetragen. (Wenn ein Wert mit gleichem Zeitstempel erneut in die rrdtool-db eingetragen wird ist dies z.B. ein Fehler)

Tabelle 8: Beschreibung der SQLite Tabelle 'rrdtool\_infos'

## 2.3.2 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.

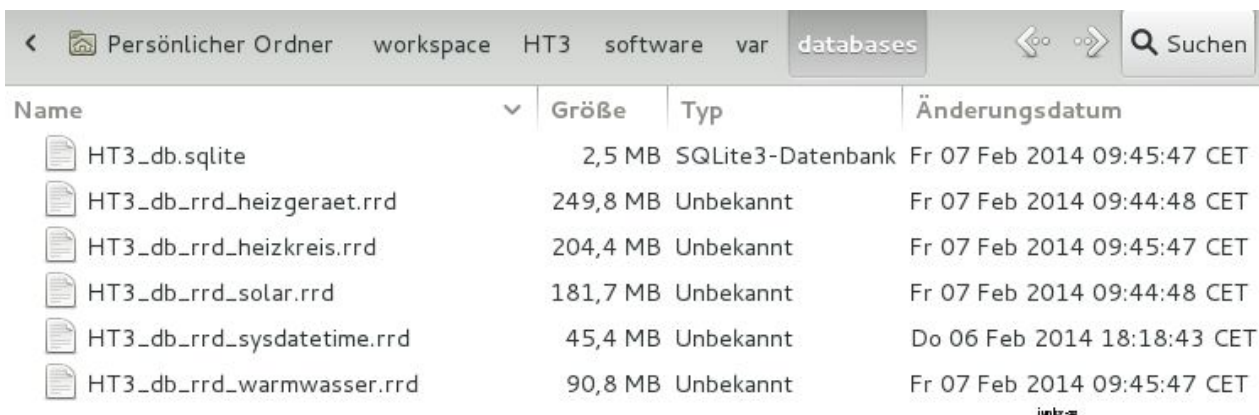
Für die Erzeugung der Datenbank und für die Daten-Aktualisierung werden perl-scripte dynamisch erzeugt und ausgeführt. Ebenso wird die Grafikerzeugung durch 'rrdgraph' mit einem perl-script realisiert.

Die Daten der SQLite-Datenbank werden in 60 Sekunden Intervallen in die rrdtool-Datenbank übertragen.

Es werden alle SQL-Datensätze aus den einzelnen Tabellen übertragen, mit Ausnahme von:

Tabelle : 'sysdatetime' und 'rrdtool\_infos'  
Tabellen-Spalten: 'Local\_date\_time', 'UTC' 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).



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 Einträge in der SQLite-Datentabelle 'rddtool\_info' sorgen für die korrekte zeitliche Übertragung in die rrdtool-Datenbank.

Siehe [2]

## 2.4 Applikationen

Alle realisierten Applikationen erfassen die HT3-Protokoll-Daten und tragen diese in die SQLite- und rrdtool-Datenbank ein. Unterschiede sind nur bei der grafischen Ausgabe vorhanden. Details im folgenden.

### 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 dem 'Nicknamen' der Systemkomponenten (HG,HK1,HK2,HK3,HK4,WW,SO,DT) und sind 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 Abbildung zeigt ein System mit einem Heizkreis und dem Heizgeräte-Type 'CSW'.

Abbildung 20: HT3 Analyser GUI

Wie die HT3-Analyser-GUI mit 3 Heizkreisen und dem Heizgeräte-Type 'KUB' aussehen

kann, zeigt das folgende Bild:

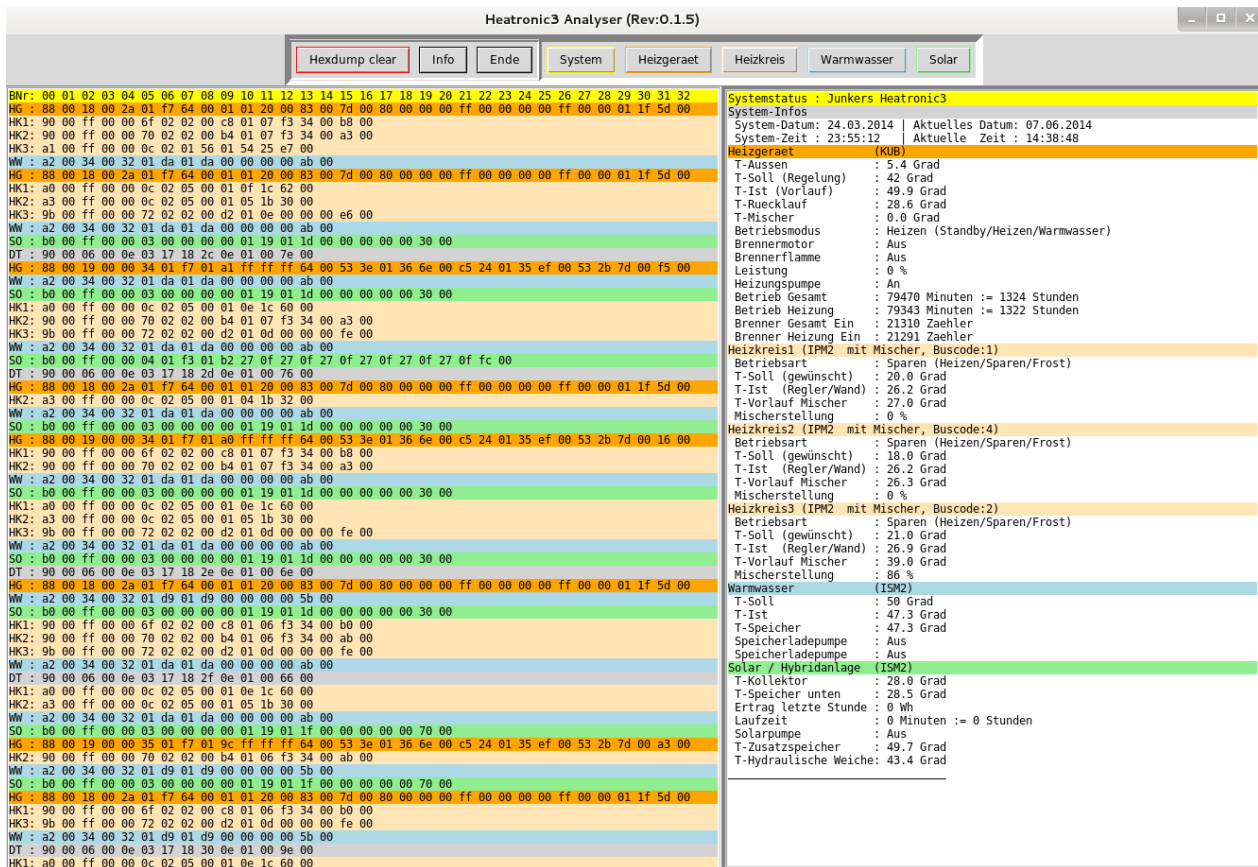


Abbildung 21: HT3 Analyser GUI (System mit 3 Heizkreisen)



## 2.4.2 HT3\_Systemstatus

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

**Heatronic3 Systemstatus (Rev:0.1.5)**

System   Heizgeraet   Heizkreis   Warmwasser   Solar   Ende

**Systemstatus : Junkers Heatronic3**

**System-Infos**

System-Datum: 07.06.2014 | Aktuelles Datum: 07.06.2014  
System-Zeit : 12:14:18 | Aktuelle Zeit : 12:13:17

**Heizgeraet (CSW)**

T-Aussen : 23.5 Grad  
T-Soll (Regelung) : 0 Grad  
T-Ist (Vorlauf) : 40.6 Grad  
T-Ruecklauf : 33.8 Grad  
T-Mischer : 40.8 Grad  
Betriebsmodus : Standby (Standby/Heizen/Warmwasser)  
Brennermotor : Aus  
Brennerflamme : Aus  
Leistung : 0 %  
Heizungspumpe : Aus  
Zirkulationspumpe : Aus  
Speicherladepumpe : An  
Betrieb Gesamt : 193201 Minuten := 3220 Stunden  
Betrieb Heizung : 175481 Minuten := 2924 Stunden  
Brenner Gesamt Ein : 6447 Zaehler  
Brenner Heizung Ein : 3237 Zaehler

**Heizkreis1 (CSW mit FB10 ohne Mischer, Buscode:1)**

Betriebsart : Heizen (Heizen/Sparen/Frost)  
T-Soll (gewünscht) : 21.5 Grad  
T-Ist (Regler/Wand) : 28.1 Grad  
T-Raum (FB10/FB100) : 22.2 Grad

**Warmwasser (CSW)**

T-Soll : 50 Grad  
T-Ist : 40.8 Grad  
T-Speicher : 47.0 Grad  
Betriebszeit : 17720 Minuten := 295 Stunden  
Brenner WarmW Ein : 3210 Zaehler  
Warmwasser-Erzeugung : Aus  
Speichertemperatur OK: Ja  
Nachladung : Aus

**Solar (ISM1)**

T-Kollektor : 241.6 Grad  
T-Speicher unten : 80.4 Grad  
Ertrag letzte Stunde : 916 Wh  
Laufzeit : 54928 Minuten := 915 Stunden  
Solarpumpe : Aus  
Kollektorfeld deaktiv: Ja  
Solarspeicher voll : Ja

### 2.4.3 HT3\_Logger

Der HT3 Logger erfasst die Protokoll-Daten und schreibt diese in die SQLite- und rrdtool-Datenbank. Eine grafische Ausgabe ist nicht vorhanden.

Der Betrieb des Loggers ist für Anwendungen ohne Grafikausgabe vorgesehen. Die Grafikausgabe wird indirekt über das rrdtool Datenbank-interface gemacht.

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

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

Die proxy-Software besteht aus den Anteilen: 1. proxy-server und 2. proxy-client(s). Diese erlaubt den Zugriff auf die Adapter-Hardware über Socket-Verbindungen und Telegrammen.

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 getesteter Standardwert.

### 2.5.1 ht\_proxy (proxy-server)

Der proxy-server 'ht\_proxy' stellt die Verbindung zwischen den Comports/ttyAMaX und der TCP/IP (Socket-Verbindung) her.

Da der 'ht\_proxy' direkt mit den comports/tty-Schnittstellen verbunden ist, muss dieser auch auf der Hardware installiert sein, die diese Schnittstellen 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 'insserv' in /etc/init.d installiert werden.
./HT3/sw/var/log/ht_proxy.log	Logfile des proxy-servers	Verzeichnis wird angelegt, falls nicht vorhanden. Daten stammen 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 'HT3\_Logger'-Client).

Die Nutzung des proxy-servers ist unabhängig von der verwendeten Erfassungshardware. Er kann mit 'HT3-miniadapter', 'HT3-microadapter' aber auch mit dem '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)	9600
HT3-MicroAdatper	/dev/ttyUSBx (PC / Laptop etc.)	9600
ht_transceiver (ht_piduino & ht_pitiny)	/dev/ttyAMA0 (RaspberryPi)	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 muss 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*'. 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

Folgende Steuer-Befehle sind mit diesem Client möglich:

Befehl	Parameter	Funktion	Bemerkung
ht_netclient.py -h	--	Hilfe-Funktion	Ausgabe der Hilfe-Funktion
ht_netclient.py -t 22.5	Temperatur (float)	Einstellung des Temperatur-Niveaus	Temperatur-Niveau bleibt auch nach einem Betriebsartwechsel erhalten. Ist z.Zeit nur für Betriebsart 'Heizen' realisiert.
ht_netclient.py -b WERT	WERTE: auto heizen sparen frost	Es wird die Betriebsart der Heizung auf den angegebenen Parameter eingestellt und die zugehörigen Temperatur-Niveaus verwendet.	Die jeweilige Betriebsart wird eingestellt und als NetCom (NC) Information am Bedienteil (Fwxyz) angezeigt.  Siehe Bilder.
ht_netclient.py -ht_cfg WERT	WERTE: 0 1 2 3	Betriebsmode des Adapters: 0 := Senden & Empfang aus. 1 := RX(Header), TX aus. 2 := RX(Raw), TX an. 3 := RX(Header), TX an.	Der kommandierte Wert wird erst nach einem Reset-Kommando an den 'ht_transceiver' aktiv.
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 eingestellter Betriebsmode bzw. eine 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.



Abbildung 23: NC "Auto"



Abbildung 24: NC "Heizen"



Abbildung 22: NC "Sparen"



Abbildung 26: NC "Frost"

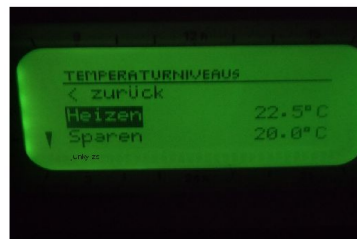


Abbildung 25: Temperatur-Niveaus

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 Wheezy'.

### 3.1 Betriebssystem

Aktualisierung des Betriebssystems 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
```

setuptools 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 <diaout> aufnehmen:

```
# sudo adduser 'username' dialout
```

Deaktivieren der default eingeschalteten TTY-Systemausgaben (RaspberryPI):

Datei: /boot/cmdline.txt anpassen

```
# sudo nano /boot/cmdline.txt
```

Zu editierende Zeile finden 'dwc\_otg.lpm\_enable=...' und anpassen:

```
von dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 ...  
in dwc_otg.lpm_enable=0 console=tty1 ...
```

Danach Datei speichern

Siehe auch [1]

### Anpassen der /etc/inittab (RaspberryPi):

Datei: /etc/inittab anpassen

```
# sudo nano /etc/inittab
```

Deaktivierung der Zeile durch Anpassung

von

```
#Spawn a getty on Raspberry Pi serial line
```

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

in

```
#Spawn a getty on Raspberry Pi serial line
```

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Danach Datei speichern

### Neustart des Raspberry Pi:

```
# reboot
```

## 3.2 Applikation

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

### 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' sollte so angelegt werden, da das Perl-Grafikscript so auf diese Struktur zugreift.

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

```
$ cd
```

```
$ git clone https://github.com/norberts1/hometop\_HT3.git
```

```
$ Eventuell zu ~/. verschieben
```

```
$ mv ~/hometop_HT3/HT3 ~/.
```

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

(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_cfg.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):

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

!! Achtung

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

## Aktivieren eines zentralen Cronjobs zur Grafikgenerierung (als root für user: **pi**):

(Konfiguriert für Default-Verzeichnisse: **./HT3/sw/var/databases** und **./HT3/sw/etc/html**)

```
# sudo cp ./HT3/sw/etc/rrdtool_draw.pl /usr/local/bin
# sudo chown pi /usr/local/bin/rrdtool_draw.pl
cronjob einrichten
# sudo nano /etc/crontab
Zeile hinzufügen
*/5 * * * * pi /usr/bin/perl -S rrdtool_draw.pl /home/pi /home/pi 1
(PParameter: 1. 2. 3.)
Datei speichern
```

Bedeutung:

Alle 5 Minuten ausführen als 'user' mit Perl das script=' rrdtool\_draw.pl' und Parametern:  
1. 'Datenbank-Verzeichnis'; 2. 'Zielverzeichnis'; 3. Anzahl Heizkreise.  
Die erzeugten Grafiken werden unter <Zielverzeichnis>/HT3/sw/etc/html abgelegt.

## Anpassung und Aktivieren des Startscripts 'ht3\_logger' (als root für user: **pi**):

Username und Verzeichnisse sind gegebenenfalls anzupassen

```
# sudo nano ./HT3/sw/etc/sysconfig/ht3_logger
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

Danach Datei kopieren:

```
# sudo cp ./HT3/sw/etc/sysconfig/ht3_logger /etc/init.d
```

Script aktivieren:

```
# cd /etc/init.d
# sudo insserv ht3_logger
```

## Anpassung und Aktivieren des Startscripts 'ht\_proxy' (als root für user:pi):

(Nur erforderlich, wenn man ht\_proxy - SERVER/CLIENT(s) verwenden will)

Username und Verzeichnisse sind gegebenenfalls anzupassen

```
# 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

Danach Datei kopieren:

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

Script aktivieren:

```
# cd /etc/init.d
# sudo inserv ht_proxy
```

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

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

Username und Verzeichnisse sind gegebenenfalls anzupassen

```
# 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

Danach Datei kopieren:

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

Script aktivieren:

```
# cd /etc/init.d
# sudo inserv httpd
```

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

Je nach Schnittstellen-Typ folgende Devices verwenden:

```
# deviceport="/dev/ttyAMA0" <!-- UART-Schnittstelle des Raspberry Pi
# 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> <!-- protocol-types are:
        RAW:=transparent ; TRX :=TBD (transceiver-messages with header) -->
<parameter name="ASYNC">
  <serialdevice>/dev/ttyAMA0</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> <!-- protocol-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>8088</portnumber>
  <logfilepath>./var/log/ht_proxy.log</logfilepath>
  <ht_transceiver_if devicename="RX">
    <parameter>
      <serialdevice>/dev/ttyAMA0</serialdevice>
      <baudrate>19200</baudrate>
      <config>"8N1"</config> <!-- only 8N1 available -->
    </parameter>
  </ht_transceiver_if>
</proxy_server>
```

....

(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>8088</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>8088</portnumber>
  <logfilepath>./var/log/ht_client_modem.log</logfilepath>
  <devicetype>MODEM</devicetype>
</proxy_client>
```

Neustart des Rechners (als root):

# reboot

Nach dem Neustart des Rechners muss der 'ht\_proxy.py'-Server (falls aktiviert) und die Applikation 'HT3\_Logger.py' automatisch gestartet worden sein.

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

Es werden dann alle 5 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:8086 und sobald PNG-Dateien erzeugt wurden (alle 5 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-Type kann direkt an einem ComPort (/dev/ttyAMA0) 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:

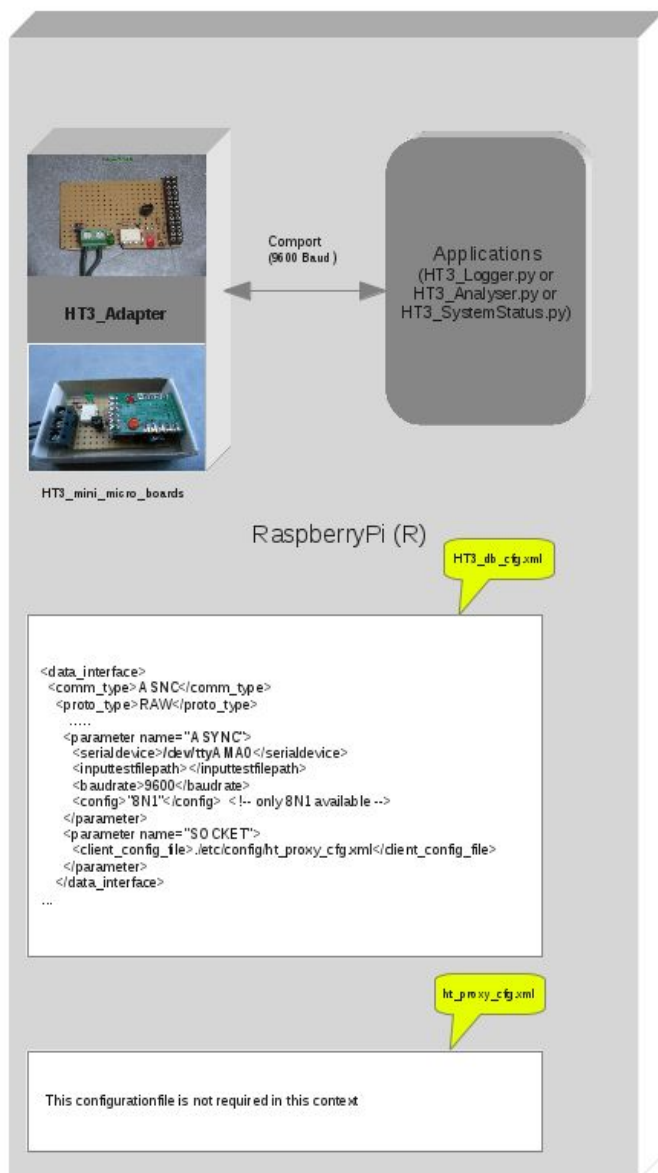
```
<data_interface>  
<comm_type>ASYNC</comm_type>
```

Jeder Adapter-Type 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>
```

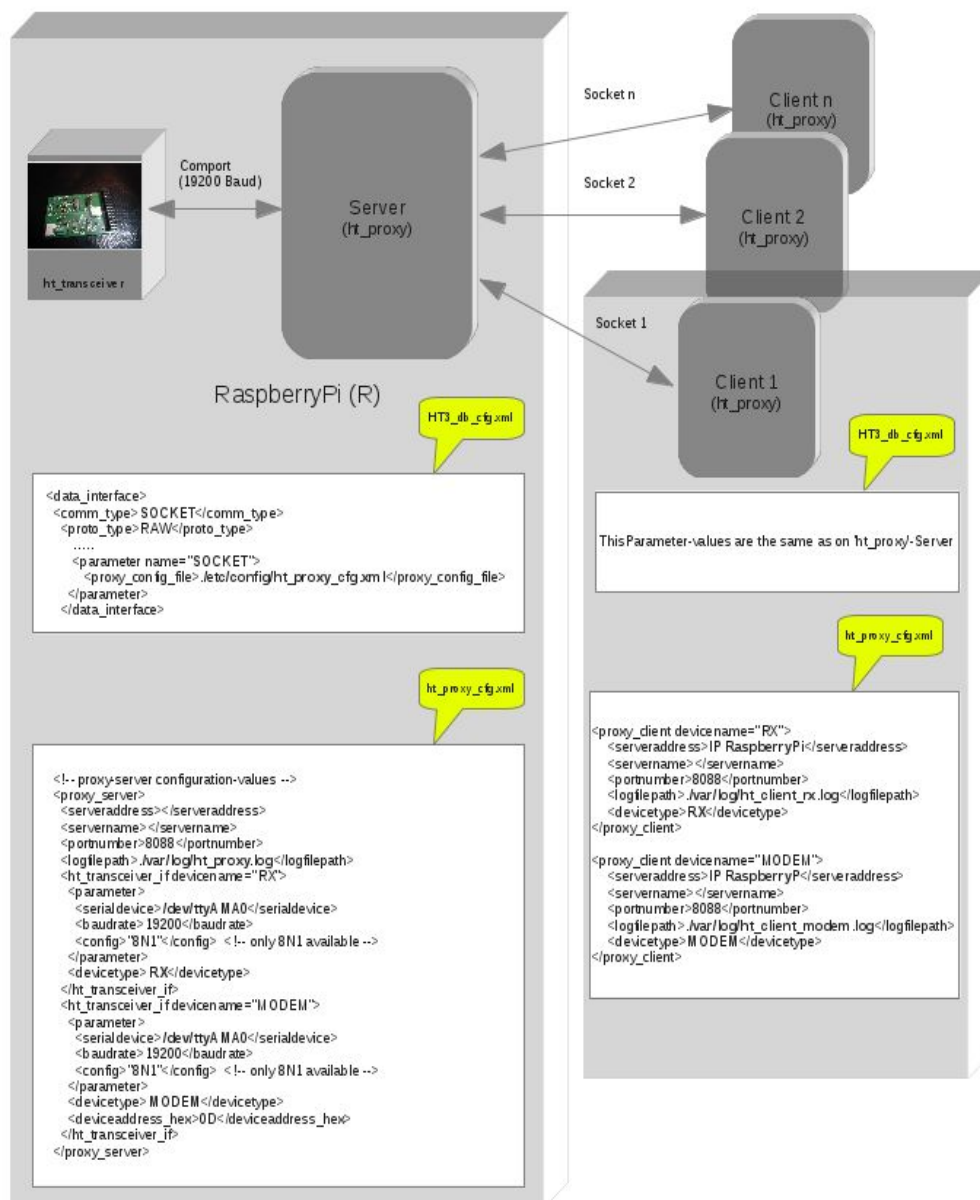
Zusätzlich müssen der ht\_proxy-server und der ht\_proxy-client eingestellt werden. Dies wird im File: ./HT3/sw/etc/config/ht\_proxy\_cfg.xml gemacht



Raspberry Pi (R) with 'HT3-mini- / HT3-micro-Adapter' and the required configuration

junkyzs@gmx.de

Abbildung 27: 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  $\leq 3$  on Raspberry Pi Rev: A/A+/B and B+.

junky-zs@gmx.de

Abbildung 28: ht\_transceiver mit ht\_proxy Konfiguration

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> <b>ASYNC</b> <baudrate> <b>9600</b> <serialdevice>/dev/ttyAMA0	--	Nein, Konflikt mit dem ComPort	Nein
HT3-microAdapter	File: HT3_db_cfg.xml Werte: <comm_type> <b>ASYNC</b> <baudrate> <b>9600</b> <serialdevice>/dev/ttyUSB(x)	--	Nein, Konflikt mit dem ComPort	Nein
ht_piduino ht_pitiny	File: HT3_db_cfg.xml Werte: <comm_type> <b>ASYNC</b> <baudrate> <b>19200</b> <serialdevice>/dev/ttyAMA0	--	Nein, Konflikt mit dem ComPort	Ja
HT3-miniAdapter	--	File: HT3_db_cfg.xml Werte: <comm_type> <b>SOCKET</b> <proxy_config_file> <b>Pfad auf File</b>  File: ht_proxy_cfg.xml <baudrate> <b>9600</b> <serialdevice>/dev/ttyAMA0	Ja, Proxy-Server muss installiert sein und laufen.	Nein
HT3-microAdapter	--	File: HT3_db_cfg.xml Werte: <comm_type> <b>SOCKET</b> <proxy_config_file> <b>Pfad auf File</b>  File: ht_proxy_cfg.xml <baudrate> <b>9600</b> <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> <b>SOCKET</b> <proxy_config_file> <b>Pfad auf File</b>  File: ht_proxy_cfg.xml <baudrate> <b>19200</b> <serialdevice>/dev/ttyAMA0	Ja, Proxy-Server muss installiert sein und laufen.	Ja

## 4 HT3 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 Anzeigintervall hier ist 2 Tage).

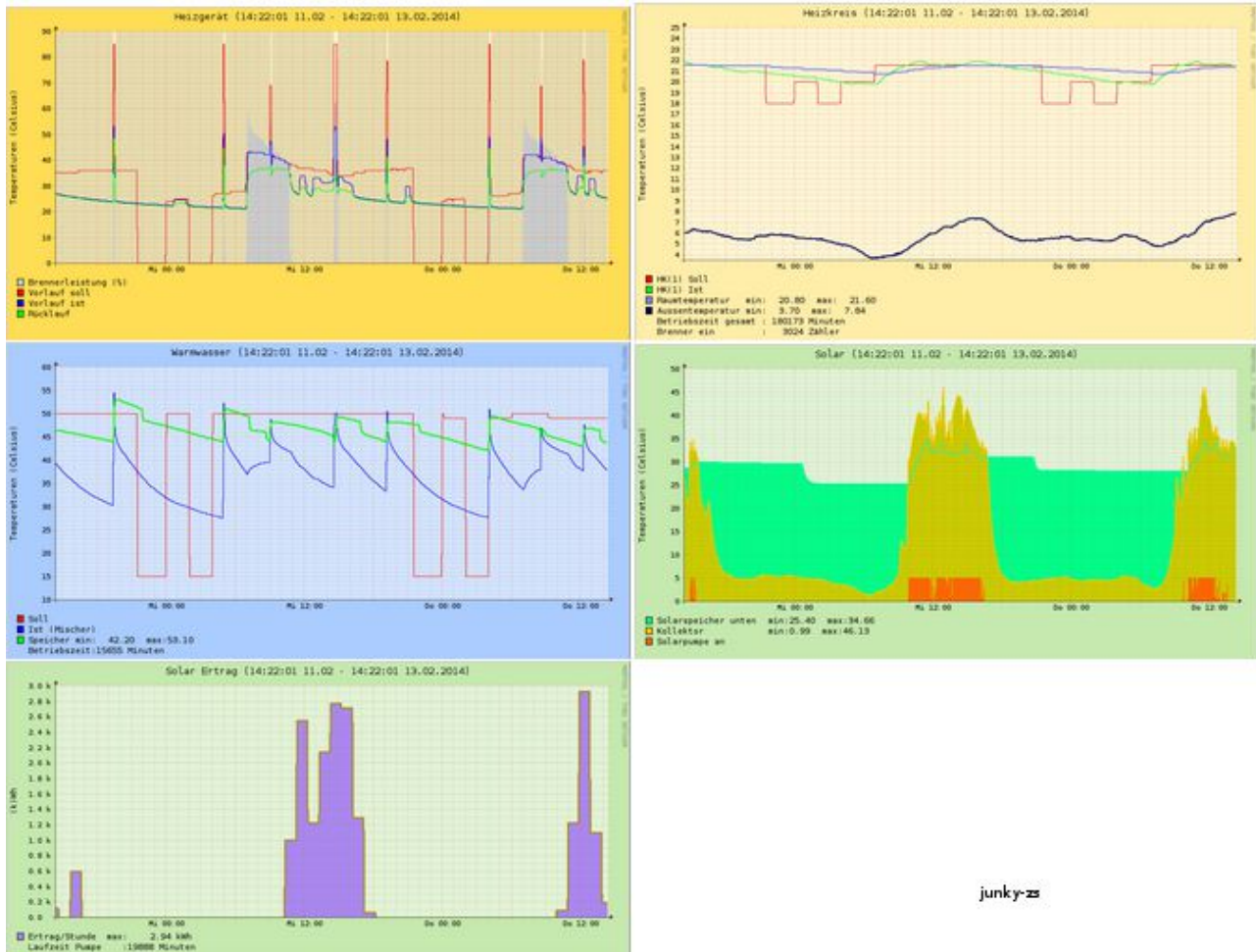


Abbildung 29: HT3 Systemhistorie im Browserfenster

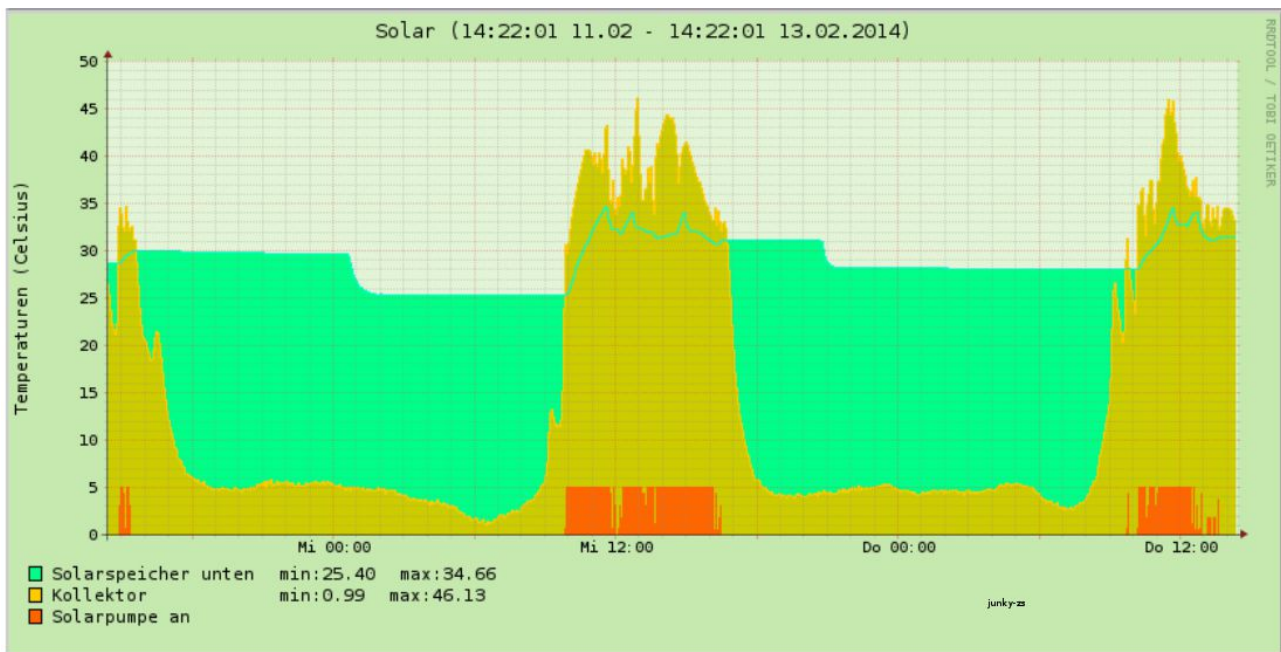


Abbildung 30: HT3 Solarhistorie im Browserfenster

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 Anzeigintervalls. 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 Michael Weigend	Verlag: mitp
Python 3	Das umfassende Handbuch Johannes Ernesti und Peter Kaiser	Verlag: Galileo Computing
[1]	<a href="http://kampus-elektroecke.de">http://kampus-elektroecke.de</a>	Elektronik, Code und mehr
[2]	<a href="http://oss.oetiker.ch/rrdtool">http://oss.oetiker.ch/rrdtool</a>	About RRDtool
[3]	<a href="http://www.mrtg.org/rrdtool/gallery/index.en.html">http://www.mrtg.org/rrdtool/gallery/index.en.html</a>	RRDtool Gallery
[4]	<a href="http://code.google.com/p/pyrrd">http://code.google.com/p/pyrrd</a>	A Pure-Python OO wrapper for RRDTool
[5]	<a href="http://www.mikrocontroller.net/forum/Haus &amp; Smart Home">http://www.mikrocontroller.net/forum/Haus &amp; Smart Home</a>	Forum: Haus & Smarthome
[6]	<a href="http://pi.gadgetoid.com/pinout/atmega328-arduino">http://pi.gadgetoid.com/pinout/atmega328-arduino</a>	Atmega328 over SPI
[7]	<a href="https://www.mikrocontroller.net/topic/317004#3925213">https://www.mikrocontroller.net/topic/317004#3925213</a>	Forum zum Thema 'ht_transceiver'
[8]	<a href="https://github.com/norberts1/hometop_HT3">https://github.com/norberts1/hometop_HT3</a>	Hometop HT3 Soft-/Hard-Ware.
[9]	<a href="https://github.com/norberts1/hometop_ht_transceiver">https://github.com/norberts1/hometop_ht_transceiver</a>	Hometop HT-Transceiver Soft-/Hard-Ware.