HANDBUCH

für den

BSB-LPB-LAN-Adapter v2 (Arduino-Version)

und die

BSB-LAN-Software

Stand: 21. September 2018

ACHTUNG:

Es gibt KEINE GARANTIE oder Gewährleistung jeglicher Art, dass dieser Adapter dein Heizungssystem NICHT beschädigt!

Jegliche Umsetzung der hier beschriebenen Schritte, jeder Nachbau des Adapters sowie jede Verwendung der beschriebenen Hard- und Software erfolgt auf eigene Verantwortung und eigenes Risiko!

Keiner der Mitwirkenden oder Autoren kann für etwaige Schäden jeglicher Art haftbar gemacht werden!

Dieses Handbuch wurde geschrieben, um den Einstieg in die Benutzung des BSB-LPB-LAN-Adapters und der dazugehörigen Software zu vereinfachen und um als Nachschlagewerk zu dienen.

Es wird empfohlen, dieses Handbuch vor einer initialen Verwendung des BSB-LPB-LAN-Adapters komplett zu lesen.

Autoren:

- Software, Schaltplan v1, Dokumentationen EN, Ideenfindung, Support bis v0.16: *Gero Schumacher (gero.schumacher [ät] gmail.com)*
- Software, Platinenlayout v1 & v2, Dokumentationen EN, Ideenfindung, Support ab v0.17: *Frederik Holst (bsb [ät] code-it.de)*
- Debugging, Handbuch, Übersetzung EN-DE, Ideenfindung, Support ab v0.17: *Ulf Dieckmann (adapter [ät] quantentunnel.de)*

Basierend auf dem Code und der Mitarbeit von vielen anderen Entwicklern! Vielen Dank!

Inhaltsverzeichnis

1. Der BSB-LPB-LAN-Adapter und die BSB-LAN-Software

Der BSB-LPB-LAN-Adapter und die dazugehörige BSB-LAN-Software wurden entwickelt, um eine Anbindung von Heizungssystemen bzw. -reglern ans LAN und somit auch einen entsprechenden Fernzugriff zu ermöglichen.
Darüber hinaus ist es u.a. möglich, Parameter (bspw. Laufzeiten, Temperaturen) auf eine microSD-Karte zu loggen und sowohl DS18B20- als auch DHT22-Sensoren zusätzlich am Adapter anzuschließen.
Die hier vorgestellte Lösung stellt eine Alternative zu den bisherigen kommerziellen Lösungen dar, die nicht nur hinsichtlich des Kostenfaktors, sondern auch in Bezug auf die vielfältigen Einsatzmöglichkeiten mehr als nur ‚interessant' ist.

Für den Einsatz des Adapters müssen die Kessel-, Solar- oder auch Wärmepumpen-Regler[1] einen BSB[2] oder LPB[3] aufweisen! Dies ist i.d.R. bei (aktuelleren[4]) Reglern der Fall, die von der Firma SIEMENS hergestellt wurden. Solche Regler werden bspw. von Heizungsherstellern wie Brötje oder Elco verbaut. ***Ob dein Heizungssystem über einen solchen Bus verfügt, ist den spezifischen technischen Unterlagen zu entnehmen.*** Diesbezügliche Anfragen können von den Autoren nicht beantwortet werden.

Ältere Heizungsregler der o.g. Hersteller weisen u.U. eine sog. PPS-Schnittstelle auf, meist in Verbindung mit einer QAA70-Bedieneinheit. Der BSB-LPB-LAN-Adapter ist auch hier einsetzbar, allerdings ist der Funktionsumfang der BSB-LAN-Software hersteller- und reglerbedingt in diesem Fall sehr eingeschränkt.

Die hier vorgestellte und beschriebene Kombination aus Hard- und Software wurde u.a. an verschiedenen Heizungssystemen von Brötje und Elco ausgiebig getestet, der Einsatz an Heizungssystemen weiterer Hersteller war ebenfalls erfolgreich. Die Kommunikation sollte prinzipiell mit allen Systemen möglich sein, die einen der o.g. Bus-Typen aufweisen. Eine Auflistung der bisher erfolgreich getesteten Systeme ist u.a. im nachfolgenden Kapitel zu finden, dennoch kann niemals ein voller Funktionsumfang garantiert werden. Bei nicht gelisteten Systemen ist u.U. ein erhöhter eigener Einsatz nötig, um die Software in vollem Umfang nutzen zu können (siehe u.a. Kapitel 10).

Die Software ist auf einem Arduino Mega2560 samt Ethernet-Shield des Typs W5100[5] lauffähig, erfahrungsgemäß ‚out of the box'. Aufgrund des geringeren Speichers ist die Verwendung von bspw. Arduino UNO, Arduino Nano o.ä. nicht möglich.
Grundsätzlich ist die Verwendung von original Arduino-Komponenten zu empfehlen, da es bei dem Einsatz von günstigen Clones u.U. zu diffusen Problemen kommen kann.

Da es teilweise unterschiedliche Pinbelegungen bei den verschiedenen LAN-Shield-Clones gibt, ist es u.U. nötig, den BSB-LPB-LAN-Adapter an andere Pins anzuschließen und die entsprechenden Änderungen hinsichtlich der Pinbelegung in der Software (Datei *BSB_lan_config.h*) anzupassen (siehe Kapitel 4 & 5).

Der Schaltplan des Adapters ist im Anhang zu finden. Außerdem wurden in der Vergangenheit Sammelbestellungen für fertige Platinen durchgeführt, bei Interesse kann man sich im Forum[6] melden oder direkt Frederik Holst (bsb [ät] code-it.de) kontaktieren.

1 Im Folgenden nur als „Regler" bezeichnet.
2 BSB = Boiler System Bus
3 LPB = Local Process Bus
4 Ausnahme: Siehe Kap. 3.3 Hinweis: Neue Modellgeneration - NICHT unterstützter Regler von Brötje
5 Der Chip-Typ W5100 wird ohne Probleme unterstützt und wurde ausgiebig getestet, er ist einem W5500 daher immer vorzuziehen. Bzgl. Verwendung eines W5500-Boards siehe Kapitel 5 & 15.11.
6 https://forum.fhem.de/index.php/topic,29762.0.html

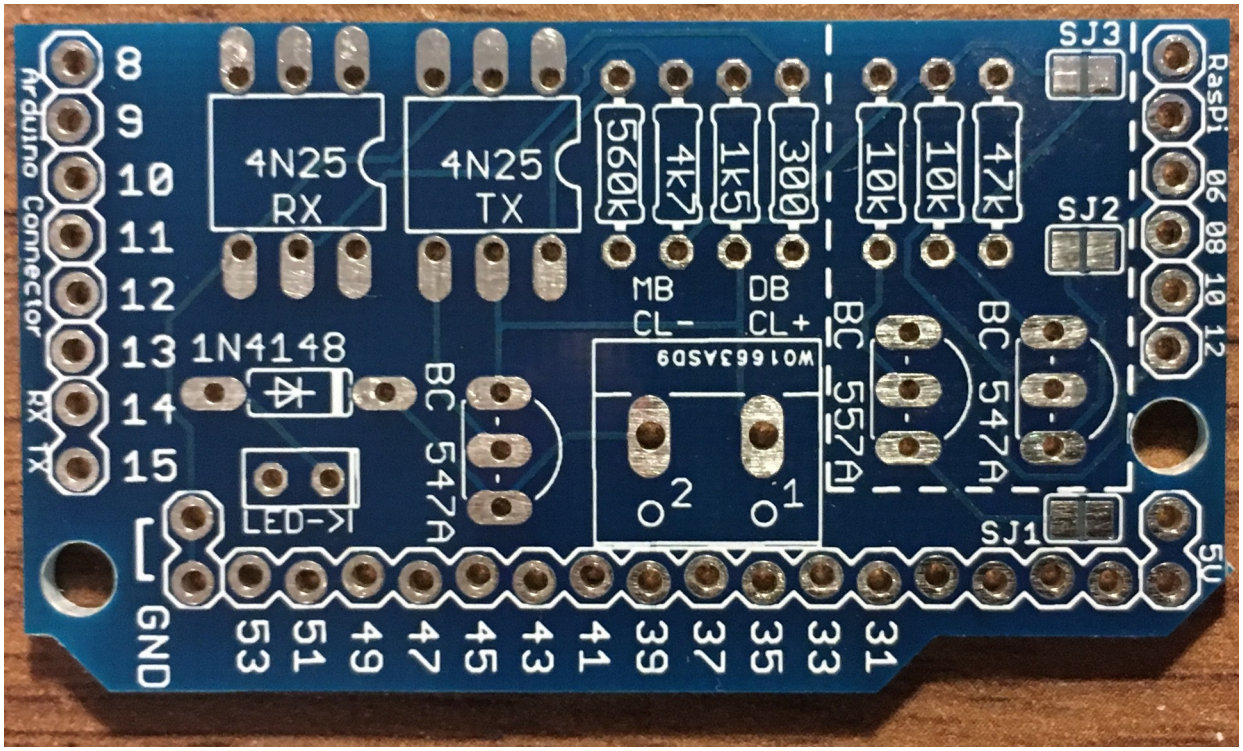


Abbildung 1: BSB-LPB-LAN-Adapter v2, unbestückt

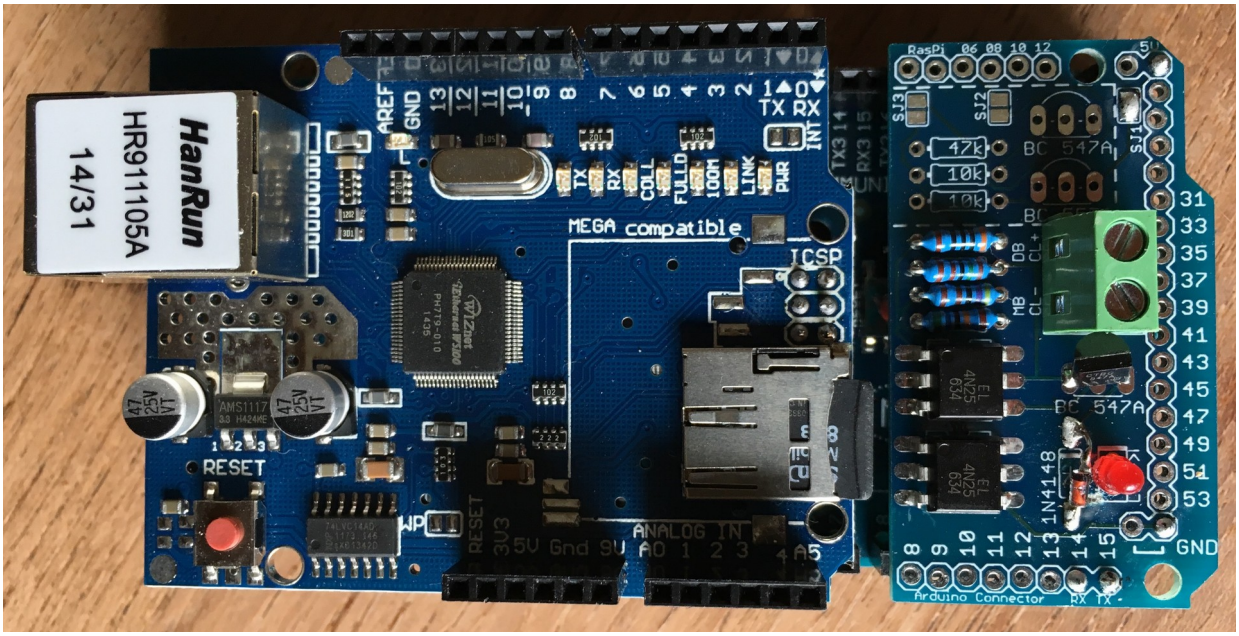


Abbildung 2: BSB-LPB-LAN-Adapter v2, bestückt, auf einem Arduino Mega2560 samt Ethernet-Shield

Hinweis:

Der aktuelle Adapter (v2) kann auch an einem Raspberry Pi 2 genutzt werden, jedoch nur unter Verwendung einer gänzlich anderen Software („bsb_gateway“). Diese kann unter folgender URL gefunden werden:

https://github.com/loehnertj/bsbgateway

Für jeglichen Support in Zusammenhang mit der bsb_gateway-Software kontaktiere bitte direkt den Autor von bsb_gateway.

Hier kann kein Support für bsb_gateway gegeben werden!

Alle Informationen in diesem Handbuch beziehen sich nur auf die Arduino-Version!

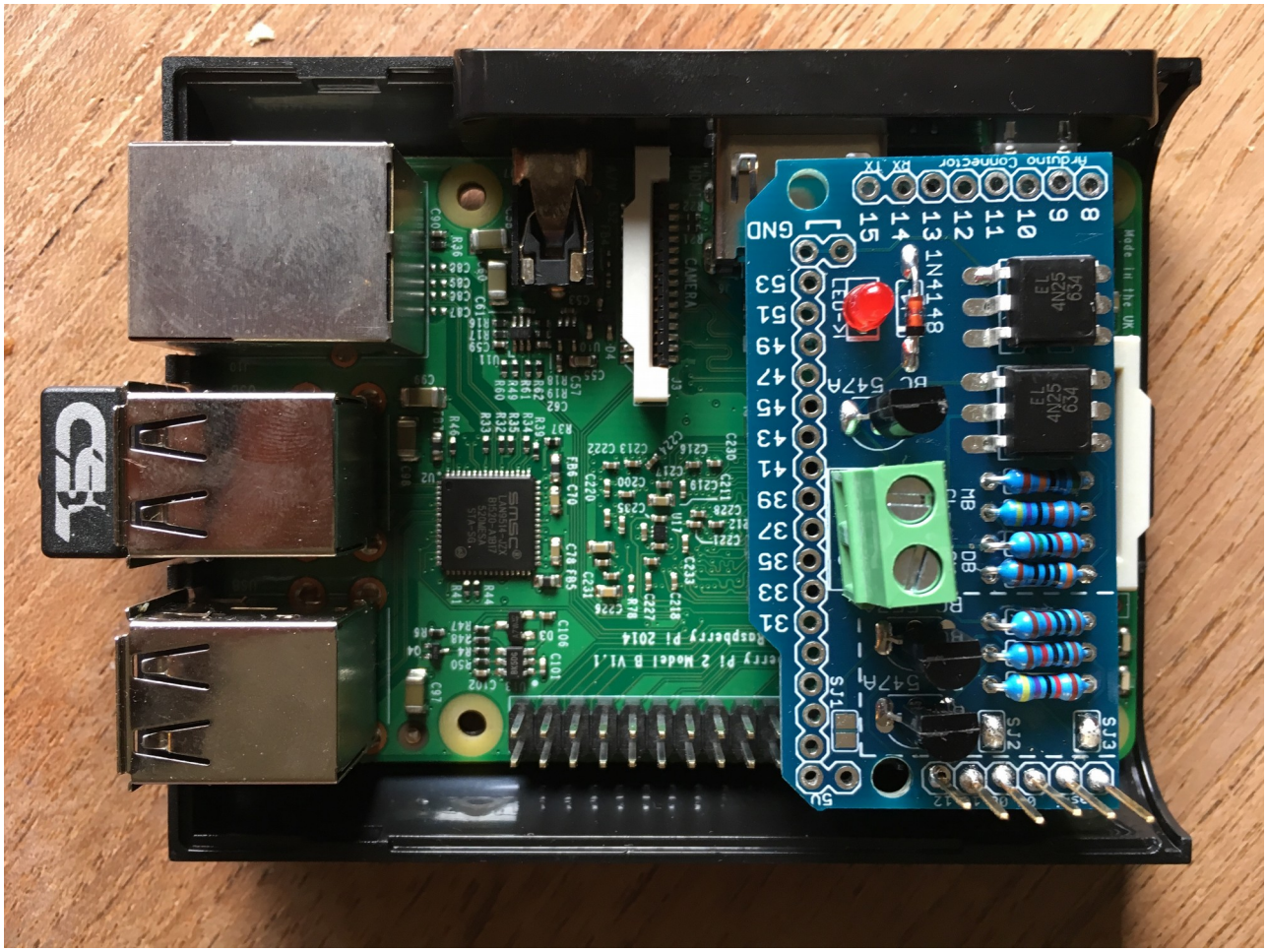


Abbildung 3: BSB-LPB-LAN-Adapter v2, bestückt, auf einem Raspberry Pi 2

2. Grundsätzliches zum BSB, LPB und zur PPS-Schnittstelle

2.1 BSB und LPB

BSB (Boiler System Bus) und LPB (Local Process Bus) sind zwei verschiedene Bus-Typen, die sich vereinfacht in zwei Nutzungsklassen unterscheiden lassen:

1. BSB ist ein ‚lokaler' Bus zur Nutzung des lokal angeschlossenen Reglers.
Angeschlossene Geräte wie bspw. die Bedieneinheit, ein Raumgerät / Fernbedienung oder auch der (via BSB) angeschlossene Adapter können nur auf den Regler zugreifen, an dem sie angeschlossen sind.

2. LPB ist ein ‚übergreifender' Bus zur Nutzung mehrerer angeschlossener Regler in einem komunikationsfähigen Verbund.
Über den LPB können mehrere Regler miteinander verbunden werden und bei korrekter Parametrierung (Stichwort Geräte- und Segmentadressen) gewisse Werte/Einstellungen/Parameter miteinander teilen bzw. sich gegenseitig beeinflussen.
Auf diese Weise kann bspw. eine Kaskadenschaltung von mehreren Brennern realisiert werden oder eine Gas- oder Öl-Heizung mit einer Solaranlage und einem Feststoffkessel regelungstechnisch 'verbunden' werden.
Der korrekte Anschluss der einzelnen Komponenten sowie die korrekte Parametrierung der jeweiligen Regler sollte im Normalfall bereits bei der Installation der Anlage durch den Heizungsinstallateur erfolgt sein.

Beispiel:

Vorhanden sind eine Öl- oder Gasheizung, ein nachgerüsteter wasserführender Kamin und eine thermische Solaranlage zur Unterstützung des Heizkreises oder der Warmwasserbereitung.
Alle drei Wärmeerzeuger sind hydraulisch an einem Pufferspeicher angeschlossen.
Die Wärme für den Heizkreis soll vom Pufferspeicher bezogen werden.

Die Regelung der Solaranlage und des Feststoffkessels übernimmt ein Solarsystemregler (SSR), die Kesselsteuerung der Heizung übernimmt in diesem Beispiel der interne Heizungsregler. Alle Sensoren, Pumpen, Mischer etc. sind am SSR angeschlossen, welcher jedoch via LPB mit dem Heizungsregler verbunden ist. Durch diese Verbindung der beiden Regler kann somit bspw. eine Pufferspeicherladung geregelt werden, bei der die Heizung nur aktiv wird, wenn weder Solar noch Feststoffkessel den Puffer laden / geladen haben.

Wenn ein Adapter via BSB an einem der beiden Regler aus oben genanntem Beispiel angeschlossen ist, kann er folglich nur auf den jeweiligen Regler 'lokal' zugreifen, an dem er angeschlossen ist (also bspw. Heizungsregler oder SSR). Ebenso verhält es sich mit den jeweiligen Bedieneinheiten der Regler, die über den ‚Bus BE' (Bus Bedieneinheit) angeschlossen sind.
Wenn ein Adapter via LPB an einem der beiden Regler aus oben genanntem Beispiel angeschlossen ist, müssen
1. die Geräte- und Segmentadressen entsprechend der LPB-Konfigurationsanforderungen eingestellt werden, und
2. beim Adapter eine Zieladresse eingestellt werden, an die die jeweiligen Anfragen des Adapters geschickt werden.

Die spezifischen technischen Daten, Leistungsmerkmale und Anforderungen an entsprechende Installationen und Parametrierungen hinsichtlich der Geräte- und Segmentadressen sind den jeweiligen technischen Dokumentationen der Hersteller zu entnehmen. Hinsichtlich des LPB seien insbesondere die Dokumentationen „LPB Systemgrundlagen“[7] und „LPB Projektierungsgrundlagen“[8] empfohlen.

Bei einigen Reglern sind die entsprechenden Anschlüsse teilweise unterschiedlich gekennzeichnet:

- Der BSB ist nicht auf allen Reglern als solcher bezeichnet, weitere Bezeichnungen sind „FB“ (Fernbedienung) sowie „CL+“ und „CL-“.

  Der zusätzliche Anschluss „G+“ führt 12V und ist für die Hintergrundbeleuchtung der entsprechenden Raumgeräte vorgesehen. Dieser ist für den Anschluss des Adapters NICHT zu verwenden!
  (Sollte der Adapter irrtümlicherweise an G+ statt an CL+ angeschlossen werden, so leuchtet zwar die LED, allerdings ist keinerlei Funktion gegeben.)

- Der LPB ist bei einigen Reglern mit „DB“(+) und „MB“(-) gekennzeichnet.

Die folgenden Abbildungen zeigen exemplarisch die verschiedenen Anschlüsse.

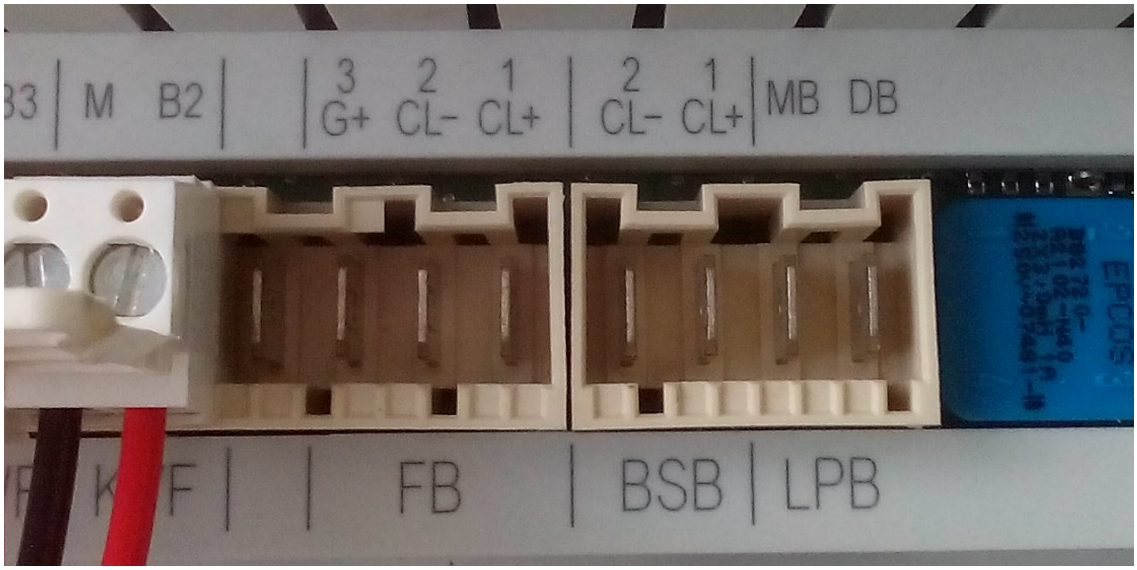


Abbildung 4: BSB & FB (CL+/CL-) und LPB (DB/MB) bei einem Brötje ISR-RVS43.222-Regler

7 Siemens Building Technologies - Landis & Staefa Division: CE1N2030D
8 Siemens Building Technologies - Landis & Staefa Division: CE1N2032D

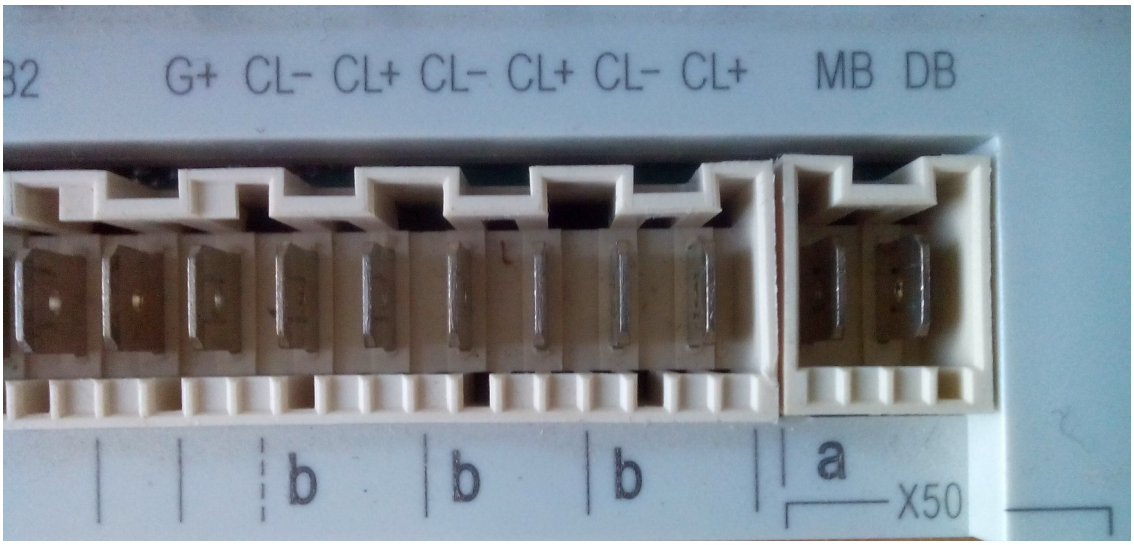


Abbildung 5: b = BSB (CL+/CL-) und a = LPB (DB/MB) bei einem Siemens RVS63.283-Regler

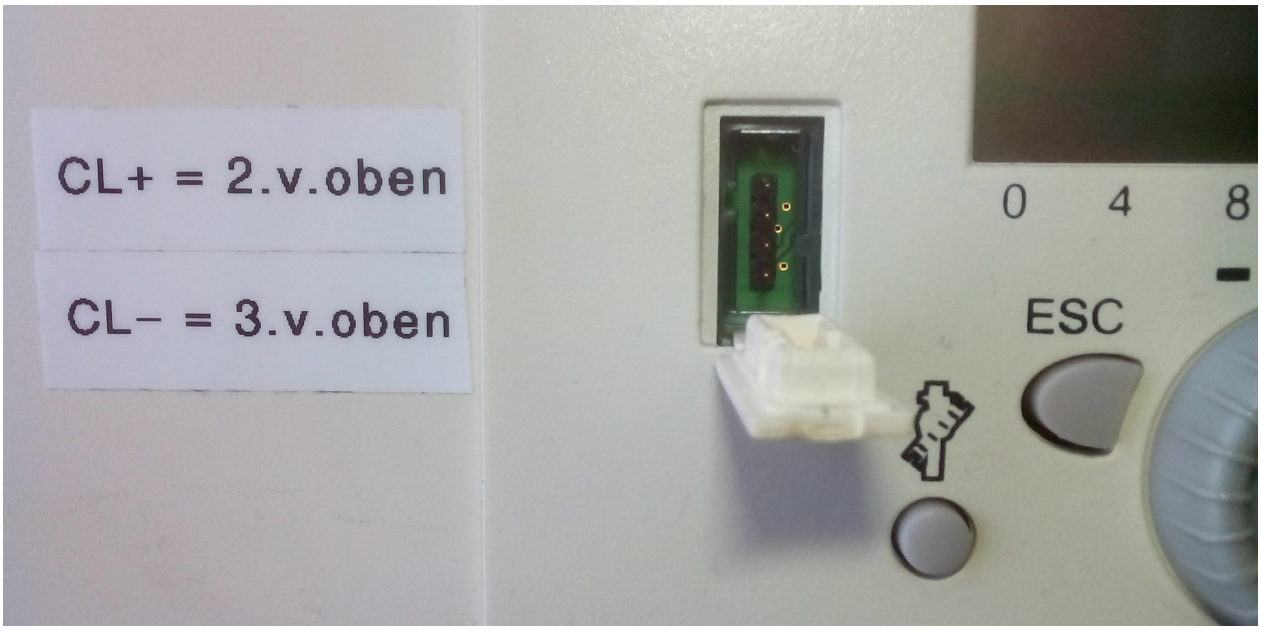


Abbildung 6: BSB (CL+/CL-) an der vierpoligen Servicebuchse vorne in der Bedieneinheit eines ISR Plus
→ Die (dauerhafte) Verwendung dieses Anschlusses ist jedoch nicht zu empfehlen.

Bei Anschluss des Adapters sollte der betreffende Regler stets ausgeschaltet sein, ebenso bei einem Entfernen des Adapters.

Es ist unbedingt darauf zu achten, dass der Regler polrichtig angeschlossen wird! Ein verkehrter Anschluss kann eine Beschädigung des Reglers und/oder Adapters zur Folge haben!

Die entsprechende Polung bzw. Bezeichnung der Anschlüsse ist auf der Adapterplatine gekennzeichnet. Bei einem Nachbau ist der Schaltplan zu beachten.

Wenn mehrere Regler (bspw. wie im obigen Beispiel) vorhanden sind, bietet es sich derzeit noch an, pro Regler jeweils einen Adapter via BSB anzuschließen, um den

jeweiligen Zugriff zu realisieren. Eine übergreifende Abfrage von Werten oder Parametern zweier oder mehrerer Regler im LPB-Verbund via Adapter kann mittlerweile zwar erfolgen, doch ist diese Funktion noch nicht ausgiebig getestet worden. Um mit einem Adapter via LPB auf verschiedene Regler zuzugreifen, ist die jeweilige Angabe der reglerspezifischen Geräteadresse als Zieladresse nötig (siehe Kapitel 8. URL-Befehle und Spezialfunktionen). Alle Geräte (Regler und Adapter) müssen sich dabei im selben Segment befinden und grundsätzlich gemäß den LPB-Projektierungsgrundlagen konfiguriert sein.

Gewisse Funktionen scheinen im Übrigen nicht via LPB unterstützt zu werden, bspw. das Senden einer Raumtemperatur an einen Regler, da diese Information vom Regler auf dem BSB erwartet wird. Da die Software ursprünglich für die Nutzung des Adapters via BSB geschrieben und erst nachträglich von Frederik um die LPB-Funktionalität erweitert wurde, ist es außerdem möglich, dass via LPB nicht alle Parameter verfügbar sind, die via BSB verfügbar wären. Die Software wird zwar stetig weiter entwickelt, derzeit bietet es sich jedoch noch an, als Anschluss den BSB zu präferieren, wenn dieser vorhanden ist.

Tipps:

Um vor Störeinflüssen möglichst geschützt zu sein, sollten die Anschlusskabel für den LPB-Anschluss gemäß LPB-Projektierungsgrundlagen[9] einen Querschnitt von 1,5mm² aufweisen, zweiadrig verdrillt und geschirmt sein (Leitungslänge max. 250m pro Busteilnehmer, max. Gesamtlänge 1000m).
Für den BSB-Anschluss sind Cu-Leitungen mit mindestens 0,8mm² (bis 20m) Querschnitt zu wählen, bspw. LIYY oder LiYCY 2 x 0,8. Bei Leitungslängen bis 80m sollte 1mm², bis 120m sollten 1,5mm² Querschnitt gewählt werden[10].
Generell ist eine parallele Verlegung mit Netzleitungen zu vermeiden (Störsignale), geschirmte Leitungen sind ungeschirmten Leitungen immer vorzuziehen.

Der Anschluss der Leitungen an die jeweiligen Kontakte sollte prinzipiell immer mit den spezifischen Steckern erfolgen.
Sollten diese nicht unmittelbar erhältlich oder verfügbar sein, können im Ausnahmefall (vorübergehend) auch isolierte 6,3mm-Kabelschuhe verwendet werden.

9 Siemens Building Technologies - Landis & Staefa Division: CE1N2032D
10 Siehe „Brötje Montageanleitung für Raumgerät RGT/RGTK“

2.2 PPS-Schnittstelle

Die PPS-Schnittstelle findet sich bei älteren Reglern und stellt eine Punkt-zu-Punkt-Schnittstelle dar, mittels derer Bedieneinheiten/Raumgeräte wie das QAA70[11] angeschlossen werden können. An demjenigen Anschluss wird analog zum QAA auch der Adapter angeschlossen.

PPS scheint bei folgenden Reglern zum Einsatz gekommen zu sein[12]: RVP digital Serie D, RVP54…, ALBATROS RVA…, LGM11…; bzw. u.a. bei folgenden Heizungen: Brötje WGB 15 / WGB 20, Weishaupt WRD 0.2 / 1.1, Sieger TG11 (mit Siegermatic S42DB), Olymp THR 5-25C, Schäfer Interdomo (mit DomoCommand DC 225).

Die beiden Geräte (Raumgerät und Regler) sprechen nur bedingt miteinander. Der Regler sendet Infos, schickt dann später mit einem einzigen Byte (0x17) eine Anforderung an das Raumgerät, das dann teilweise auf vorhergehende Regler-Infos reagiert, andererseits aber auch nach eigenem Rhythmus seine Infos sendet. Und das teilweise in unterschiedlicher Häufigkeit.
Der Bus kommt so kaum zur Ruhe, i.d.R. werden bis zu zwei Telegramme pro Sekunde ausgetauscht, entsprechend schnell muss die Software dann auch antworten. Kommt auf bestimmte Anfragen des Reglers keine oder nicht die richtige Antwort, wird angenommen, dass es kein Raumgerät mehr gibt und der Regler verfällt wieder in einen Suchmodus.

Der Funktionsumfang ist hierbei nur rudimentär und beschränkt sich derzeit mittels BSB-LAN auf etwa ein Dutzend Parameter, die man lesen/schreiben kann:

Raumtemperatur Ist
Raumtemperatur Soll
Außentemperatur (read-only)
Außentemperatur gemischt (read-only)
Position Drehknopf
Kesselvorlauftemperatur (read-only)
Mischervorlauftemperatur (read-only)
Status Trinkwasserbetrieb (read-only)
Trinkwassertemperatur Ist (read-only)
Trinkwassertemperatur Soll
Betriebsart
Anwesenheit

Immerhin lassen sich damit aber die wichtigsten Funktionen einer intelligenten Heizungssteuerung umsetzen, indem man z.B. gewichtete Raumtemperaturen sendet und die Solltemperaturen nach vielfältigeren Kriterien steuern kann.

Hinweis:
Noch offen ist, ob wegen des Punkt-zu-Punkt-Designs ein QAA mit dem Adapter friedlich koexistieren kann oder ob man sich am Ende für eine Variante entscheiden muss.

11 Eventuell auch das QAA50, dies konnte bisher jedoch noch nicht verifiziert werden.
12 Siehe „Siemens Raumgerät QAA70 Basisdokumentation“, CE1P1638D

3. Unterstützte Heizungssysteme und Regler

Prinzipiell unterstützt BSB-LAN von der Firma Siemens hergestellte (Heizungs-)Regler, die einen BSB und/oder LPB aufweisen. Diese werden von verschiedenen Heizungsherstellern ‚gebrandet' und verbaut.

ACHTUNG:
Aus aktuellem Anlass sei hier darauf hingewiesen, dass die Heizungshersteller offensichtlich eine neue Heizungs- und Reglergeneration auf den Markt gebracht haben, die nach bisherigem Wissensstand NICHT kompatibel mit BSB-LAN ist! Siehe hierzu Kapitel 3.3 Hinweis: Neue Modellgeneration - NICHT unterstützter Regler von Brötje.

3.1 Erfolgreich getestete Heizungssysteme

Im Folgenden findest du eine Auflistung derjenigen Heizungssysteme und Regler, bei denen von einem erfolgreichen Einsatz des Adapters berichtet wurde. Da jedoch nicht jeder sein Heizungssystem, den verbauten Regler und die Ausgabe der Parameterabfrage `http://<IP-Adresse>/6220-6228` meldet, ist davon auszugehen, dass in der Praxis noch weitere Systeme erfolgreich mit dem BSB-LPB-LAN-Adapter und der BSB-LAN-Software betrieben werden (können).

Bei den (aktuelleren) Reglern kann i.d.R. grundsätzlich zwischen zwei Reglertypen unterschieden werden: Regler des Typs RVS und LMx. Diese zwei Typen unterscheiden sich u.a. in der Anschlussvielfalt und einigen verfügbaren Parametern. Weitere Angaben sind im entsprechenden Kapitel zu finden.

Hinweis und Bitte:
Um die Liste vervollständigen zu können und anderen möglichen Nutzern den Einstieg zu erleichtern, sei hier nochmals ausdrücklich auf die Bitte hingewiesen, ein noch nicht aufgeführtes Heizungssystem und/oder einen noch nicht aufgeführten Regler unter Angabe der genauen Herstellerbezeichnung sowie der Ausgabe der Parameterabfrage `http://<IP-Adresse>/6220-6228` (via Adapter!) und der verwendeten Anschlussart (BSB/LPB/PPS) zu melden.
Danke!

Beispielausgabe einer solchen Abfrage bei einer „Brötje NovoCondens SOB 26C":

```
6220 Konfiguration - Software- Version: 1.3
6221 Konfiguration - Entwicklungs-Index: error 7 (parameter not supported)
6222 Konfiguration - Gerätebetriebsstunden: 12345 h
6223 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014
6224 Konfiguration - Geräte-Identifikation: RVS43.222/100
6225 Konfiguration - Gerätefamilie: 96
6226 Konfiguration - Gerätevariante: 100
6227 Konfiguration - Objektverzeichnis-Version: 1.0
6228 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014
```

3.1.1 Brötje

- Brötje BBK 22E [LMS14] (Gasbrenner)
- Brötje BBS Pro Evo 15C [LMU74] (Gasbrenner)
- Brötje EcoCondens BBS 15E [LMS14] (Gasbrenner)
- Brötje EcoCondens BBS 20E [LMS14] (Gasbrenner)
- Brötje EcoCondens BBS EVO 20 G [LMS15] (Gasbrenner)
- Brötje EcoTherm Kompakt WMS 12 [LMS 15] (Gasbrenner)
- Brötje EcoTherm Kompakt WMS 24 [LMS 15] (Gasbrenner)
- Brötje ISR-SSR [RVS63.283] (Solarsystemregler)
- Brötje ISR-ZR1, ZR2 [RVS46.530] (Zonenregler)
- Brötje LogoBloc Unit L-UB 25C [RVS43.122] (Ölbrenner)
- Brötje NovoCondens BOB 20 [RVS43.325] (Ölbrenner)
- Brötje NovoCondens SOB 26 [RVA63.242] (Ölbrenner)
- Brötje NovoCondens SOB 22C [RVS43.222] (Ölbrenner)
- Brötje NovoCondens SOB 26C [RVS43.222] (Ölbrenner) + EWM [RVS75.390]
- Brötje NovoCondens WOB 20D [RVS43.325] (Ölbrenner)
- Brötje SensoTherm BLW 12 B [RVS21.825] (Wärmepumpe)
- Brötje SensoTherm BLW 15 B [RVS21.825] (Wärmepumpe)
- Brötje SensoTherm BSW-K [RVS61.843] (Wärmepumpe)
- Brötje TrioCondens BGB 20E [LMS14] (Gasbrenner)
- Brötje WBS 14D [LMU74] (Gasbrenner)
- Brötje WBS 22E [LMS14] (Gasbrenner)
- Brötje WGB 15 E [LMS14] (Gasbrenner)
- Brötje WGB 20C [LMU74] (Gasbrenner)
- Brötje WGB EVO 20H [LMS15] (Gasbrenner)
- Brötje WGB Pro EVO 20C [LMU75] (Gasbrenner)
- Brötje WGB S 17/20 E EcoTherm Plus [LMS14] (Gasbrenner)
- Brötje WGB-U 15H [LMS14] (Gasbrenner)

ACHTUNG:
Die neuen Modellreihen Brötje WLS/WLC und BOK sind NICHT mit BSB-LAN kompatibel!

3.1.2 Elco

- Elco Aerotop G07-14 [RVS61.843] (Wärmepumpe)
- Elco Aerotop T10C [RVS61.843] (Wärmepumpe)
- Elco Aquatop 8es [RVS51.843] (entspricht CTA Optihead OH1-8es) (Wärmepumpe)
- Elco Straton 21 [RVS63.283] (Ölbrenner)
- Elco Thision S Plus 13 [LMS14] (Gasbrenner)
- Elco Thision S13.1 E [LMU7x] (Gasbrenner)
- Elco Thision S17.1 [LMU74] (Gasbrenner)
- Elco Thision S17.1 [RVS63.283] (Gasbrenner)
- Elco Thision S25.1 [RSV63.283] (Gasbrenner) + MM [AVS75.390]

3.1.3 Weitere Hersteller

- Atlantic Alféa Extensa + [RVS21.831] (Wärmepumpe)
- Baxi Luna Platinum+ [LMS15] (Gasbrenner)
- CTC 380 IC [RVS43.143] (Ölbrenner)
- Fujitsu Waterstage WSYK 160 DC 9 [RVS21.827] (Wärmepumpe)
- Fujitsu Waterstage WSYP 100 DG 6 [RVS21.831] (Wärmepumpe)
- Sieger TG11 [RVP54.100] (Ölbrenner)
- Weishaupt WTU-25 G mit WRS-CPU B2/E [RVS23.220] (Ölbrenner)

3.2 Detailliertere Auflistung und Beschreibung der unterstützten Regler

Die folgende Reglerauflistung und -beschreibung soll u.a. einen kurzen Überblick über die bereits von BSB-LAN unterstützten Geräte und deren rudimentären Unterschiede geben. Auf die unterschiedliche reglerspezifische Verfügbarkeit von speziellen Parametern wird nicht weiter eingegangen. Es sei jedoch darauf hingewiesen, dass mittels BSB-LAN grundsätzlich etliche Parameter verfügbar sind, die mittels integrierter Bedieneinheit nicht verfügbar sind.

Die folgende Übersicht beinhaltet Angaben der Konfigurationsparameter 6220-6228. Diese Parameter können i.d.R. nur via Adapter abgefragt werden!

Eine Ausnahme bei der nachfolgenden Auflistung der Reglertypen stellt das Modell AVS75.390 dar. Hierbei handelt es sich um ein Erweiterungsmodul (EWM) für den RVS-Reglertyp, an dem sich weitere Fühler und Verbraucher anschließen und somit im System integrieren lassen. Pro RVS-Regler können bis zu zwei EWM angeschlossen werden. Die Parametrierung und Einbindung erfolgt über den eigentlichen RVS-Regler bzw. die eigentliche Bedieneinheit des RVS-Reglers.

Die Verbindung zwischen RVS-Regler und EWM erfolgt über den Anschluss ‚Bus EM', das EWM selbst hat keinen zusätzlichen LPB- oder BSB-Anschluss.

Der Zugriff auf ein EWM ist somit nur indirekt über die jeweils spezifischen Parameter möglich, die die Einstellungen und Funktionen des EWMs definieren und beschreiben.

Tabelle 1: Erweiterungsmodul AVS75.390

Geräte-Identifikation [/6224]		**Geräte-Familie [/6225]**	**Geräte-Variante [/6226]**	**Obj.Verz.-Version [/6227]**	**Software-Version [/6220]**	**Nutzbarer Bustyp**		
Reglertyp	**Kundenvariante**					**BSB**	**LPB**	**PPS**
AVS75.390	/100 (Brötje) /109 (Siemens) /260 (Elco)	-	-	-	-	-	-	-

Hinweis:
Im Folgenden nicht aufgeführte Regler bitte wie in Kapitel 10. Auslesen neuer Parameter-Telegramme beschrieben auslesen und melden. Danke!

3.2.1 LMx-Regler

Im Folgenden werden die Regler des Typs LMS und LMU aufgeführt. Diese sind erfahrungsgemäß bei Gasheizungen/-thermen verbaut.

Regler der Serie LMS scheinen die Nachfolger der LMU-Serie und somit die aktuelle Reglergeneration zu sein.
Der (Funktions-)Unterschied zwischen dem LMS14 und dem LMS15[13] scheint in der „Sitherm Pro“-Anwendung zur Optimierung des gesamten Verbrennungsprozesses zu liegen, die anscheinend nur die LMS15-Regler aufweisen.

Der LMx-Reglertyp weist i.d.R. nur einen BSB-Anschluss auf, LPB muss bei Bedarf mittels eines ClipIn-Moduls (OCI420) nachgerüstet werden.
Als Bedieneinheit kommt i.d.R. eine Variante des Siemens AVS37.294 zum Einsatz (Bezeichnung bspw. „ISR Plus“ bei Brötje).

Tabelle 2: LMS- und LMU-Regler[14]

Geräte-Identifikation [/6224]		Geräte-Familie [/6225]	Geräte-Variante [/6226]	Obj.Verz.-Version [/6227]	Software-Version [/6220]	Nutzbarer Bustyp		
Reglertyp	**Kundenvariante**					**BSB**	**LPB**	**PPS**
LMS14	.001A100 (Brötje) .001A100 (Brötje) .001A100 (Brötje) .001A100 (Brötje) .001B100 (Brötje) .002A100 (Brötje)	162 162 162 162 ? 162	14 16 16 17 ? 5	0.1 0.2 0.3 0.2 ? 1.7	2.3 2.3 2.3 2.3 ? 3.5	✓	-	-
	.002A167 (Elco)	203	?	?	?	✓	-	-
LMS15	.000A349 (Brötje) .000A349 (Baxi) .001A100 (Brötje)	123 163 163	1 ? 16	0.1 ? 0.6	4.2 ? 3.8	✓	-	-
LMU74	? (Brötje) ? (Brötje) ? (Elco) .100A136 (Elco)	97 97 97 97	100 100 136 136	2.5 2.9 0.2 0.4	11.0 11.0 11.1 11.2	✓	-	-
LMU75	? (Brötje)	98	?	?	?	✓	-	-

13 Der LMS15 scheint noch Parameter aufzuweisen, die bisher noch nicht implementiert wurden.
14 Bei ?-Einträgen liegen derzeit keine detaillierteren Angaben der/des jeweiligen Nutzer/s vor.

3.2.2 RVx-Regler

Im Folgenden werden die Regler des Typs RVA, RVP und RVS (aktueller Reglertyp) aufgeführt. Diese scheinen i.d.R. bei Ölheizungen, Wärmepumpen und verschiedenen ‚alleinstehenden' Reglern (Zonenregler, Solarsystemregler) zum Einsatz zu kommen.

3.2.2.1 RVA- und RVP-Regler

Regler des Typs RVA gehören anscheinend zur vorherigen Reglergeneration und weisen je nach Modell nur einen LPB- und/oder PPS-Anschluss auf (keinen BSB).
Als Bedieneinheit ist meist eine Variante der Eurocontrol-Steuerung (Brötje) verbaut.
Eine Bedienung mittels BSB-LAN ist nur in deutlich geringerem Umfang als bei der aktuellen Reglergeneration RVS möglich.

Regler des Typs RVP scheinen noch älter als RVA-Regler zu sein und weisen lediglich eine PPS-Schnittstelle auf.
Der Bedienungsumfang mittels BSB-LAN ist bei diesem Reglertyp nur in sehr eingeschränktem Umfang möglich.

Tabelle 3: RVA- und RVP-Regler (alte Reglergenerationen)[15]

Geräte-Identifikation [/6224]		Geräte-Familie [/6225]	Geräte-Variante [/6226]	Obj.Verz.-Version [/6227]	Software-Version [/6220]	Nutzbarer Bustyp		
Reglertyp	**Kundenvariante**					**BSB**	**LPB**	**PPS**
RVA63.242	? (Brötje) ? (Brötje)	28 28	100 109	302.0 302.0	2.5 3.6	-	✓	-
RVP54.100	? (Sieger)	?	?	?	?	-	-	✓

15 Bei ?-Einträgen liegen derzeit keine detaillierteren Angaben der/des jeweiligen Nutzer/s vor.

3.2.2.1 RVS-Regler

Regler des Typs RVS scheinen die ‚aktuelle' Reglergeneration darzustellen und werden i.d.R. von BSB-LAN vollständig unterstützt[16]. Sie weisen häufig sowohl einen LPB-, als auch mehrere BSB-Anschlüsse auf.
Eine Ausnahme scheinen die Regler der Reihe RVS21 zu sein, die bei Fujitsu-Wärmepumpen zum Einsatz kommen: Diese scheinen nur einen BSB aufzuweisen.
Bei Gasgeräten kommen RVS-Regler anscheinend nur zum Einsatz, wenn bspw. ein Solarthermie-Paket mitinstalliert wurde (Beispiel bei einer „Elco Thision S17.1": LMU74-Regler bei Installation ohne Solarthermie, RVS63.283-Regler bei Installation mit Solarthermie).
Als Bedieneinheit kommt dabei i.d.R. eine Variante des Siemens AVS37.294 zum Einsatz (Bezeichnung bspw. „ISR Plus" bei Brötje).

Tabelle 4: RVS-Regler (aktuelle Reglergeneration)[17]

Geräte-Identifikation [/6224]		Geräte-Familie [/6225]	Geräte-Variante [/6226]	Obj.Verz.-Version [/6227]	Software-Version [/6220]	Nutzbarer Bustyp		
Reglertyp	**Kundenvariante**					**BSB**	**LPB**	**PPS**
RVS21.825	E/100 (Brötje)	205	100	301.2	7.2	✓	-	-
RVS21.827	/127 (Fujitsu)	170	?	?	?	✓	-	-
RVS21.831	F/127 (Fujitsu) F/127 (Atlantic Fujitsu)	211 211	127 127	402.1 402.4	8.3 8.5	✓	-	-
RVS23.220	/320 (Weishaupt)	50	?	?	?	-	✓	-
RVS43.122	? (Brötje)	95	?	?	?	✓	✓	-
RVS43.143	/110 (CTC)	103	110	100.5	3.4	✓	✓	-
RVS43.222	/100 (Brötje)	96	100	1.0	1.3	✓	✓	-
RVS43.325	/100 (Brötje)	138	100	100.0	2.3	✓	✓	-
RVS46.530	/100 (Brötje)	107	100	100.5	3.4	-	✓	-
RVS51.843	/169 (Elco)	85	?	?	?	✓	✓	-
RVS61.843	E/100 (Brötje) E/160 (Elco) E/560 (Elco)	108 108 108	100 160 217	301.0 2.5 301.2	7.2 2.7 7.4	✓	✓	-
RVS63.283	/100 (Brötje) /109 (Siemens) /160 (Elco) /200 (Brötje) /360 (Elco) /460 (Elco)	90 90 90 90 90 90	? 109 160 239 234 ?	? 101.4 101.6 101.4 101.6 ?	? 3.4 3.5 3.4 3.5 ?	✓	✓	-
RVS65.583	/200 (Brötje)	116	239	101.4	3.3	✓	✓	-

16 Der RVS43.325 scheint noch Parameter aufzuweisen, die bisher noch nicht implementiert wurden.
17 Bei ?-Einträgen liegen derzeit keine detaillierteren Angaben der/des jeweiligen Nutzer/s vor.

3.3 Hinweis: Neue Modellgeneration - NICHT unterstützter Regler von Brötje

Aus aktuellem Anlass sei an dieser Stelle darauf hingewiesen, dass die genannten Heizungshersteller neue Gerätemodelle auf den Markt gebracht haben, deren Regler nach bisherigem Wissensstand NICHT mit BSB-LAN kompatibel sind.

Bei Brötje handelt es sich hierbei um die Modellreihe WLS/WLC und BOK.
Bei diesen Modellen sind scheinbar ‚IWR CAN'-basierte Regler verbaut, die weder einen LPB noch einen BSB aufweisen.

Das folgende Bild einer WLC24-Platine zeigt die dort vorhandenen Anschlüsse.

Abbildung 7: Anschlüsse des neuen Reglertyps einer Brötje WLC24 - inkompatibel mit BSB-LAN!

Neben einer Servicebuchse (vermutlich IWR CAN) sind dort ein nicht weiter dokumentierter ‚L-Bus' und ein ‚R-Bus' zugänglich.
Am ‚R-Bus' (Raumgeräte-Bus) kann bei Bedarf entweder ein Raumthermostat (On/Off), ein entsprechendes QAA-Raumgerät oder das neue ‚smarte' Raumgerät „Brötje IDA" angeschlossen werden.

ACHTUNG: An keinem dieser Anschlüsse ist BSB-LAN anschließbar!

4. Installation der Arduino IDE und Konfiguration des Adapters

- Downloade und installiere die aktuelle Version der Arduino IDE von https://www.arduino.cc/en/Main/Software (Windows-, Mac- und Linux-Version verfügbar).
- Schließe den Arduino (samt Ethernet-Shield und Adapter) via USB an deinem PC an.
- Downloade die aktuelle BSB-LAN-Version von https://github.com/fredlcore/bsb_lan und entpacke die heruntergeladene Datei *bsb_lan-master.zip*.
- Benenne den nun erstellten Ordner *bsb_lan-master* in ***BSB_lan*** um!
- Benenne die Datei *BSB_lan_config.h.default* in ***BSB_lan_config.h*** um!
- Öffne den BSB_lan-Sketch mittels eines Doppelklicks auf die Datei *BSB_lan.ino* im BSB_lan-Ordner. Die dazugehörigen Dateien *BSB_lan_config.h* und *BSB_land_defs.h* werden automatisch mit geladen.
- Konfiguriere die IP-Adresse in *BSB_lan_config.h* deinem Netzwerk entsprechend. Die voreingestellte IP 192.168.178.88 funktioniert mit den meisten Standard-Routern wie bspw. Fritz!Box, aber prüfe, ob die IP bereits anderweitig vergeben ist, damit es nicht zu einer Adresskollision kommt.
- *Wichtig:* Passe nun die weiteren Einstellungen in der Datei *BSB_lan_config.h* deinen Wünschen und Hardwaregegebenheiten (Pinbelegungen, angeschlossene DS18B20- und/oder DHT22-Sensoren, zu loggende Parameter etc.) entsprechend an!
- Wähle "Arduino/Genuino Mega or Mega 2560" unter Tools/Board bzw. Werkzeuge/Board.
- Wähle "ATmega 2560" unter Tools/Processor bzw. Werkzeuge/Prozessor.
- Wähle "AVRISP mkII" unter Tools/Programmer bzw. Werkzeuge/Programmer.
- Passe die Datei *BSB_lan_config.h* deinen Wünschen entsprechend an (siehe nächstes Kapitel) und lade den Sketch mittels Sketch/Upload bzw. Sketch/Hochladen auf den Arduino.
- Verbinde den Arduino mittels LAN-Kabel mit deinem Router/Switch. Stelle dabei sicher, dass eine Stromversorgung für den Arduino via USB oder externem Netzteil besteht!
- Öffne die Seite `http://<IP-Adresse>/` (oder `http://<IP-Adresse>/<passkey>/` wenn die Passkey-Funktion (s.u.) genutzt wird) um zu sehen, ob alles korrekt kompiliert und hochgeladen wurde.
  Die Startseite des Webinterface des Adapters sollte erscheinen.
  Sollte sie nicht erscheinen, drücke einmal kurz auf den Reset-Knopf des Arduino und rufe die Startseite erneut auf.
  Unter `http://<IP-Adresse>/C` (bzw. Menüpunkt „Konfiguration“ im Webinterface) kannst du deine Konfiguration überprüfen.

  Wenn alle einstellungsrelevanten Parameter in der Datei BSB_lan_config.h angepasst sind (siehe nächstes Kapitel) und der Zugriff auf das Webinterface möglich ist, fahre mit der Funktionsüberprüfung des Adapters fort.

5. Einstellungsrelevante Parameter der BSB-LAN-Software

Folgende Parameter in der Datei *BSB_lan_config.h* können bzw. sollten vor der Verwendung des Adapters angepasst werden:

- Typ des LAN-Shield-Chips (W5100 oder W5500):
  Standardmäßig wird die Verwendung eines LAN-Shields mit dem Chip des Typs W5100 vorausgesetzt (s. Aufkleber auf der LAN-Buchse des Shields). Soll aber ein Shield mit Chip-Typ W5500 verwendet werden, ist das entsprechende Definement zu aktivieren:
  `#define ETHERNET_W5500`

  Zusätzlich muss die Datei *Ethernet2.zip* im Unterverzeichnis *src* entpackt werden.

- IP-Adresse:
  `#define IPAddr 192,168,178,88`

- GatewayIP: Durch Aktivierung des Definements und Anpassung der IP kann hier optional die IP eines (nicht-Standard-)Gateways definiert werden:
  `#define GatewayIP 192,168,178,1`

- Ethernet-Port:
  `#define Port 80`

- Um das System vor einem ungewollten Zugriff von außen zu schützen, kann die Funktion des Sicherheitsschlüssels (`PASSKEY`) aktiviert werden (sehr einfach und nicht wirklich sicher!):
  `#define PASSKEY "1234"`

  Falls die PASSKEY-Funktion aktiviert ist, muss die URL bei einem Aufruf des Webinterfaces den definierten Schlüssel als erstes Element enthalten, bspw.
  `http://<IP-Adresse>/<passkey>/`

  um die Hilfeseite zu sehen.
  Bitte nicht den Slash hinter dem Passkey vergessen!

  Die URLs in den folgenden Beispielen müssen um die PASSKEY-Definition erweitert werden, falls die Funktion aktiviert wurde.

- Darüber hinaus gibt es zwei weitere Sicherheitsfunktionen: `TRUSTED_IP` und `USER_PASS_B64`

  `TRUSTED_IP` (und `TRUSTED_IP2` für eine weitere IP) kann man auf das letzte Segment einer vertrauenswürdigen IP setzen (z.B. des FHEM-Servers), dann ist der Zugriff nur über die IP mit dieser Endung möglich. Lautet die vertrauenswürdige IP des Clients bspw. 192.168.178.20, so ist `#define TRUSTED_IP 20` einzustellen.

  Mit `USER_PASS_B64` kann ein in Base64-codierter String nach dem Muster *username:passwort* als Zugangssperre gesetzt werden. Voreingestellt ist hier der Benutzername "atari" und das Passwort "800xl" (codiert: YXRhcmk6ODAweGw=). Um eine andere Kombination zu nutzen, gehe bspw. auf die Website https://www.base64encode.org , lasse dein neues Passwort im Format *username:password* erstellen und füge die Codierung hinter dem Definement ein:
  `#define USER_PASS_B64 "YXRhcmk6ODAweGw="`

- Konfiguration des Heizungssystems:
  ```
  const int fixed_device_family = 0;
  const int fixed_device_variant = 0;
  ```
  Wenn die Werte auf 0 gesetzt sind, ist die automatische Erkennung des angeschlossenen Reglers beim Starten des Arduino aktiviert.
  Alternativ kann hier die Ausgabe von `http://<IP-Adresse>/6225/6226` eingetragen werden (6225 = Gerätefamilie / device family & 6226 = Gerätevariante / device variant). Ein fest eingestellter Wert (laut 6225&6226) stellt sicher, dass BSB-LAN auch dann noch korrekt arbeitet, wenn die Heizung bzw. der Regler erst nach dem Starten des Arduino eingeschaltet wird (da in dem Fall die automatische Erkennung des angeschlossenen Reglers nicht funktionieren kann, da ja keine Rückmeldung vom Regler kommt).
- Die Sprache der Benutzeroberfläche des Webinterface des Adapters ist auf Deutsch voreingestellt. Möchte man die Sprache auf Englisch umstellen, muss das entsprechende Definement deaktiviert werden:
  ```
  //#define LANG_DE;
  ```
- Sollen bei einer Abfrage via Webinterface auch die unbekannten Parameter (Fehlermeldung „error 7 (parameter not supportet)“) angezeigt werden, so muss das Definement
  ```
  #define HIDE_UNKNOWN
  ```
  auskommentiert werden.
- Sollen OneWire-Temperatursensoren (DS18B20) verwendet werden, muss die entsprechende Pinbelegung (DATA-Anschluss des Sensors am Adapterboard) definiert werden (<n> = Pinnummer):
  ```
  #define ONE_WIRE_BUS <n>
  ```
- Sollen DHT22-Sensoren (Temperatur & Feuchtigkeit) verwendet werden, muss die entsprechende Pinbelegung (DATA-Anschluss des Sensors am Adapterboard) definiert werden (<n> = Pinnummer):
  ```
  #define DHT_BUS <n>
  ```
- Sollen 24h-Durchschnittswerte von bestimmten Parametern berechnet werden, müssen diese bei der entsprechenden Variable eingetragen werden, bspw.:
  ```
  int avg_parameters[20] = {
   8700,     // Außentemperatur
   8740      // Raumtemperatur-Ist
  };
  ```
- Sollen bestimmte Werte/Parameter auf eine microSD-Karte geloggt werden, ist eine FAT32-formatierte Karte im entsprechenden Slot auf dem Ethernet-Shield einzulegen und das entsprechende Definement zu aktivieren:
  ```
  #define LOGGER
  ```
  Die zu loggenden Parameter müssen dann bei der entsprechenden Variable eingetragen werden, bspw.:
  ```
  int log_parameters[20] = {
   8700,     // Außentemperatur
   8740      // Raumtemperatur-Ist
  };
  ```

Wenn mehrere DS18B20- oder DHT22-Sensoren geloggt werden sollen, müssen diese bei den Log-Parametern entsprechend einzeln untereinander aufgeführt werden, bspw.:

```
[...]
20200,     // Spezialparameter 20200-20299: DS18B20-Sensoren 1-100 (/T)
20201,
20202,
[...]
```

loggt die Werte der DS18B20-Sensoren 1-3.

Hinweis:
Zum Loggen der Brennerstarts und -laufzeiten müssen die Spezialparameter 2001 und 2002 aufgeführt werden (siehe auch die Beschreibung in der Datei BSB_lan_config.h). Bei einem zweistufiger Ölbrenner, dessen Regler die entsprechenden Broadcasts schickt und bei dem eine Differenzierung der Brennerstufen möglich ist[18] müssen hier zusätzlich 2003 und 2004 mit aufgeführt werden!

- Soll die IPWE-Erweiterung aktiviert werden, ist das entsprechende Definement `#define IPWE`

  zu aktivieren, die gewünschten Parameter sind wie gewohnt einzutragen.

  Tipp:
  Werden DS18B20- und/oder DHT22-Sensoren verwendet, werden diese hier standardmäßig mit angezeigt (`http://<IP-Adresse>/ipwe.cgi`). Dabei wird neben den gemessenen Werten auch die jeweils spezifische Hardwarekennung der Sensoren aufgeführt. Dies ist besonders bei einer Ersteinrichtung für eine eindeutige Unterscheidung der einzelnen Sensoren hilfreich.

- Sollen optionale MAX!-Thermostate zum Einsatz kommen, muss das entsprechende Definement `#define MAX_CUL 192,168,178,5` aktiviert sowie die URL und die spezifischen MAX!-IDs entsprechend angepasst werden.

- Soll der Arduino per URL-Befehl mittels `http://<IP-Adresse>/N` resettet werden können, muss das entsprechende Definement aktiviert werden:
  `#define RESET`

- MAC-Adresse des Ethernet-Shields:
  `byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEA};`

  Üblicherweise befindet sie sich auf einem Aufkleber auf dem Ethernet-Shield, diese ist dann einzutragen. Sollte kein Aufkleber mit einer spezifischen Adresse vorhanden sein, kann die voreingestellte Adresse beibehalten werden. Eine Änderung ist i.d.R. nur nötig, wenn mehr als ein Adapter verwendet wird.

- Konfiguration des Adapters:
  `BSB bus(68,69,<my_addr>,<dest_addr>);`

  RX-Pin, TX-Pin, eigene Bus-Adresse (voreingestellt auf 0x06=RGT1), Bus-Adresse des Zielsystems (voreingestellt auf 0x00=Heizungsregler).

  Voreingestellt wird der Adapter mit `BSB bus(68,69)` am angeschlossenen Regler als RGT1 angemeldet.

18 Derzeit nur RVS43.325, siehe Hinweis in Kapitel 8.1 unter „/B“.

Wenn bereits ein Raumgerät (RGT1) vorhanden ist, kann bzw. sollte der Adapter u.U. als RGT2 am angeschlossenen Regler angemeldet werden, um etwaige Adresskollisionen[19] zu vermeiden:
`BSB bus(68,69,7);`

Hinweis:
Bitte beachte, dass ein als RGT1 angemeldeter Adapter eventuell keine Temperaturen an einen HK2 senden kann, und ein als RGT2 angemeldeter Adapter eventuell keine Temperaturen an einen HK1 senden kann!
Vereinzelt kam es schon vor, dass Raumtemperaturen kurioserweise nur von einem als RGT2 angemeldeten Adapter vom HK1 berücksichtigt wurden.
Ob eine Bedienung eines HK1 mit einem als RGT2 (oder HK2 mit einem als RGT1) angemeldeten Adapter in vollem Umfang möglich ist, wurde bisher noch nicht ausgiebig getestet.

- Bus-Protokoll:
  `uint8_t bus_type = bus.setBusType(0);`

  Voreingestellt ist 0 für BSB, für LPB ist 1 einzustellen, für PPS hingegen 2.

- In der Voreinstellung ist der Zugriff des Adapters auf den Regler auf Lesen beschränkt, d.h. ein Setzen bzw. Verändern von Parametern der Heizungssteuerung per Adapter ist standardmäßig nicht möglich.
  Das betreffende Definement lautet:
  `#define DEFAULT_FLAG FL_RONLY;`

  Wer den Status ändern will, um *generell* Werte und Einstellungen des Reglers per Adapter verändern zu können, muss das Flag auf 0 setzen:
  `#define DEFAULT_FLAG 0;`

  Ist diese Funktion nur bei *ausgewählten* Parametern (z.B. 10000 oder 710) gewünscht, muss bei dem genannten Definement nach wie vor das Flag generell auf `FL_RONLY` gesetzt sein und dann in der Datei *BSB_lan_defs.h* der Wert `DEFAULT_FLAG` des gewünschten Parameters durch `0` ersetzt werden.

- Definement `#define DEBUG`: Soll BSB-LAN auf nicht-freigebene reglerspezifische CommandIDs überprüft werden (siehe 8.2.5 Überprüfen auf nicht-freigegebene reglerspezifische Command IDs), muss das Definement vorübergehend aktiviert werden. Nach Abfrage von `http://<IP-Adresse>/Q` sowie für den Normalbetrieb kann das Definement wieder auskommentiert werden.

19 Eine Adresskollision wurde bisher nur bei einem *RGB (QAA55)* und einem Adapter festgestellt, die beide gleichzeitig als Raumgerät 1 angemeldet waren. Ein *RGT (QAA75)* und ein Adapter mit gleichzeitiger Anmeldung als Raumgerät 1 verursachte im Testbetrieb hingegen keine Adresskollision.

6. Funktionsüberprüfung und erste Nutzung des Adapters

Um zu überprüfen, ob der angeschlossene Regler korrekt erkannt wird, sollte bei der Ersteinrichtung eine Funktionsüberprüfung vorgenommen werden.
Dazu bietet sich folgende Vorgehensweise an:

1. Den Regler ausschalten und *polrichtig* mit dem Adapter via BSB verbinden.
   Hinweis:
   Wenn (später) der LPB genutzt werden soll, muss sowohl der Bus-Typ in der Datei *BSB_lan_config.h* als auch der Anschluss am Regler geändert werden!
2. Den Regler einschalten und überprüfen, ob die rote LED auf dem Adapter leuchtet. Sollte die LED in unregelmäßigen Abständen flackern, ist dies keine Fehlfunktion, sondern zeigt Aktivität auf dem Bus an.
3. Den Adapter nun via USB mit dem PC und via LAN mit dem Netzwerk verbinden.
4. Nun die Arduino IDE starten, den korrekten Anschluss des Arduino Mega 2560 auswählen (COM-Port) und dann unter ‚Werkzeuge' den ‚Seriellen Monitor' starten.
5. a) Wird der angeschlossene Regler automatisch korrekt erkannt, steht am Anfang der Ausgabe des seriellen Monitors bei „Device family" und „Device variant" jeweils ein Wert, der nicht „0" ist.
   Beispiel:
   ```
   […]
   Device family: 96
   Device variant: 100
   […]
   ```

   Nun sollte die Verwendung von BSB-LAN möglich sein.

   Hinweis:
   Sollte ein bisher nicht in Kapitel 3. Unterstützte Heizungssysteme und Regler aufgeführter Regler zum Einsatz kommen, bitte die entsprechenden Daten wie beschrieben auslesen und melden.

   b) Wird der angeschlossene Regler nicht automatisch korrekt erkannt, steht bei „Device family" und „Device variant" jeweils eine „0", zusätzlich stehen vor „Device family" sechs Zeilen „query failed".
   Beispiel:
   ```
   […]
   query failed
   query failed
   query failed
   query failed
   query failed
   query failed
   Device family: 0
   Device variant: 0
   […]
   ```

   Meist liegt der Grund hierfür dann in einem Problem des Hardware-Setups oder der Verkabelung. Weitere Hinweise zur Fehlersuche finden sich im Kapitel 14. Etwaige Probleme und deren mögliche Ursachen.

7. BSB-LAN Web - das Webinterface des Adapters

Die Startseite des Webinterface wird angezeigt, wenn ohne weitere Parameter auf die URL des Servers zugegriffen wird:
`http://<IP-Adresse>`

Bei Verwendung eines Passkeys oder weiterer optionaler Sicherheitsfunktionen muss die URL entsprechend erweitert werden, bei Passkey-Verwendung bspw.:
`http://<IP-Adresse>/<passkey>/`

Bitte den Slash hinter dem Passkey nicht vergessen!

Abbildung 8: Screenshot der Startseite des Webinterface

Prinzipiell sind alle Parameter in Kategorien zusammengefasst, die den im Display dargestellten Untermenükategorien entsprechen, wenn auf den Regler des Heizungssystems vom integrierten Bedienteil aus zugegriffen wird.

Ein Klick auf den Menüpunkt „Heizungsfunktionen“ zeigt eine vollständige Übersicht der Kategorien, die wiederum ebenfalls anwählbar sind (s. Abb. 8).

Ein Klick auf eine der gezeigten Kategorien (bspw. Heizkreis 1) startet eine Komplettabfrage der jeweiligen Kategorie, also aller Parameter, die in dieser Kategorie verfügbar sind (s. Abb. 9).

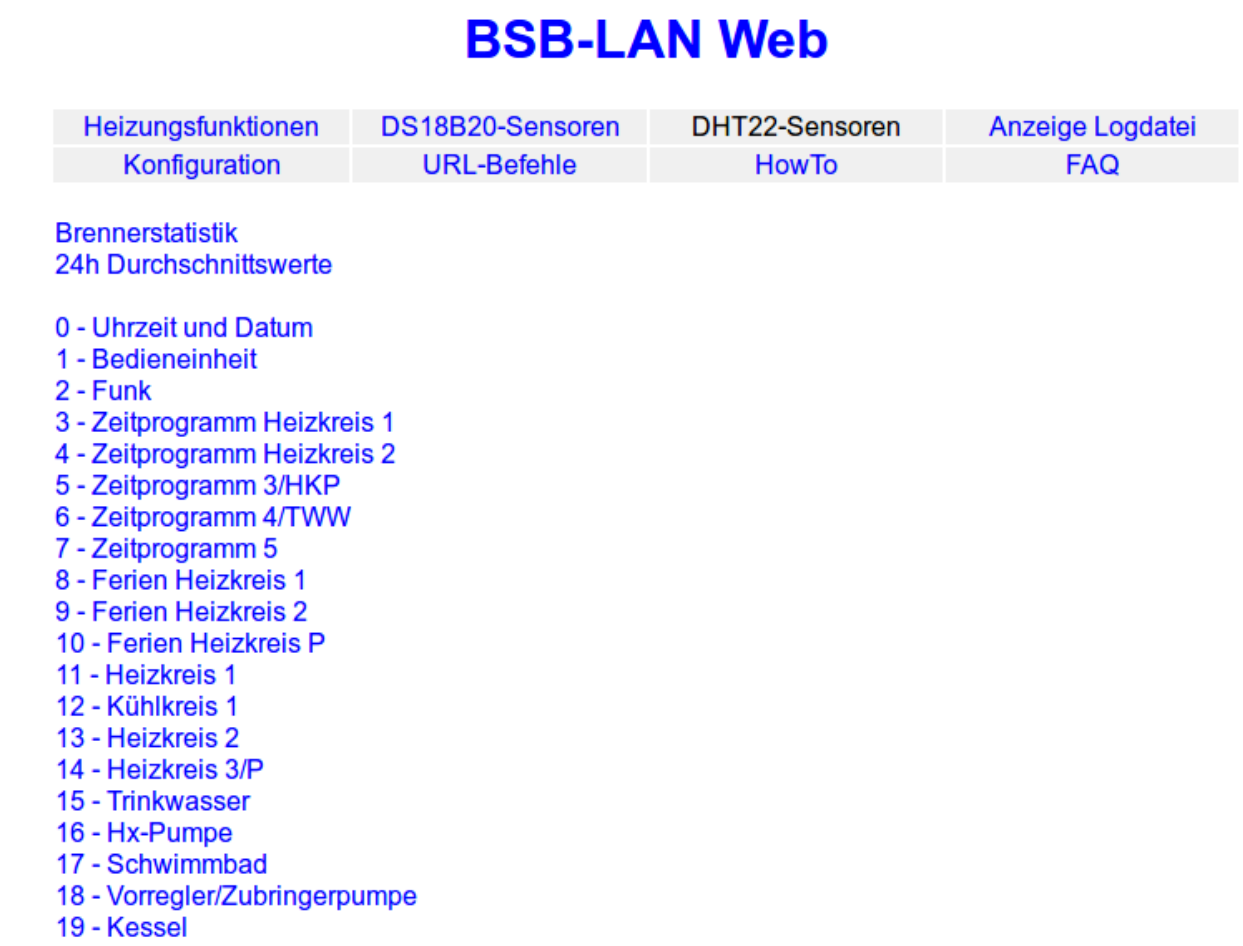


Abbildung 9: Screenshot Menü "Heizungsfunktionen" mit den Parameter-Kategorien (Auszug)

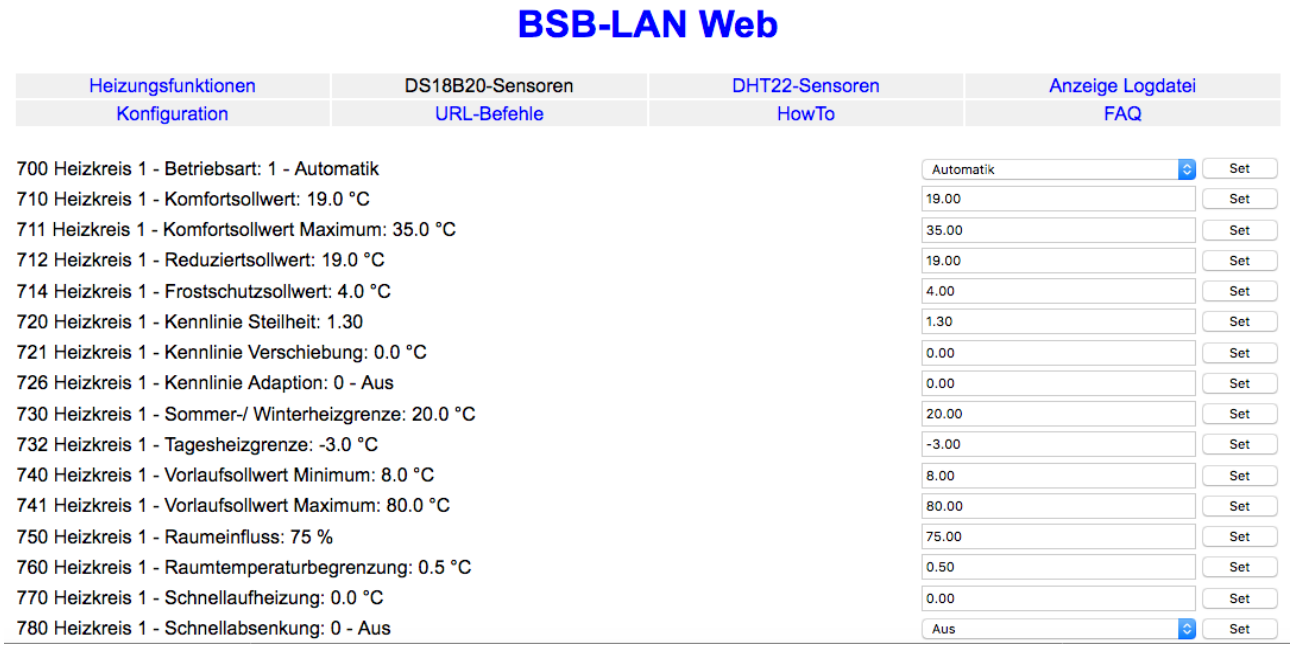


Abbildung 10: Screenshot Parameterauflistung Kategorie "Heizkreis 1" (Auszug) bei aktivierter Set-Funktion

Unter der URL
`http://<IP-Adresse>/C`
wird eine Übersicht der Konfiguration dargestellt. Dort sind u.a. der Monitor-Modus, der Verbositäts-Level, die definierten Pins für optional angeschlossene Sensoren, die zu loggenden Parameter und Parameter, von denen 24h-Mittelwerte berechnet werden sollen, auf einen Blick überprüfbar.

Eine schwarze Schrift bei den Schaltflächen für DS18B20- und DHT22-Sensoren zeigt an, dass diese nicht definiert sind.

Bei dem folgenden abgebildeten Beispiel sind lediglich DS18B20-Sensoren definiert und es werden keinerlei 24h-Mittelwerte berechnet. Alle zehn Minuten (→ 600 Sekunden) werden Temperaturwerte von insgesamt sechs DS18B20-Sensoren geloggt.

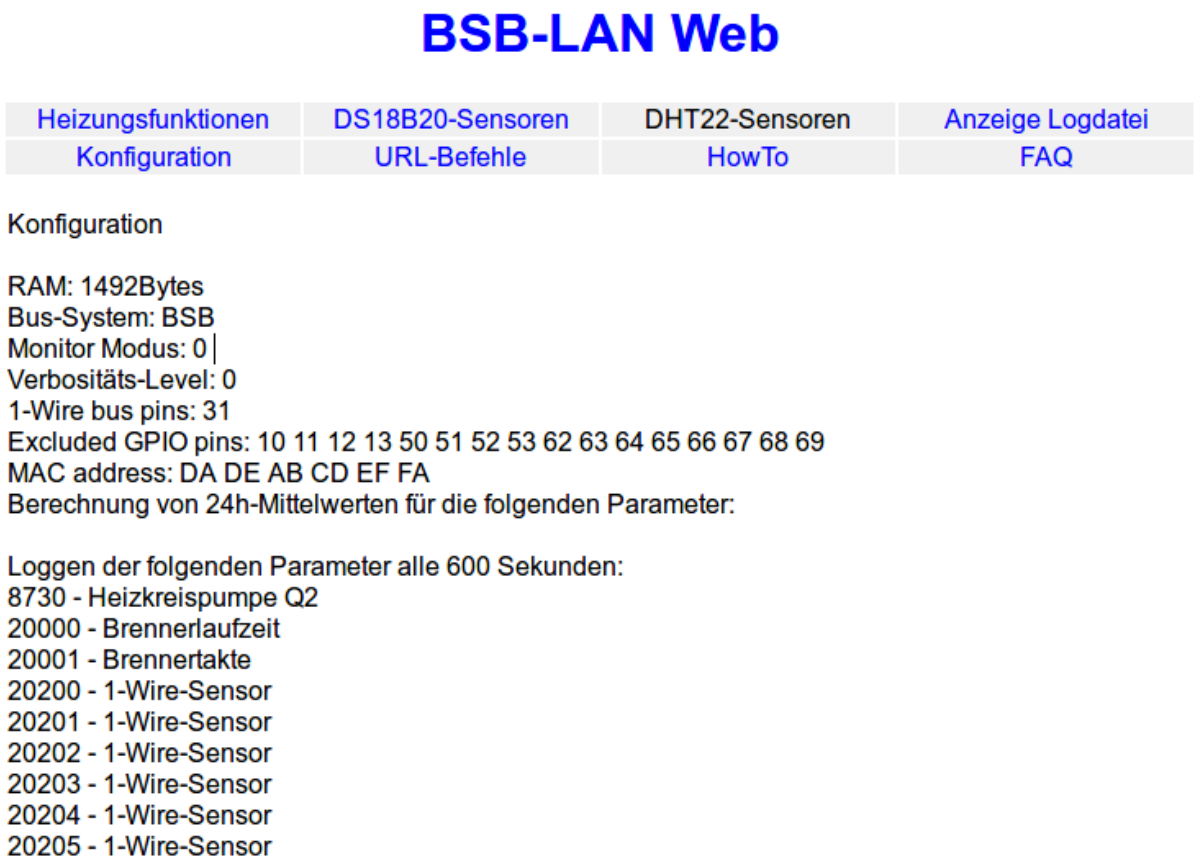


Abbildung 11: Screenshot der Übersichtsseite "Konfiguration" (<IP-Adresse>/C)

Der Menüpunkt „URL-Befehle“ zeigt eine Auflistung und kurze Erklärung der URL-Befehle.

Grau hinterlegte Zeilen zeigen an, dass der Befehl nicht verfügbar ist. Im nachfolgend gezeigten Beispiel betrifft dies die nicht-installierten DHT22-Sensoren.

BSB-LAN Web

Heizungsfunktionen	DS18B20-Sensoren	DHT22-Sensoren	Anzeige Logdatei
Konfiguration	URL-Befehle	HowTo	FAQ

Erweiterte Befehle:

/K	Alle verfügbaren Kategorien auflisten.
/Kx	Alle Werte von Kategorie x abfragen.
/x-y	Alle Werte eines Zeilenbereichs abfragen (von Zeile x bis Zeile y).
/Sx=v	Setze Wert v (value) für den Parameter x (Leerzeichen nach = deaktiviert den Wert).
/Ix=v	Sende eine INF Nachricht für den Parameter x mit dem Wert v.
/Ex	Alle enum-Werte für Parameter x auflisten.
/Rx	Frage den Reset-Wert für Parameter x ab.
/Px	Setzen des Bus-Protokolls (0=BSB, 1=LPB).
/Vn	Setze den Verbositäts-Level auf n.
/Mn	Bus-Monitor aktivieren/deaktivieren (n=0 deaktivieren, n=1 aktivieren).
/Gxx	Abfragen des GPIO Pins xx.
/Gxx=y	Setzen des GPIO Pins xx auf high (y=1) oder low (y=0).
/A	24h-Durchschnittswerte von ausgewählten Parametern anzeigen (in BSB_lan_config.h definieren).
/A=x,y,z	Ändern der 24h-Durchschnittswerte in x,y,z (bis zu 20 Parameter).
/B	Anzeige der akkumulierten Broadcast-Telegramme zu Brenner- und TWW-Aktivitäten.
/B0	Zurücksetzen der akkumulierten Broadcast-Telegramme.
/T	Abfrage von angeschlossenen DS18B20 Temperatursensoren (optional).
	Aktiviere das Definement `#define DHT_BUS` in BSB_lan_config.h für den folgenden Befehl:
/H	Abfrage von DHT22 Feuchtigkeits-/Temperatursensoren (optional).
/D	Darstellung des Logfiles datalog.txt auf der microSD-Karte.
/D0	Löschen bzw. Zurücksetzen des Logfiles datalog.txt auf der microSD-Karte.
/L=x,y,z	Setzt das Logging-Intervall auf x Sekunden und (optional) die Logging-Parameter auf y und z (bis zu 20 Parameter). Um das Loggen zu deaktivieren, kann L=0,0 genutzt werden.
/LU=x	Wenn Bus-Telegramme geloggt werden (Logging-Parameter 30000 als einzigen Parameter setzen!), logge nur unbekannte commandIDs (x=1) oder alle Telegramme (x=0).
/LU=x	Wenn Bus-Telegramme geloggt werden (Logging-Parameter 30000 als einzigen Parameter setzen!), logge nur Broadcast-Telegramme (x=1) oder alle Telegramme (x=0).
/X	Reset des Arduino durchführen.

Mehrere Abfragen können miteinander verkettet werden, z.B. `/K0/710/8000-8999/T`

Abbildung 12: Screenshot Menü "URL-Befehle"

Zusätzlich zum Webinterface kann somit auf alle Funktionen mittels Eingabe des entsprechenden Befehls direkt zugegriffen werden. Dies ist nützlich, wenn der Adapter in Verbindung mit Heimautomationssystemen wie bspw. FHEM genutzt wird.

Eine Übersicht und ausführlichere Beschreibung der URL-Befehle findet sich im Kapitel 8.

Generell werden alle Heizungsparameter anhand ihrer Zeilennummern abgefragt. Eine nahezu vollständige Übersicht findet sich bspw. im Systemhandbuch des Brötje ISR Plus.

Einige Zeilen sind 'virtuell' und wurden hinzugefügt, um bspw. den Zugang zu komplexen Einstellungen wie den Tagesprogrammen zu erleichtern.

Eine grafische Darstellung des Logfiles erfolgt bei Klick auf „Anzeige Logdatei“.

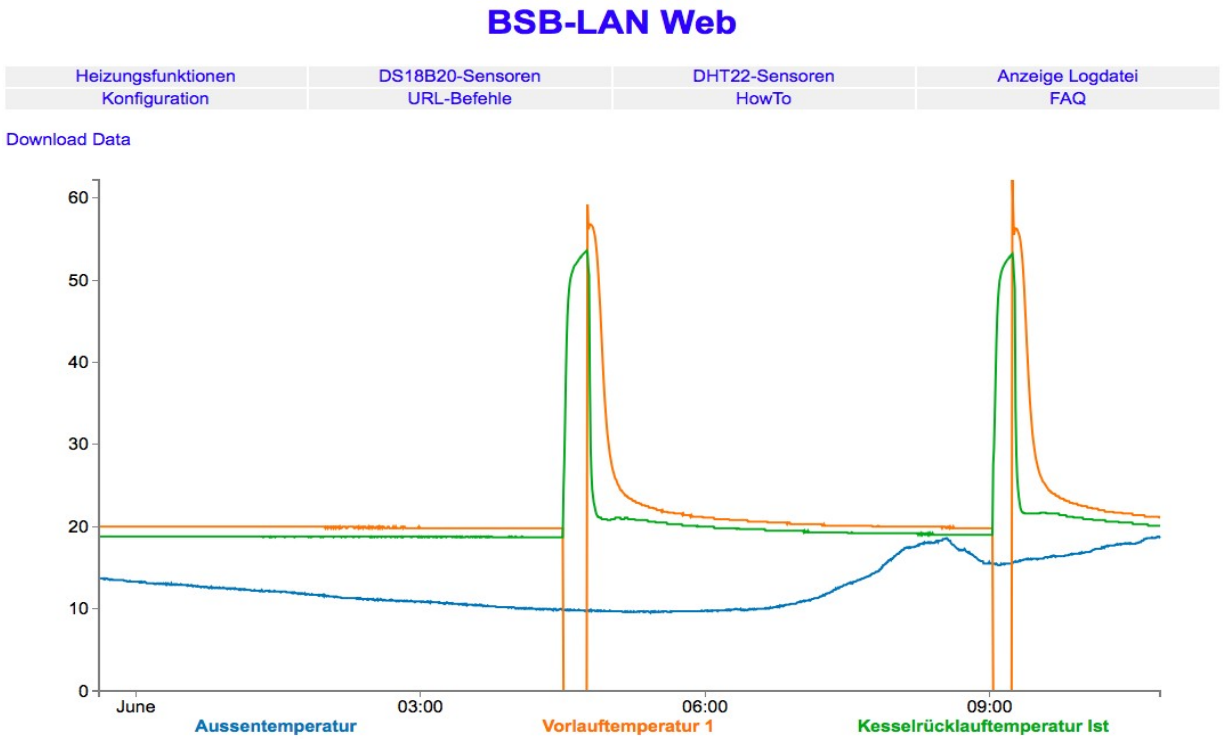


Abbildung 13: Screenshot der grafischen Darstellung der Logdatei

Bitte beachte, dass der Arduino nicht multitaskingfähig ist. Eine neue Abfrage kann erst erfolgen, nachdem die vorhergehende Abfrage komplett beendet ist. Speziell die Abfrage mehrerer Parameter, ganzer Kategorien oder auch des SD-Logfiles kann u.U. eine längere Zeit in Anspruch nehmen, während dieser der Adapter nicht ‚ansprechbar‘ ist.

8. URL-Befehle und Spezialfunktionen

Da das Webinterface prinzipiell nur ‚aufgesetzt' ist, um eine Bedienung ohne weitere Programme wie bspw. FHEM zu ermöglichen, ist ein direkter Zugriff auf die einzelnen Funktionen und Parameter mittels anderer Programme grundsätzlich möglich.

8.1 Auflistung und Beschreibung der URL-Befehle

- **Alle Kategorien auflisten:**

  `http://<IP-Adresse>/K`
  Bei diesem Befehl kommuniziert der Adapter nicht mit dem Heizungssystem.
  Es ist eine softwareseitige, interne Funktion.

- **Alle enum-Werte für Parameter <x> auflisten:**

  `http://<IP-Adresse>/E<x>`
  Bei diesem Befehl kommuniziert der Adapter nicht mit dem Heizungssystem.
  Es ist eine softwareseitige, interne Funktion. Dieser Befehl ist nur für Parameter des Typs VT_ENUM verfügbar.

- **Alle Werte von Kategorie <x> abfragen:**

  `http://<IP-Adresse>/K<x>`

- **Alle Werte von Parameter <x> abfragen:**

  `http://<IP-Adresse>/<x>`

- **Alle Werte eines Parameterbereichs (von-bis) abfragen:**

  `http://<IP-Adresse>/<x>-<y>`

- **Mehrere Abfragen können miteinander verkettet werden, z.B.:**

  `http://<IP-Adresse>/K11/8000/8003/8005/8300/8301/8730-8732/8820`

- **Frage den Reset-Wert für Parameter <x> ab:**

  `http://<IP-Adresse>/R<x>`
  Im Display der integrierten Heizungssteuerung gibt es für einige Parameter eine Reset-Option. Ein Reset wird vorgenommen, indem das System nach dem Reset-Wert gefragt wird und dieser anschließend gesetzt wird.

- **Setze Wert <v> (value) für den Parameter <x> mit optionalem Ziel <z>:**

  `http://<IP-Adresse>/S<x>=<v!z>`
  Die gewünschte Gerätezieladresse ist als <z> einzufügen, wenn <!z> nicht eingegeben wird, wird die Standardzieladresse verwendet.

  Um einen Parameter auf 'abgeschaltet/deaktiviert' zu setzen, muss lediglich ein leerer Wert eingefügt werden: `http://<IP-Adresse>/S<x>=`

Hinweis: Voreingestellt ist nur der Lesezugriff erlaubt, ein Verändern von Einstellungen ist per default nicht möglich. Um dies zu ändern, ist Schreibzugriff für den entsprechenden Parameter zu gewähren. Siehe hierzu den entsprechenden Punkt in Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software.

ACHTUNG: Diese Funktion ist nicht ausgiebig getestet! Bitte sei vorsichtig mit dieser Funktion und nutze sie ausschließlich auf dein eigenes Risiko hin. Das Format des Wertes hängt von seinem Typ ab. Einige Parameter können abgeschaltet werden.

- **Sende eine INF-Nachricht für den Parameter <x> mit dem Wert <v>:**

  `http://<IP-Adresse>/I<x>=<v>`
  Einige Werte können nicht direkt gesetzt werden. Das Heizungssystem wird mit einer TYPE_INF-Nachricht informiert, bspw. bei der Raumtemperatur:
  `http://<IP-Adresse>/I10000=19.5` → Raumtemperatur beträgt 19.5°C

- **Bus-Typ (BSB, LPB oder PPS) vorübergehend ändern:**

  `http://<IP-Adresse>/P<x>`
  Wechselt zwischen BSB (<x>=0), LPB (<x>=1) und PPS (<x>=2).
  Nach einem Reset/Neustart des Arduino wird die Einstellung aus der Datei *BSB_lan_config.h* verwendet. Um den Bus-Typ dauerhaft festzulegen, sollte die Option `setBusType config` in der Datei *BSB_lan_config.h* entsprechend angepasst werden.

- **Zusätzlich die eigene oder die Zieladresse mittels URL-Befehl wechseln:**

  Dazu muss der Befehl
  `http://<IP-Adresse>/P<x,y,z>`
  genutzt werden, wobei
  <x> = Bus (0 = BSB, 1 = LPB, 2 = PPS),
  <y> = eigene Adresse (default 0x06 = RGT1) und
  <z> = Zieladresse (default 0x00 = Geräteadresse 1) sind.
  Leerwerte bei den Adressen belassen den bisherigen Wert (= Adresse).
  ACHTUNG: Diese Funktion wurde noch nicht ausgiebig getestet!

- **Setze den Verbositäts-Level auf <n>:**

  `http://<IP-Adresse>/V<n>`
  Der voreingestellte Verbositäts-Level ist 1. Somit wird standardmäßig der Bus überwacht und alle Daten werden zusätzlich im Raw-Hex-Format dargestellt.
  Soll der Modus deaktivert werden, so ist <n> auf 0 zu setzen (URL-Befehl: `/V0`).

  Der Verbositäts-Level betrifft sowohl die serielle Konsole des Arduino als auch (optional) das Loggen der Bus-Daten auf die microSD-Karte, so dass die Speicherkarte u.U. sehr schnell voll wird! Im Fall des Loggens auf die interne microSD-Karte ist es daher empfehlenswert, den Verbostitätsmodus bereits in der Datei *BSB_lan_config.h* zu deaktivieren (`byte verbose = 0`).
  Die html-Ausgabe bleibt mit `/V1` unverändert.

- **Bus-Monitor aktivieren:**

  `http://<IP-Adresse>/M<n>`
  Standardmäßig ist der Monitor-Modus deaktiviert (<n>=0).
  Wenn <n> auf 1 gesetzt wird, werden alle Bytes auf dem Bus überwacht. Telegramme werden durch Umbruchzeichen als solche erkannt. Jedes Telegramm wird im Hex-Format auf der seriellen Konsole mit einem Zeitstempel in Millisekunden dargestellt.
  Die Ausgabe der Überwachung betrifft nur die serielle Konsole des Arduino, die html-Ausgabe bleibt unverändert.
  Zum Deaktivieren des Monitor-Modus ist <n> wieder auf 0 zu setzen (URL-Befehl: `/M0`).

- **Setzen/Abfragen der GPIO-Pins (GPIO wird als OUTPUT genutzt):**

  `http://<IP-Adresse>/G<xx>[=<y>]`
  Gibt den momentanen Status von GPIO Pin <xx> an, wobei <y>=0 LOW und <y>=1 HIGH ist. Kann ebenfalls benutzt werden, um den Pin auf <y>=0 (LOW) oder <y>=1 (HIGH) zu setzen (bspw. bei einem optional angeschlossenen Relaisboard).
  Reservierte Pins, die nicht gesetzt werden dürfen, können in der *BSB_lan_config.h* unter dem Parameter GPIO_exclude gesperrt werden.

- **Abfragen der GPIO-Pins (GPIO wird als INPUT genutzt):**

  `http://<IP-Adresse>/G<xx>,I`
  Für die reine Abfrage eines externes Gerätes, das an einen GPIO angeschlossen ist (z.B. ein einfaches Koppelrelais), da die Pins per default auf ‚output' gesetzt sind. Der Pin bleibt nach diesem Befehl so lange auf ‚input', bis das nächste Mal mit `/G<xx>=<y>` ein Wert geschrieben wird - ab da ist er dann bis zum nächsten „I" wieder auf ‚output'.

- **24h-Durchschnittswerte von ausgewählten Parametern anzeigen:**

  `http://<IP-Adresse>/A[=parameter1,...,parameter20]`
  Zeigt rollierende 24h-Durchschnittswerte ausgewählter Parameter an. Die Initiale Festlegung dieser Parameter erfolgt in der Datei *BSB_lan_config.h* in der Variable *avg_parameters*.
  Während der Laufzeit kann "`/A=[parameter1],...,[parameter20]`" verwendet werden, um (bis zu 20) neue Parameter zu definieren.

- **Abfrage von DS18B20-Temperatursensoren:**

  `http://<IP-Adresse>/T`
  Gibt die Temperaturwerte von optional angeschlossenen DS18B20-Sensoren aus.

- **Abfrage von DHT22-Feuchtigkeits-/Temperatursensoren:**

  `http://<IP-Adresse>/H`
  Gibt die Temperatur- & Feuchtigkeitswerte von optional angeschlossenen DHT22-Sensoren aus.

- **Akkumulierte Brennerlaufzeit anzeigen:**

  `http://<IP-Adresse>/B`
  Fragt sowohl die akkumulierte Brennerlaufzeit (in Sekunden) und die Brennerstarts/-takte als auch die Anzahl und die Dauer der Ladungen (in Sekunden) des Trinkwasserspeichers ab, die anhand von Broadcast-Nachrichten ermittelt wurden.

  `http://<IP-Adresse>/B0` setzt den Zähler zurück.

  Bei zweistufigen Ölbrennern wird zudem *eventuell* zwischen Stufe 1 und 2 differenziert und die jeweiligen Starts und Laufzeiten werden angezeigt.

  Hinweis:
  Diese Unterscheidung der beiden Brennerstufen findet aufgrund von spezifischen Broadcasts (BC) statt, also der vom Regler automatisch gesendeten Meldungen. Diese stufen-spezifischen BCs werden jedoch nicht von allen Reglern gesendet, die bei zweistufigen Ölbrennern verbaut wurden. Derzeit scheint es, als wenn die entsprechenden BCs nur bei dem Reglertyp RVS43.325 gesendet werden. Bei anderen Reglern werden die Brennerstarts und -laufzeiten kumuliert bei Stufe 1 angezeigt (beinhalten also auch Stufe 2!).
  Mittels Abfrage der Parameter 8330-8333 können die vom Regler gezählten Starts und Betriebsstunden der beiden Stufen nach wie vor abgerufen werden.

  Sollen die Brennerstarts und -laufzeiten von zweistufigen (Öl-)Brennern geloggt werden, beachte bitte den entsprechenden Hinweis in Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software.

- **Aktivieren/Deaktivieren des Loggens auf die microSD-Karte:**

  Prinzipiell erfolgt das Aktivieren/Deaktivieren der Log-Funktion durch das entsprechende Definement in der Datei *BSB_lan_config.h* vor dem Flashen. Während des Betriebes kann jedoch das Loggen deaktiviert werden, indem man folgende Parameter definiert:
  `http://<IP-Adresse>/L=0,0`
  Zum Aktivieren werden dann wieder das Intervall und die gewünschten Parameter eingetragen. Bei einem Reset/Neustart des Arduino werden die Einstellungen aus der Datei *BSB_lan_config.h* verwendet – eine dauerhafte Umstellung der Logging-Parameter sollte also dort erfolgen.

  Hinweis: Der per default aktivierte Verbositäts-Modus sollte im Fall eines dauerhaften Loggens auf die microSD-Karte in der Konfiguration deaktiviert werden (s.o.).

- **Konfiguration des Logfiles:**

  `http://<IP-Adresse>/L=<x>[,<parameter1>,<...>,<parameter20>]`
  Setzt während der Laufzeit das Logging-Intervall auf <x> Sekunden und (optional) die Logging-Parameter auf [parameter1], [parameter2] etc.

  Das Logging muss durch das Definement `#define LOGGING` in der Datei *BSB_lan_config.h* aktiviert werden und kann initial anhand der Variablen `log_parameters` und `log_interval` konfiguriert werden (s.o.).

- **Konfiguration des Loggens von Bus-Telegrammen:**

  `http://<IP-Adresse>/LU=<x>`
  Wenn Bus-Telegramme geloggt werden (Parameter 30000 als einzigen Parameter loggen), logge nur die unbekannten Command IDs (<x>=1) oder alle (<x>=0) Telegramme.

  `http://<IP-Adresse>/LB=<x>`
  Wenn Bus-Telegramme geloggt werden (Parameter 30000 als einzigen Parameter loggen), logge nur die Broadcasts (<x>=1) oder alle (<x>=0) Telegramme.

- **Darstellung des Logfiles:**

  `http://<IP-Adresse>/D`
  Zeigt den Inhalt der Datei *datalog.txt*, die sich auf der microSD-Karte im Slot des Ethernet-Shields befindet.

  Mittels
  `http://<IP-Adresse>/D0`
  kann die Datei *datalog.txt* zurückgesetzt werden, gleichzeitig wird eine korrekte CSV-Header-Datei generiert (dieser Schritt wird zudem für die erste Benutzung empfohlen, bevor das Loggen startet).

  Wer Parameter auf SD-Karte loggt, hat neben der reinen Textform auch die Möglichkeit, unter
  `http://<IP-Adresse>/DG`
  einen Graphen angezeigt zu bekommen.

  Hinweis:
  Für `/DG` muss bei Javascript-Blockern die Domain d3js.org freigegeben werden, da der Arduino weiterhin nur die CSV-Datei in den Browser lädt und diese dann mit dem D3-Framework grafisch aufbereitet wird.
  Wird die Log-Datei via Webinterface mittels Klick auf „Anzeige Logdatei“ aufgerufen, erfolgt standardmäßig zuerst die grafische Darstellung.

- **Resetten/Restarten des Arduino:**

  `http://<IP-Adresse>/N`
  Reset und anschließender Restart des Arduino nach einem Pausieren für ca. 15 Sekunden (Voraussetzung: `#define RESET` in *BSB_lan_config.h*, s.o.).

  Hinweis: Hierbei wird gleichzeitig auch noch das EEPROM des Arduino mit Nullen überschrieben. Dies hat z.Z. jedoch nur Auswirkung für PPS bzw. MAX!-Nutzer: Bei PPS werden damit die zwischengespeicherten Werte gelöscht, bei MAX! die registrierten Geräte (diese müssen sich dann durch einen Druck auf die Anlerntaste gegenüber BSB-LAN neu identifizieren).

- **Abfrage von MAX!-Thermostaten:**

  `http://<IP-Adresse>/X`
  Gibt die Temperaturen von optional angeschlossenen MAX!-Thermostaten wieder. Diese sind zuvor in der Datei *BSB_lan_config.h* zu definieren.

  Hinweis: Bei MAX!-Geräten, die in BSB-LAN aufgenommen werden sollen, muss jeweils einmal die Anlern-Taste gedrückt werden (zu Erkennen an dem anschließenden 30-Sekunden-Countdown). Ein bestehendes Pairing zwischen den Geräten und einem Max!Cube bzw. CUL wird dabei nicht gestört und kann parallel betrieben werden.

- **Ausgabe einer Parameterabfrage im JSON-Format:**

  Siehe hierzu den entsprechenden Punkt unter 8.2 Spezialfunktionen.

- **Überprüfen von BSB-LAN auf nicht-freigegebene reglerspezifische CommandIDs:**

  Siehe hierzu den entsprechenden Punkt unter 8.2 Spezialfunktionen.

8.2 Spezialfunktionen

Der Übersichtlichkeit halber sind im Folgenden nochmals einige ausgewählte Spezialfunktionen bzw. spezielle URL-Befehle gesondert aufgeführt und beschrieben.

8.2.1 Raumtemperatur übermitteln

Mittels einer INF-Nachricht kann eine Raumtemperatur an den Regler gesendet werden, um einen Raumeinfluss bei der Berechnung der VL-Temperatur geltend zu machen.

Für die Raumtemperatur HK1[20] ist der Spezialparameter 10000, für den HK2[21] der Parameter 10001 zu nutzen.

Beispiel: Der URL-Befehl für den HK1, um eine Raumtemperatur von 19.5°C zu übermitteln, lautet: `http://<IP-Adresse>/I10000=19.5`

Hinweis:
Um diese Funktion zu nutzen, muss die Funktion ‚Raumeinfluss' vorher im Regler aktiviert und der Einflussfaktor prozentual festgelegt werden (Parameter 750 für HK1, Parameter 1050 für HK2).
Wird nur ein Temperaturwert als Einflussfaktor gemessen und übermittelt, ist die Temperaturmessung in einem Führungs- / Referenzraum zu empfehlen, in dem sich keinerlei weitere Wärmequelle (bspw. Kaminofen, große Fenster in Südlage etc.) befindet.

8.2.2 Präsenztaste simulieren

Die Funktion der Präsenztaste ist mit dem Spezialparameter 701 (für HK1) und 1001 (für HK2) implementiert und als SET-Befehl auszuführen. Die genannten Parameter müssen schreibbar sein (siehe Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software).

Bei aktivem Automatikprogramm ist dabei `http://<IP-Adresse>/S701=1` für den Wechsel auf ‚Betriebsart Reduziert' und `http://<IP-Adresse>/S701=0` für den Wechsel auf ‚Betriebsart Komfort' zu setzen.
Der jeweilige Wechsel ist bis zur nächsten Betriebsart-Umschaltung laut Zeitprogramm gültig.

Es ist jedoch bisher noch nicht umfangreich getestet, ob diese Funktion bei jedem Regler Wirkung zeigt oder nicht.

8.2.3 Manuellen TWW-Push ausführen

Bei einigen Reglern ist die (nahezu undokumentierte) Funktion eines manuellen Trinkwasser-Pushs[22] verfügbar. Um einen manuellen TWW-Push auszulösen, muss dazu die TWW-Taste an der ISR-Bedieneinheit gedrückt und für etwa drei Sekunden gehalten werden, bis im Display eine entsprechende Meldung erscheint.

Bei einigen Reglern kann diese Funktion mittels eines SET-Befehls erfolgen. Dieser lautet `http://<IP-Adresse>/S1601=1` – der Spezialparameter 1601 muss dazu schreibbar sein (siehe Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software).

20 Der Adapter muss evtl. als RGT1 konfiguriert sein, um HK1 zu bedienen.
21 Der Adapter muss evtl. als RGT2 konfiguriert sein, um HK2 zu bedienen.
22 Trinkwasserladung bei Bedarf auf Knopfdruck initiieren

8.2.4 Abrufen und Steuern mittels JSON

Hinweis: Diese Funktion ist derzeit noch in der (Weiter-)Entwicklung, es kann also noch Veränderungen hinsichtlich der Befehle und/oder Funktionen geben!

Parameterabfragen sowie das Setzen von Werten kann ebenfalls mittels JSON erfolgen.

- **Abfrage von Kategorien:**

  `http://<IP-Adresse>/JK=<xx>`
  Abfrage einer spezifischen Kategorie (<xx>=Kategorienummer)

  `http://<IP-Adresse>/JK=ALL`
  Abfrage aller Kategorien (samt Min. und Max.)

- **Abfragen und Setzen von Parametern per HTTP POST:**

  Hierbei ist der Aufruf der URL
  `http://<IP-Adresse>/JQ` für eine Abfrage und
  `http://<IP-Adresse>/JS` für das Setzen von Parametern zu verwenden.

  Folgende Parameter sind dabei möglich:

  `http://<IP-Adresse>/JQ`
  Senden: "Parameter"
  Empfangen: "Parameter", "Value", "Unit", "DataType" (0 = Zahl, 1 = ENUM, 2 = Wochentag, 3 = Stunde/Minute, 4 = Datum/Uhrzeit, 5 = Tag/Monat, 6 = String)

  `http://<IP-Adresse>/JS`
  Senden: "Parameter", "Value" (nur numerisch), "Type" (0 = INF, 1 = SET)
  Empfangen: "Parameter", "Status" (0 = Fehler, 1 = OK, 2 = Parameter read-only)

8.2.5 Überprüfen auf nicht-freigegebene reglerspezifische Command IDs

Hinweis: Es ist empfehlenswert, diese Abfrage einmalig auszuführen.

Diese Funktion ist nur bei aktiviertem Debug-Definement `#define DEBUG` in der Datei *BSB_lan_config.h* verfügbar!

`http://<IP-Adresse>/Q`

Diese Funktion geht alle Command IDs durch, die in der Datei *BSB_lan_defs.h* hinterlegt sind und schickt diejenigen, die nicht für den eigenen Reglertyp hinterlegt sind, als Anfrage-Parameter (Typ QUR, 0x06) an den Regler.
Das passiert bei Parametern, bei denen bisher nur eine Command ID bekannt ist, ständig und erzeugt die bekannten „error 7 (parameter not supported)“-Fehlermeldungen.

Sobald aber mehr als eine Command ID bekannt ist, bleibt die bisherige Command ID i.d.R. auf "DEV_ALL", ist also für alle Regler der Standard, und die neue Command ID wird erst einmal nur für die Therme freigeschaltet, die diese Command ID gemeldet hat.

Da es aber auch genauso gut umgekehrt sein kann, dass die "neue" Command ID der Standard ist, und die "alte" Command ID der Sonderfall, geht `/Q` nun die Command IDs durch, die nicht dem eigenen Regler zugewiesen sind. Häufig können dort noch eine Reihe "neuer" Parameter freigeschaltet werden.

Zur Info:
Es wird hierbei immer nur eine Anfrage mit einer Command ID an den Regler geschickt! Der Regler beantwortet diese entweder mit einer Fehlermeldung (Typ ERR, 0x08) oder einer Antwort mit einem Datenpaket (Typ ANS, 0x07).
In keinem Fall werden dabei Werte gesetzt oder Reglereinstellungen verändert! Dafür müsste ein ganz anderer Telegramm-Typ gesetzt werden (entweder Typ SET, 0x03 oder Typ INF, 0x02) - das macht `/Q` explizit nicht!

Wenn bereits alle Parameter für den Reglertyp bekannt und freigegeben sind, sieht die auf `http://<IP-Adresse>/Q`
folgende Webausgabe exemplarisch so aus:

```
Gerätefamilie: 90
Gerätevariante: 100
Start Test…
Test Ende.
```

Eine entsprechende Webausgabe bei bisher nicht-freigegebenen Parametern für den spezifischen Regler hingegen sieht exemplarisch so aus:

```
Gerätefamilie: 90
Gerätevariante: 100
Start Test...
2214
DC 86 00 0B 06 3D 0D 08 EB E7 3A
DC 80 06 0E 07 0D 3D 08 EB 00 0F 00 1B 94 5024
5024 Trinkwasserspeicher - TWW Schaltdifferenz 1 ein: error 7 (parameter not
supported)
DC 86 00 0B 06 3D 31 07 1D C8 19
DC 80 06 0E 07 31 3D 07 1D 00 01 40 A6 94 6031
6031 Konfiguration - Relaisausgang QX22 Modul 1: error 7 (parameter not
supported)
DC 86 00 0B 06 3D 05 07 86 E3 AE
DC 80 06 0D 07 05 3D 07 86 00 00 2C 55 8314
8314 Diagnose Erzeuger - Kesselrücklauftemperatur Ist: error 7 (parameter
not supported)
DC 86 00 0B 06 3D 11 05 1A 58 5A
DC 80 06 0E 07 11 3D 05 1A 01 00 00 91 09
Test Ende.
```

In diesem Fall sollte die Webausgabe bitte kopiert und gemeldet werden, damit eine entsprechende Anpassung vorgenommen werden kann.

9. Loggen von Daten

9.1 Verwendung des Adapters als Standalone-Logger mittels BSB-LAN

Stecke eine FAT32-formatierte microSD-Karte[23] in den Speicherkartenplatz des Ethernet-Shields, bevor du den Arduino einschaltest.
Aktiviere vor dem Flashen das Definement `#define LOGGER` in der Datei *BSB_lan_config.h*, füge die zu loggenden Parameter zur Variable `log_parameters` hinzu und bestimme das Logintervall mit der Variable `log_interval`. Bitte beachte auch die entsprechenden Punkte in Kapitel 8.1 Auflistung und Beschreibung der URL-Befehle.
Später können während der Laufzeit sowohl das Intervall als auch die Logging-Parameter mittels des Befehls "`/L=[Intervall],[Parameter1],...,[Parameter20]`" geändert werden.

Sämtliche Daten werden auf der Karte in der Datei *datalog.txt* im CSV-Format gespeichert und können somit leicht in Excel oder OpenOffice Calc importiert werden.
Der Dateiinhalt kann mit dem URL-Befehl "`/D`" eingesehen werden, eine graphische Darstellung der Logdateien erfolgt mittels "`/DG`".
Um die Datei *datalog.txt* zu löschen und neu zu erstellen, benutze den Befehl "`/D0`". Dies sollte ebenfalls bei der ersten Benutzung erfolgen, da hierdurch die Datei mit dem passenden CSV-Header initiiert wird.

Bitte beachte, dass der Arduino keine exakte Uhr ist. Auch wenn du bspw. das Intervall auf 60 Sekunden eingestellt hast, weicht die in der Datei dargestellte Zeit (welche von der Heizungssteuerung empfangen wird) möglicherweise davon ab - dies kann bis zu einer Sekunde pro Minute betragen.
Sollte eine exakte Logzeit unbedingt erforderlich sein, kannst du die durchschnittliche Zeitabweichung zwischen der Arduino-Zeit und der wirklichen Zeit ermitteln, das Log-Intervall entsprechend anpassen und bspw. 59 Sekunden anstatt 60 Sekunden einstellen.

9.2 Verwendung des Adapters als Remote-Logger

Neben dem Einsatz komplexer Systeme wie bspw. FHEM und den spezifischen Log-Lösungen kann bspw. folgender Befehl[24] periodisch ausgeführt werden (z.B. per cron job):

```
DATE=`date +%Y%m%d%H%M%S`; wget -qO- http://192.168.178.88/8310/720/710 |
egrep "(8310|720|710)" | sed "s/^/$DATE /" >> log.txt
```

Das aus diesem Beispiel resultierende Logfile '*log.txt*' enthält die aufgezeichneten Werte der Parameter 8310, 720 und 710.
Später kannst du das Logfile basierend auf den Parameternummern sortieren, nutze hierfür den Befehl '`sort`':

```
sort -k2 log.txt
```

23 Falls keine Karten >2GB erkannt werden, nutze eine kleinere Karte und formatiere sie mit FAT16.
24 Die IP, ggf. aktivierte optionale Sicherheitsfunktionen, die gewünschten Parameter etc. sind anzupassen.

10. Auslesen neuer Parameter-Telegramme

In diesem Kapitel wird die Vorgehensweise des Auslesens neuer Parameter-Telegramme beschrieben. Es ist dabei in eine ausführliche Beschreibung für Einsteiger, eine kurzgefasste Beschreibung für erfahrene Nutzer sowie eine Beschreibung des Implementierens neuer Command IDs für Nutzer mit Programmierkenntnissen aufgegliedert.

Um das Nachfolgende nicht in jedem Abschnitt aufzuführen, sei es hier einleitend erwähnt:

Sollte ein Parameter mehrere Einstellungsoptionen bieten, so wären idealerweise alle Optionen durchzugehen und die einzelnen Telegramme mit der eindeutigen Bezeichnung der jeweiligen Option zu notieren. Erfahrungsgemäß muss hierfür jedoch jede Option einmal mittels „OK“ der Bedieneinheit bestätigt und somit eingestellt werden, nach Beendigung sollte dann wieder die ursprüngliche Einstellung vorgenommen werden. Solltest du dir bzgl. des Verstellens bestimmter Reglereinstellungen hinsichtlich möglicher Folgen unsicher sein, belasse es bitte im Zweifelsfall beim Auslesen der aktuell eingestellten Option.
Für den Fall, dass mehrere Einstellungsoptionen verfügbar sind, ist es jedoch wichtig, dass in jedem Fall die aufgezählte Reihenfolge der Optionen mit der in der Bedieneinheit angezeigten Reihenfolge übereinstimmt und die aktuelle Einstellung notiert wird!

Nachdem das Auslesen der neuen Telegramme abgeschlossen ist, kann bei Bedarf (bspw. wenn Daten auf die microSD-Karte geloggt werden) der Verbositäts-Modus wieder deaktiviert werden (`http://<IP-Adresse>/V0`).

Rufe nun bitte zusätzlich einmal die Parameter 6220-6228 im Webinterface auf (`http://<IP-Adresse>/6220-6228`) und füge sie zu deinen Notizen hinzu, ebenso wie den Hersteller und die genaue Modellbezeichnung des Heizungssystems.

Ein Beispiel für eine verwertbare Auflistung neuer Parameter und den entsprechenden Zusatzangaben findest du im Kapitel 10.4 Beispiel für eine ‚Meldedatei‘.

Wenn du alles abgeschlossen hast, speichere deine Auflistung bitte als einfache .txt-Datei. Schicke die Datei nun entweder an bsb [ät] code-it.de oder melde dich im FHEM-Forum (https://forum.fhem.de/index.php/topic,29762.0.html).

Vielen Dank für deine Unterstützung!

10.1 Ausführliche Beschreibung des Auslesens neuer Telegramme (für Einsteiger)

Achtung: Bitte beachte auch die einleitenden Hinweise am Anfang des Kapitels!

Um die entsprechenden Telegramme neuer Parameter auszulesen, muss das Hardware-Setup (Arduino, LAN-Shield, Adapter und die Verbindung mit dem Heizungsregler) sowie die Installation der BSB-LAN-Software abgeschlossen und verwendungsfähig sein. Der Zugriff auf das Webinterface von BSB-LAN muss ebenfalls gegeben sein.

Zusätzlich zur bestehenden Verkabelung muss nun der Arduino mit deinem Laptop/PC via USB verbunden werden.

Starte dann die Arduino IDE und wähle unter „Menü Werkzeuge → Port“ den korrekten USB-Anschluss aus.
Überprüfe außerdem, ob unter „Menü Werkzeuge → Board → Arduino Mega 2560“ und unter „Menü Werkzeuge → Prozessor → Atmega 2560“ ausgewählt ist.

Starte nun den seriellen Monitor unter „Menü Werkzeuge → Serieller Monitor“ und überprüfe, ob unten rechts in der Fußzeile „115200 Baud“ eingestellt ist.

Aktiviere dann den Verbositäts-Modus mit dem URL-Befehl `http://<IP-Adresse>/V1` im Webinterface von BSB-LAN. Bereits jetzt werden schon einige Informationen und Telegramme im seriellen Monitor dargestellt.

Schalte als nächstes beim Heizungssystem über die integrierte Bedieneinheit zu dem Parameter, den du analysieren möchtest (mittels des Drehrades, der Pfeiltasten oder der spezifischen Eingabemöglichkeiten deiner Heizungssteuerung).

Warte auf 'Ruhe' auf dem Bus, dann schalte einen Parameter weiter vor und gleich wieder zurück zu dem Parameter, den du analysieren möchtest. Das Hin- und Herschalten soll nur sicherstellen, dass die letzte Nachricht auf dem Bus wirklich der Parameter ist, den du suchst.
Nun solltest du das spezifische Parameter-Telegramm in der Ausgabe des seriellen Monitors sehen.
Solange in der Bedieneinheit der aufgerufene Parameter angezeigt wird, kommt in etwa zehnsekündigem Abstand regelmäßig das entsprechende Telegramm über den Bus.
Aber Achtung: Der Heizungsregler sendet zwischendurch automatisch auch andere Status-Telegramme (die sogenannten ‚broadcasts‘) - sei daher bitte aufmerksam beim Auslesen des gewünschten neuen Parameter-Telegramms!

Beispiel:

```
DISP->HEIZ QUR      113D305F
DC 8A 00 0B 06 3D 11 30 5F AB EC

HEIZ->DISP ANS      113D305F 00 00
DC 80 0A 0D 07 11 3D 30 5F 00 00 03 A1

DISP->HEIZ QUR      113D3063
DC 8A 00 0B 06 3D 11 30 63 5C 33

HEIZ->DISP ANS      113D3063 00 00 16
DC 80 0A 0E 07 11 3D 30 63 00 00 16 AD 0B
```

Die ersten vier Zeilen in obigem Beispiel sind von dem Parameter, zu dem hingeschaltet wurde. Die letzten vier Zeilen (hier im Beispiel kursiv dargestellt) stammen von dem Parameter, den du nun analysieren möchtest.

QUR bedeutet Anfrage, ANS die zugehörige Antwort. Anstelle von DISP wird eventuell RGT1 oder RGT2 angezeigt, dies ist abhängig vom jeweiligen Gerät, mit dem du die Eingaben am Heizungssystem tätigst (integrierte Steuerung oder angeschlossenes Raumgerät/Fernbedienung) und nicht weiter von Interesse.

Hast du nun also das betreffende Telegramm entdeckt, bietet es sich an, in der Fußzeile des seriellen Monitors unten links den Haken bei „Autoscroll“ durch einen Klick auf denselben zu entfernen. Somit bleibt die Auflistung des seriellen Monitors an dieser Stelle stehen und wird nicht durch weitere eintreffende Telegramme immer wieder verschoben. Nun ist es auch möglich, mit copy&paste zu arbeiten.

Markiere und kopiere nun also das entsprechende Frage- und Antworttelegramm (also die genannten letzten vier Zeilen in obigem Beispiel) und füge sie in eine Textdatei ein.
Hinweis: Copy&paste funktioniert im seriellen Monitor erfahrungsgemäß nicht immer zuverlässig, also überprüfe bitte jedes Mal beim Einfügen in die Textdatei, ob es sich auch wirklich um das gewünschte Telegramm handelt.

WICHTIG:
Schreibe nun *zusätzlich* zum Telegramm die entsprechende Parameternummer, die genaue Parameterbezeichnung sowie ggf. die jeweilige Werte-Einheit hinzu!

Notiere bitte außerdem die jeweils angezeigte aktive Einstellung bzw. den im Moment der Abfrage angezeigten Wert!
Dies ist zwingend notwendig, da ohne die aufgezählten zusätzlichen Informationen das reine Telegramm nutzlos ist!

Wenn du einen weiteren Parameter auslesen möchtest, wiederhole das beschriebene Procedere entsprechend. Es bietet sich an, vorher wieder den Haken bei „Autoscroll“ zu setzen, bis du den nächsten zu notierenden Parameter aufgerufen hast.

Wenn du sämtliche neuen Parameter (und/oder Einstellungsoptionen neuer Parameter) samt Telegrammen und Beschreibungen etc. notiert und das Auslesen somit beendet hast, kannst du den seriellen Monitor schließen.

10.2 Kurze Beschreibung des Auslesens neuer Telegramme (für erfahrene Nutzer)

Achtung: Bitte beachte auch die einleitenden Hinweise am Anfang des Kapitels!

Bei angeschlossenem Adapter und eingeschaltetem Regler die Arduino IDE und den seriellen Monitor starten, dann den Verbositäts-Modus des Adapters aktivieren (`/V1`).
Den betreffenden Parameter an der Regler-Bedieneinheit aufrufen und das zugehörige Telegramm im seriellen Monitor mittels copy&paste in eine Textdatei kopieren.

WICHTIG:
Schreibe nun *zusätzlich* zum Telegramm die entsprechende Parameternummer, die genaue Parameterbezeichnung sowie ggf. die jeweilige Werte-Einheit hinzu!
Notiere bitte außerdem die jeweils angezeigte aktive Einstellung bzw. den im Moment der Abfrage angezeigten Wert!
Dies ist zwingend notwendig, da das reine Telegramm ohne die aufgezählten zusätzlichen Informationen nutzlos ist!

Fahre so für jeden weiteren Parameter fort.

10.3 Implementieren neuer Command IDs (für Programmierer)

Das folgende Procedere sei nur Anwendern empfohlen, die Programmierkenntnisse haben und die neuen Command IDs eigenständig in die Datei *BSB_lan_defs.h* implementieren möchten.

Einsteigern wird ausdrücklich die bereits ausführlich beschriebene Vorgehensweise des Auslesens empfohlen, bei der die neuen Telegramme letztlich nur ‚gemeldet' werden!

Zum Auslesen der Telegramme ist prinzipiell die unter 10.2 Kurze Beschreibung des Auslesens neuer Telegramme (für erfahrene Nutzer) beschriebene Vorgehensweise zu empfehlen.

Erläuterung der Struktur der Datentelegramme anhand des folgenden Beispiels:

```
HEIZ->DISP ANS      113D3063 00 00 16
DC 80 0A 0E 07 11 3D 30 63 00 00 16 AD 0B
```

Byte 1:	Immer 0xDC (Startbyte, Beginn des Telegramms)
Byte 2:	Quellgerät plus 0x80 (Quellgeräte: 0x00 = Heizungssystem, 0x06 = Raumgerät 1, 0x07 = Raumgerät 2, 0x0A = Display, 0x7F = Broadcast)
Byte 3:	Zielgerät (gleiche Adresswerte wie bei Quellgerät)
Byte 4:	Telegrammlänge (Startbyte des Telegramms wird nicht mitgezählt)
Byte 5:	Nachrichtentyp (0x02 = INF/Info, 0x03 = SET/setzen, 0x04 = ACK/ akzeptiert, 0x05 = NACK/nicht akzeptiert, 0x06 = QUR/Abfrage, 0x07 = ANS/Antwort, 0x08 = ERR/Fehler)
Byte 6-9:	**Command ID (→ diese ist es, die wir brauchen!)**
Byte 10-x:	Payload data (optional)
Letzte zwei Bytes:	CRC-Checksumme

1. Das Datentelegramm im obigen Beispiel hat die Command ID 0x113D3063.
   Bitte beachte, dass die ersten beiden Bytes der Command ID beim Nachrichtentyp "Abfragen" (QUR / 0x06) vertauscht sind! Stelle daher bitte sicher, dass du stets auf das richtige Telegramm achtest: Typ "Antwort" (ANS / 0x07).
2. Suche den Bereich "*global command table*" in der Datei *BSB_lan_defs.h* und überprüfe, ob für diesen Befehl bereits ein Eintrag existiert (suche nach STRxxxx , wobei xxxx die Parameternummer darstellt). Falls es ihn bereits gibt, fahre fort mit Schritt 4.
3. Sollte der Parameter noch nicht in dem "*global command table*" gelistet sein, musst du einen Eintrag im Bereich "*menu strings*" wie folgt erstellen:

   ```
   const char STRxxxx[] PROGMEM = "Parameter_Name_or_Function";
   ```

   Nun kopiere eine Zeile des Bereichs vom "*global command table*", wo der neue Parameter numerisch passt. Fahre fort mit Schritt 4, aber anstatt `CMD_UNKNOWN` zu ersetzen, müssen logischerweise die Command ID und der Wertetyp der kopierten Zeile ersetzt werden.
4. Ersetze `CMD_UNKNOWN` durch die Command ID, die du gerade herausgefunden hast. Falls der zurückgegebene Wertetyp (column 3) `VT_UNKNOWN` ist, versuche herauszufinden, welcher Parametertyp von der Liste am Anfang der Datei passt.

Beispiel:
Wenn der Parameter einen Temperaturwert zurückgeben soll, kannst du `VT_TEMP`, `VT_TEMP_SHORT`, `VT_TEMP_SHORT5` oder `VT_TEMP_WORD` ausprobieren.
Für Parameter, die mehrere Optionen bereitstellen, musst du eine entsprechende Zeile im Abschnitt "*ENUM tables*" hinzufügen.

5. Wenn der Adapter den gleichen Wert ausgibt wie auf dem Display der integrierten Heizungssteuerung dargestellt, hast du den richtigen Wert gefunden und der Parameter ist nun voll funktionsfähig (bspw. sollte das Abfragen und Setzen von Werten nun funktionieren). Herzlichen Glückwunsch!
6. Wenn du fertig bist, überprüfe nochmals, ob die neue Command ID nicht bereits irgendwo anders in *BSB_lan_defs.h* verwendet wird (wenn du bspw. nach der Command ID suchst, sollte sie nur einmal gefunden werden).
   Da das BSB-Protokoll anscheinend nicht standardisiert ist und die unterschiedlichen Hersteller (zumindest bei spezifischeren Parametern) keine Rücksicht darauf zu nehmen scheinen, wie andere Hersteller die Command IDs verwenden, ist es möglich, dass gleiche Command IDs für verschiedene Parameter bei verschiedenen Heizungssystemen existieren. Sollte es passieren, dass eine Command ID nun doppelt in der Datei *BSB_lan_defs.h* existiert, markiere diese bitte deutlich bevor du uns das Update schickst und teile uns bitte mit, welches Heizungssystem genau du verwendest (`/6220-6228`). Wir werden dann bedingungsabhängige Compiler-Flags hinzufügen, so dass das Heizungssystem X anders kompiliert wird als Heizungssystem Y und somit letztlich beide Systeme die mehrdeutigen Command IDs für die korrekten Parameter nutzen können.
7. Bitte sende die neuen bzw. aktualisierten Zeilen an bsb [ät] code-it.de oder poste sie im FHEM-Forum (https://forum.fhem.de/index.php/topic,29762.0.html).

   Solltest du eine Diff-Datei nutzen, überprüfe bitte vor dem Erstellen nochmals, ob du wirklich die aktuelle *BSB_lan_defs.h* des GitHub-Repositoriums heruntergeladen hast, da manchmal die Dateien aktualisiert werden, ohne dass direkt eine neue Version veröffentlicht wird.

10.4 Beispiel für eine ‚Meldedatei'

Hier ein Beispiel für eine erstellte ‚Meldedatei', die alle notwendigen Informationen für eine weitere Verarbeitung und Implementierung der neuen Parameter enthält:

```
Brötje NovoCondens SOB 26 C (Öl)

6220 Konfiguration - Software- Version: 1.3
6221 Konfiguration - Entwicklungs-Index: error 7 (parameter not supported)
6222 Konfiguration - Gerätebetriebsstunden: 12345 h
6223 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014
6224 Konfiguration - Geräte-Identifikation: RVS43.222/100
6225 Konfiguration - Gerätefamilie: 96
6226 Konfiguration - Gerätevariante: 100
6227 Konfiguration - Objektverzeichnis-Version: 1.0
6228 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014


Parameter 2270 Kessel – Rücklaufsollwert Minimum °C
→ wird vom Arduino/BSB bei Abfrage mit 60°C angezeigt,
angezeigter Ist-Wert laut RGT-Bedieneinheit: 8°C
RGT1->HEIZ QUR      053D0908
DC 86 00 0B 06 3D 05 09 08 B0 E7
HEIZ->RGT1 ANS      053D0908 00 02 00
DC 80 06 0E 07 05 3D 09 08 00 02 00 4B 02


Parameter 5010 Trinkwasserspeicher – Ladung
Mögliche Parameteroptionen: [Einmal/Tag | Mehrmals/Tag]
Ist: Mehrmals/Tag
RGT1->HEIZ QUR      253D0737
DC 86 00 0B 06 3D 25 07 37 D2 92
HEIZ->RGT1 ANS      253D0737 00 FF
DC 80 06 0D 07 25 3D 07 37 00 FF CE 62


Parameter 5050 Trinkwasserspeicher – Ladetemperatur Maximum °C
Mögliche Einstelloptionen: [8°C - 90°C]
Ist: 60°C
RGT1->HEIZ QUR      253D08A3
DC 86 00 0B 06 3D 25 08 A3 01 91
HEIZ->RGT1 ANS      253D08A3 00 0F 00
DC 80 06 0E 07 25 3D 08 A3 00 0F 00 0D 90
```

11. Nutzung von externen Programmen

Da der Adapter lediglich eine Schnittstelle darstellt, mittels derer man Zugriff auf den Heizungsregler via Computer erhält, können selbstverständlich externe Programme zum Einsatz kommen. Somit kann die Heizungssteuerung in komplexe Heimautomationssysteme eingebunden werden. Auch das Erstellen von umfassenden Logdateien und deren grafische Aufbereitung kann auf diese Weise erfolgen.

Da es hierfür verschiedene Softwarelösungen gibt, kann hier weder eine umfangreiche Vorstellung der verschiedenen Systeme erfolgen, noch eine grundsätzliche Anleitung zur Integration des Adapters oder einer Heizungsregelung per se gegeben werden. Sollte jedoch bereits eines der nachfolgend erwähnten Systeme zum Einsatz kommen, so sind die folgenden Beispielkonfigurationen hoffentlich eine kleine Starthilfe, um die eigenen Vorstellungen umzusetzen.

Sollten Fragen bzgl. der folgenden Beispiele oder anderer Konfigurationsmöglichkeiten auftreten, so ist bitte von diesbezüglichen Anfragen abzusehen – diese sollten mittels entsprechender Literatur oder in den jeweils spezifischen Foren geklärt werden.

Hinweis:
Selbstverständlich müssen stets sowohl die IP als auch -falls aktiviert- die optionalen Sicherheitsfunktionen bei den folgenden Beispielen entsprechend angepasst werden. Ebenso müssen Parameter, die gesetzt werden sollen, schreibbar sein (siehe Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software).

Solltest du ein anderes System als die im Folgenden aufgeführten verwenden, so würde ich mich über die Zusendung eines Beispielscripts zur Anbindung des Adapters freuen! Sende es mir dazu einfach als .txt-Datei an adapter [ät] quantentunnel.de – danke!

11.1 FHEM

11.1.1 Einbindung mittels BSB-LAN-Modul

Derzeit wird vom FHEM-Forumsmitglied „justme1968“ ein BSB-LAN-Modul für FHEM entwickelt: https://forum.fhem.de/index.php/topic,84381.0.html
Vielen Dank!

UPDATE: Eine erste (Test-)Version ist bereits verfügbar!

11.1.2 Einbindung mittels HTTPMOD-Modul

Die FHEM-Beispielscripte stammen vom FHEM-Forumsmitglied „freetz“.
Vielen Dank!

Um auf die Webschnittstelle des Adapters zuzugreifen, kann das Modul HTTPMOD in FHEM genutzt werden.

Beispielscript für eine Parameterabfrage und die Übermittlung einer Raumtemperatur:

Der Beispielcode fragt die Parameter 8700, 8743 und 8314 alle 300 Sekunden ab und weist diese dem Gerät "THISION" (Name des Heizungssystems) und den Readings "Aussentemperatur", "Vorlauftemperatur" und "Ruecklauftemperatur" zu.

Darüber hinaus stellt es ein Reading "Istwert" bereit, das per FHEM gesetzt werden kann, um dem Heizungssystem die aktuelle Zimmertemperatur mitzuteilen (Parameter 10000).

Zu guter Letzt berechnet es die Differenz zwischen "Vorlauftemperatur" und "Rücklauftemperatur" und weist diese Differenz dem Reading "Spreizung" zu.

Bitte beachte: Die Regex-Bedingungen müssen vom Beginn des Strings an (also der Parameternummer wie bspw. 8700) matchen und nicht erst ab einem späteren Teil des Strings.

```
define THISION HTTPMOD http://192.168.178.88/8700/8743/8314 300
attr THISION userattr reading0Name reading0Regex reading1Name reading1Regex
reading2Name reading2Regex reading0Expr set0Name set0URL
attr THISION event-on-change-reading .*
attr THISION reading0Name Aussentemperatur
attr THISION reading0Regex 8700 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading1Name Vorlauftemperatur
attr THISION reading1Regex 8743 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading2Name Ruecklauftemperatur
attr THISION reading2Regex 8314 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading0Expr $val=~s/[\r\n]//g;;$val
attr THISION set0Name Istwert
attr THISION set0URL http://192.168.178.88/I10000=$val
attr THISION timeout 5
attr THISION userReadings Spreizung
{ sprintf("%.1f",ReadingsVal("THISION","Vorlauftemperatur",0)-
ReadingsVal("THISION","Ruecklauftemperatur",0));; }
```

Beispielscript für die Abfrage und Ansteuerung eines Relaisboards:

Das Folgende ist ein Beispiel für eine FHEM-Konfiguration, bei dem die drei Relais-Ports namens "Heater", "Fan" und "Bell" abgefragt und gesteuert werden, die an die entsprechenden GPIO-Pins 7, 6 und 5 angeschlossen sind.

```
define EthRelais HTTPMOD http://192.168.178.88/GO5/GO6/GO7 30
attr EthRelais userattr reading0Name reading0Regex reading1Name
reading1Regex reading2Name reading2Regex reading0Expr reading0Map set0Name
set0URL set1Name set1URL set2Name set2URL setIMap setParseResponse:0,1
setRegex
attr EthRelais event-on-change-reading .*
attr EthRelais reading0Name Heater
attr EthRelais reading0Regex GPIO7:[ \t](\d)
attr EthRelais reading1Name Fan
attr EthRelais reading1Regex GPIO6:[ \t](\d)
attr EthRelais reading2Name Bell
attr EthRelais reading2Regex GPIO5:[ \t](\d)
attr EthRelais room Heizung
attr EthRelais set0Name Heater
attr EthRelais set0URL http://192.168.178.88/GO7=$val
attr EthRelais set1Name Fan
attr EthRelais set1URL http://192.168.178.88/GO6=$val
attr EthRelais set2Name Bell
attr EthRelais set2URL http://192.168.178.88/GO5=$val
attr EthRelais setParseResponse 1
attr EthRelais setRegex GPIO[0-9]+:[ \t](\d)
attr EthRelais timeout 5
```

11.2 openHAB

Die openHAB-Beispielscripte stammen vom FHEM-Forumsmitglied „acfischer42“. Vielen Dank!

Es existiert derzeit kein komplettes Binding fuer BSB-LAN. Mit dem HTTP Binding[25] und der Javascript Transformation[26] ist es allerdings durchaus möglich, Werte auszulesen und auch zu schreiben.

Ein Loggen der Daten kann bspw. mit InfluxDB erfolgen, die Visualisierung mit Grafana. Selbstverständlich sind die notwendigen Addons wie bspw. die Javascript Transformation vorhergehend zu installieren.

Beispiel einer Item-Konfiguration:

```
Number hz_aussentemp "Aussentemperatur [%.1f °C]" <temperature> (Heizunglog)
{ http="<[http://192.168.178.88/8700:60000:JS(bsbinput.js)]" }
String hz_700 "Heizkreis 1 Betriebsart [%s]" <temperature> (Heizunglog)
{ http="<[http://192.168.178.88/700:1000:JS(bsbinput_string.js)]" }
```

Das folgende Javascript ist als *bsbinput.js* im Verzeichnis *transformations* abzulegen.

Beispielscript für Abfragen von Parametern, bei denen ein Wert ausgegeben wird:

```
(function(i) {
    var outputres;
      var results = [];
      value1 = i;
      // define regex to search for the line in the http response
      var regEx = '<input type=text id=\'value1\' VALUE=\'[-]*[0-9]+\.[0-
9]+';
      var re = new RegExp(regEx, 'gim');

 do {
    match = re.exec(value1);
    if (match){
        results.push(match[0]);
    }
} while (match);

      outputres = results[0]
      //extract actual value from the output
      var output=outputres.substr(outputres.indexOf("VALUE='")
+7,outputres.length);
      return output;
})(input)
```

25 Siehe Dokumentation unter https://docs.openhab.org/addons/bindings/http1/readme.html
26 Siehe Dokumentation unter https://docs.openhab.org/addons/transformations/javascript/readme.html

Beispielscript für direkte Abfragen von enum-Werten:

```
(function(i) {
        var outputres;
      var results = [];
      value1 = i;
      // define regex to search for the line in the http response
      var regEx = '<option value=\'[0-9]+\' SELECTED>.*</option>';
      var re = new RegExp(regEx, 'gim');

 do {
    match = re.exec(value1);
    if (match){
        results.push(match[0]);
    }
} while (match);

      outputres = results[0]
      //extract actual value from the output
      var l=outputres.indexOf("</o")-outputres.indexOf(">")-1
      var output=outputres.substr(outputres.indexOf(">")+1,l);
           return output;
})(input)
```

Das Schreiben von Daten erfolgt über Rules:

```
rule "RoomTemp"

when
      Item iSet_temp changed
then
      sendHttpGetRequest("http://192.168.178.88/I10000=" +
iSet_temp.state.toString)
end
```

11.3 HomeMatic (EQ3)

Die HomeMatic-Beispielscripte stammen vom FHEM-Forumsmitglied „PaulM“.
Weitere Beispielscripte von ihm sind im FHEM-Forum zu finden:
https://forum.fhem.de/index.php/topic,29762.msg769167.html#msg769167
Vielen Dank!

Zur Einbindung in HomeMatic bietet sich die Verwendung von CuxD und wget an.

Beispiel zur Abfrage des Parameters ‚8326 Brennermodulation‘ mittels CuxD:

```
! Skriptanfang:
! BSB-Abfrage
string url='http://192.168.178.88/8326'; !IP anpassen
! WriteLine("URL: " # url); !nur Kontrolle
! --timeout=seconds  ! während der Dauer der Abfrage werden von der CCU
! keine anderen Skripte ausgeführt !!!
! siehe CUxD-Handbuch 5.8.2 System.Exec
! dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
! ("wget -t 5 -T 20 -q -O - '"# url #"'");  ! abgekürzte Wget Syntax
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget --tries=5
--timeout=20 --quiet --output-document=- '"# url #"'");
! ausführliche Wget Syntax
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("Antwort: " # stdout);
! Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
! ab hier kommen auswertbare Informationen
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
! wenn Rückmeldung mit allen Daten angezeigt werden soll,
! Ausrufezeichen nächste Zeile entfernen
! WriteLine("Antwort ohne Vorspann: " # stdout);
string wort = "8326"; !Brennermodulation
integer laenge = wort.Length();
! WriteLine("laenge: " # laenge); zum Testen für andere Parameter
! für Skripttest Ausrufezeichen entfernen
```

```
integer pos = stdout.Find(wort);
! WriteLine("pos:" #pos);
pos = pos + 39; !bei anderen Parametern meist zwischen 25 und 50
string daten = stdout.Substr((pos + laenge +1), 100);
! WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
! keine Prüfung, da auch 0 sein kann
dom.GetObject("H_Brennermodulation").State(zahl);
WriteLine("H_Brennermodulation: "#zahl);
! nur für Chart CUxD
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State
("H_Brennermodulation"#";"#zahl);
WriteLine("Hallo Welt!");
! Skriptende:
```

Beispiel zum Setzen der Betriebsart auf Komfort-Betrieb mit ‚S700=3' mittels CuxD:

```
! Skriptanfang:
! Heizung KOMFORT (=AN - keine Nachtabsenkung)
! Anweisung senden
string urlset ='http://192.168.178.88/S700=3'; !IP anpassen
WriteLine("Befehl: " # urlset);
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
("wget -t 5 -T 20 -q -O - '"# urlset #"'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! Kontrollabfrage
string url='http://192.168.178.88/700'; !IP anpassen
! WriteLine("URL: " # url);
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
("wget -t 5 -T 20 -q -O - '"# url #"'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("Antwort: " # stdout);
! Beim stdout den ueberfluessigen Vorspann entfernen
```

```
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
! wenn Rückmeldung mit allen Daten angezeigt werden soll,
! Ausrufezeichen nächste Zeile entfernen
! WriteLine("Antwort ohne Vorspann: " # stdout);
string wort = "700"; !Heizbetrieb
integer laenge = wort.Length();
! WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
! WriteLine("pos:" #pos);
pos = pos + 28;
! WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 100);
! WriteLine("daten :"#daten);
string substring = daten.Substr(0, 1);
integer zahl = substring.ToInteger();
WriteLine("aktueller Heizbetrieb (0 bis 3): " # zahl);
if (zahl == 0) {dom.GetObject('H_Heizbetrieb').State("Heizung AUS");}
if (zahl == 1) {dom.GetObject('H_Heizbetrieb').State("Heizung Automatik");}
if (zahl == 2) {dom.GetObject('H_Heizbetrieb').State("Heizung
Nachtabsenkung");}
if (zahl == 3) {dom.GetObject('H_Heizbetrieb').State("Heizung Komfort");}
! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Heizbetrieb"#";"#zahl);
! alle Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
! Programmnamen ggf. anpassen
programObj.ProgramExecute();
WriteLine("Hallo Welt!");
! Skriptende:
```

12. Verwendung optionaler Sensoren: DHT22 und DS18B20

Es besteht die Möglichkeit, zusätzliche Sensoren des Typs DS18B20 (OneWire-Temperatursensor) und DHT22 (Temperatur- und Feuchtigkeitssensor) direkt an bestimmte Pins des Adapters bzw. Arduino anzuschließen. Die entsprechenden Bibliotheken für die Arduino IDE sind bereits im Softwarepaket des Adapters integriert.

Der Anschluss der Sensoren kann i.d.R. an GND und +5V des Adapters (unter zusätzlicher Verwendung der fühlerspezifischen PullUp-Widerstände) stattfinden.

Zur Nutzung dieser Sensoren muss lediglich die Konfiguration in der Datei *BSB_lan_config.h* entsprechend angepasst werden: Es sind die jeweiligen Definements zu aktivieren und die für DATA genutzten Digitaleingänge bzw. Pins festzulegen.
Siehe hierzu auch Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software.

Auf die Daten der Sensoren kann nach erfolgter Installation über das Webinterface (jeweilige Links im oberen Bereich) oder mittels der URL-Befehle `/T` (für DS18B20) und `/H` (für DHT22) zugegriffen werden.

Sollen die gemessenen Werte geloggt werden oder sind 24h-Mittelwertsberechnungen gewünscht, so kann dies mit den jeweiligen Anpassungen in der Datei *BSB_lan_config.h* ganz einfach realisiert werden.

Tipp:

Werden DS18B20- und/oder DHT22-Sensoren verwendet, werden diese unter
`http://<IP-Adresse>/ipwe.cgi`
standardmäßig mit angezeigt.
Dabei wird neben den gemessenen Werten auch die jeweils spezifische Hardwarekennung der Sensoren aufgeführt. Dies ist besonders bei einer Ersteinrichtung für eine eindeutige Unterscheidung der einzelnen Sensoren hilfreich.
Voraussetzung ist, dass das ipwe-Definement in der Datei *BSB_lan_config.h* aktiviert ist (siehe Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software).

Auf die näheren Spezifikationen und die elektrische Installation dieser beliebten Messkomponenten wird an dieser Stelle nicht weiter eingegangen, es wird hiermit lediglich auf die üblichen Informationsquellen verwiesen. Dennoch seien im Folgenden ein paar hilfreiche Tipps erwähnt, die den Einsatz und Betrieb zuverlässiger gestalten.

12.1 Hinweise zu DHT22-Temperatur-/Feuchtigkeitssensoren

Bitte beachte: Kommen mehrere DHT22-Sensoren zum Einsatz, so muss für jeden DATA-Anschluss ein eigener Pin am Arduino genutzt und in der Datei *BSB_lan_config.h* definiert werden.

12.2 Hinweise zu DS18B20-Temperatursensoren

DS18B20-Sensoren sind (neben der üblichen Bauart) auch in wasserdicht gekapselten Ausführungen mit verschiedenen Kabellängen erhältlich. Diese Ausführung macht den Einsatz gerade im Bereich der Heizungssteuerung sehr interessant, da hiermit schnell und kostengünstig eine individuelle Installation für diverse Temperaturmessungen realisiert werden kann.

Tipps für die elektrische Installation:

Kommen mehrere DS18B20-Sensoren und/oder größere Leitungslängen zum Einsatz, hat es sich bewährt, je einen 100nF-Keramikkondensator (und ggf. noch einen 10µF-Tantalkondensator zusätzlich) möglichst nah am Sensor in die Leitung zwischen GND und VCC (+5V) zu positionieren, um einen Spannungsabfall bei der Abfrage zu kompensieren.

Der Wert des PullUp-Widerstandes am Adapterausgang zwischen DATA und VCC (+5V) ist für einen problemlosen Betrieb u.U. kleiner als die üblicherweise empfohlenen 4,7kΩ zu wählen.

Von der Verwendung des sogenannten ‚parasitären Modus' ist abzuraten.
Die Verwendung einer geschirmten Steuerleitung ist zu empfehlen.
Um etwaige von der Versorgungsspannung des Arduino-Netzteils ausgehende Störeinflüsse zu minimieren, kann die Zuleitung der Stromversorgung arduinoseitig etwa vier bis fünfmal durch einen Ferritring geführt werden.

Tipps für die Verwendung im Bereich der Heizungsinstallation:

Sollen die Sensoren für Temperaturmessungen an Rohren zum Einsatz kommen (bspw. HK-VL/-RL), so ist es empfehlenswert, ein Bett aus Wärmeleitpaste für den Kontaktbereich zu verwenden.
Darüber hinaus haben Tests gezeigt, dass die Positionierung nach einem Knick an der Außenseite eines Rohres ideal zu sein scheint, da hier die Kerntemperatur des Strömungsmediums aufgrund der auftretenden Verwirbelungen nah an die Rohrwand gelangt.
Die Metallhülse der gekapselten Bauform sollte möglichst mit einer metallenen Rohrschelle am Rohr fixiert werden. Das Kabel selbst sollte zusätzlich mit einem Kabelbinder fixiert werden, um Zugkräfte an der Fühlerhülse sowie ein Verrutschen des Fühlers zu vermeiden.
Die Rohrdämmung sollte nach Anbringen des Fühlers (unter der Dämmung) wieder gewissenhaft verschlossen werden. Löcher, Einschnitte o.ä. in Fühlernähe sind zu vermeiden. Werden Fühler an bisher ungedämmten Rohren montiert, so ist zumindest für den Bereich des Fühlers eine zusätzliche Rohrisolierung empfehlenswert, um Messwertverfälschungen durch bspw. Raum- oder Zugluft zu vermeiden.

Kommen die Fühler in Tauchhülsen oder Klemmschienen zum Einsatz, ist ggf. auch hier die Verwendung von zusätzlicher Wärmeleitpaste zu empfehlen.

Im Allgemeinen sollten die Fühler etwa ein bis zwei Meter von einer zusätzlichen Wärmequelle (wie bspw. Heizkessel, Pufferspeicher o.ä.) entfernt montiert werden.

Bitte beachte: Bereits installierte Fühler (bspw. in Tauchülsen von Mischern, Pufferspeichern etc.), die bereits an einen Heizungs- oder Solarregler angeschlossen sind, haben immer Vorrang! Keinesfalls sollte deren Installation oder der Kontakt mit dem zu messenden Element durch eine zusätzliche Montage von DS18B20-Sensoren leiden!

13. Etwaige Fehlermeldungen und deren mögliche Ursachen

13.1 Fehlermeldung „unknown type <xxxxxxxx>“

Dieser Fehler sagt aus, dass für diesen Parameter keine Umrechnungsanweisung vorliegt, um die Rohdaten in eine entsprechende Einheit (Zeit, Temperatur, Prozent, Druck etc.) umzuwandeln.

Um den Fehler zu beheben, sollte das jeweilige Telegramm / die Command ID des betreffenden Parameters sowie der zugehörige Wert ausgelesen und gemeldet werden. Sollten mehrere Einstellungsoptionen für einen Parameter verfügbar sein, muss zusätzlich jede Option ausgelesen werden, damit eine eindeutige Zuordnung stattfinden kann.

13.2 Fehlermeldung "error 7 (parameter not supported)"

Die zugehörige Command ID wird nicht erkannt oder der entsprechende Parameter wird vom Regler nicht unterstützt (bspw. spezifische Parameter, die eine Gasheizung betreffen und bei einer Ölheizung dementsprechend nicht verfügbar sind).

Fehlermeldungen dieses Typs werden seit v0.41 der Übersichtlichkeit halber per default ausgeblendet (die entsprechenden Parameter bspw. bei einer Komplettabfrage aber dennoch abgefragt). Möchtest du sie dennoch angezeigt bekommen, so ist das entsprechende Definement `#define HIDE_UNKNOWN` in der Datei *BSB_lan_config.h* auszukommentieren (`//#define HIDE_UNKNOWN`).

Zur Überprüfung, ob die CommandID vom Regler prinzipiell unterstützt wird, jedoch für diese Gerätefamilie nicht freigegeben ist, aktiviere bitte vorübergehend das Definement `#define DEBUG` in der Datei *BSB_lan_config.h* und führe dann den Befehl `/Q` aus. Siehe hierzu auch 8.2.5 Überprüfen auf nicht-freigegebene reglerspezifische Command IDs. Bei diesem Befehl werden trotz aktivem ‚HIDE_UNKNOWN‘-Definement etwaige error7-Fehlermeldungen angezeigt.

13.3 Fehlermeldung "query failed"

Diese Meldung erscheint, wenn auf die Anfrage des Adapters keine (sinnvolle) Antwort des Reglers kommt.

Mögliche Ursachen sind meist hardwareseitig zu suchen (bspw. fehlerhafte RX- und/oder TX-Verbindung, falsch verbaute Komponenten oder auch ein timeout aufgrund eines ausgeschalteten oder nicht angeschlossenen Reglers).

14. Etwaige Probleme und deren mögliche Ursachen

14.1 Die rote LED des Adapters leuchtet nicht

- Regler ist ausgeschaltet
- Adapter ist nicht mit dem Regler via BSB oder LPB verbunden
- Adapter ist falsch mit dem Regler verbunden (CL+/CL- bzw. DB/MB vertauscht)
- Evtl. Hardwarefehler des Adapters (bspw. defektes Bauteil, Fehler im Aufbau)
- Evtl. Wackelkontakt beim Bus-Anschluss (Rx/Tx oder CL+/CL-)

14.2 Die rote LED leuchtet, aber es ist keine Abfrage möglich

- Evtl. Adapter falsch angeschlossen (an G+ statt an CL+)
- Evtl. Wackelkontakt beim Busanschluss (Rx/Tx oder CL+/CL-)
- Siehe Punkt „Keine Parameterabfrage möglich“

14.3 Zugriff auf das Webinterface nicht möglich

- Adapter hat keine Stromversorgung
- Adapter ist nicht mit dem LAN verbunden
- IP- und/oder MAC-Adresse des Adapters ist nicht korrekt
- Sicherheitsfunktionen „`Passkey`“, „`TRUSTED_IP`“ und/oder „`USER_PASS_B64`“ aktiviert/deaktiviert → URL nicht angepasst, Zugriff von falscher IP etc.
- → Testweise Reset-Knopf des Arduino bzw. LAN-Shields drücken
- LAN-Shield mit W5100-Chip bestückt? Sollte ein W5500 zum Einsatz kommen, sind die entsprechenden Punkte bei der Installation zu berücksichtigen (s. Kap. 5 & 14.10)
- Adapter und/oder Arduino fehlerhaft (→ vereinzelt kam es zu diffusen Problemen bei der Verwendung von günstigen ‚China‘-Arduinos)

14.4 Keine Parameterabfrage möglich

- Siehe Punkt „Die rote LED des Adapters leuchtet nicht“
- Siehe Punkt „Die rote LED leuchtet, aber es ist keine Abfrage möglich“
- Siehe Punkt „Zugriff auf das Webinterface nicht möglich“
- Rx- und/oder Tx-Belegung nicht korrekt, Pinbelegung und/oder Adapteranschluss stimmt nicht mit der Angabe in der Datei *BSB_lan_config.h* überein
- Falscher Bus-Typ (BSB/LPB)

14.5 Regler wird nicht korrekt erkannt

- Regler ist ausgeschaltet
- Regler wurde erst nach dem Arduino angeschaltet (automatische Reglererkennung funktioniert dann nicht)
- Regler ist nicht oder falsch mit dem Adapter verbunden
- Gerätefamilie und -variante (`http://<IP-Adresse>/6225/6226`) des Reglers unbekannt

14.6 HK1 kann nicht bedient werden

- Adapter ist evtl. als RGT2 konfiguriert

14.7 Es kann keine Raumtemperatur an einen HK1 gesendet werden

- Adapter ist evtl. als RGT2 konfiguriert

14.8 HK2 kann nicht bedient werden

- Adapter ist evtl. als RGT1 konfiguriert

14.9 Es kann keine Raumtemperatur an einen HK2 gesendet werden

- Adapter ist evtl. als RGT1 konfiguriert

14.10 Einstellungen des Reglers können nicht via Adapter verändert werden

- Zugriff des Adapters ist auf Lesen beschränkt („`FL_RONLY`“)

14.11 Der Adapter reagiert manchmal nicht auf Abfragen

- Der Arduino ist nicht multitaskingfähig - warte, bis eine Abfrage abgeschlossen ist (insbesondere umfangreichere Abfragen wie bspw. ganze Kategorien oder auch die Darstellung des Logfiles dauern u.U. recht lange)

14.12 Bei der Abfrage der Logdatei passiert ‚nichts‘

- Es ist keine microSD-Karte eingelegt
- Das Loggen auf microSD-Karte war oder ist deaktiviert
- Die Logdatei ist sehr groß, die (graphische) Darstellung dauert entsprechend länger

14.13 Es werden keine 24h-Durchschnittswerte angezeigt

- Das entsprechende Definement ist nicht aktiviert
- Es sind keine zu berechnenden Parameter angegeben

14.14 Bei der Abfrage der Daten von DS18B20-/DHT22-Sensoren passiert ‚nichts‘

- Es sind keine Sensoren angeschlossen
- Die entsprechenden Definements sind nicht aktiviert
- Die Pinbelegung ist nicht korrekt eingestellt
- Die Sensoren sind fehlerhaft installiert oder defekt

14.15 Die DS18B20-Sensoren zeigen falsche Werte an

- Die Stromversorgung und Installation prüfen (Größe des PullUp-Widerstands prüfen, Kondensatoren verbauen, Verkabelung prüfen, richtige Topologie verwenden etc.)

14.16 Der ‚Serielle Monitor‘ der Arduino IDE liefert keine Daten

- Der Adapter ist nicht zusätzlich via USB angeschlossen
- Falscher Anschluss (COM-Port) oder falsches Board in der Arduino IDE ausgewählt
- Falsche Baudrate eingestellt → auf 115200 Baud einstellen
- Adapter nicht am Regler angeschlossen, Regler ist ausgeschaltet → siehe o.g. Punkte

15. FAQ

15.1 Kann ich Adapter & Software mit einem Raspberry Pi nutzen?

Ja und nein.
Der Adapter kann mit einem Raspberry Pi 2 verwendet werden, wenn andere Pinheader genutzt werden (weibliche statt männliche) und die Platine entsprechend mit den Komponenten für die RPi-Nutzung bestückt ist (R11-13, Q11+12, SJ2+3).

Die BSB-LAN-Software kann NICHT mit einem RPi verwendet werden, sie ist ausschließlich auf dem hier vorgestellten Arduino-System lauffähig!
Zur Nutzung des Adapters mit einem RPi muss eine vollkommen andere Software genutzt werden. Weitere Informationen diesbezüglich sind am Ende von Kapitel 1. Der BSB-LPB-LAN-Adapter und die BSB-LAN-Software zu finden.

15.2 Kann ich einen Adapter gleichzeitig an zwei Regler anschließen?

Nein, das geht leider nicht.

Derzeit benötigt man für jeden Regler einen Adapter bzw. ein komplettes Hardware-Setup (Arduino, Ethernet-Shield, Adapter), um die jeweiligen reglerspezifischen Parameter via BSB abrufen zu können.
Sollten jedoch mehrere Regler vorhanden und bereits miteinander via LPB verbunden sein, beachte bitte die folgende FAQ.

15.3 Kann ich einen Adapter via LPB anschließen und mehrere Regler abfragen?

Ja, wenn die vorhandenen Regler bereits korrekt via LPB miteinander verbunden und entsprechend konfiguriert sind (korrekte LPB-Adressvergabe).
Die Möglichkeit, Abfragen fallweise an unterschiedliche Regler zu senden, ist mittlerweile gegeben, jedoch ist diese Funktion noch nicht ausgiebig getestet. Siehe hierzu den entsprechenden Punkt in Kapitel 8.1 Auflistung und Beschreibung der URL-Befehle.

15.4 Ist ein multifunktionaler Eingang des Reglers direkt via Adapter schaltbar?

Nein!

Die multifunktionalen Eingänge der Regler (bspw. H1, H2, H3 etc.) sind nicht direkt an den Adapter anzuschließen!

Soll bspw. eine Betriebsartumschaltung oder Erzeugersperre mittels H1 als Arbeitskontakt realisiert werden, so muss der jeweilige Eingang den Herstellerangaben entsprechend parametriert und belegt werden. Eine Steuerung dieser Art muss mittels eines anzuschließenden Relais erfolgen, dessen reglerseitiger Ausgang unbedingt potentialfrei sein muss, d.h. es darf keinerlei Fremdspannung anliegen! Das Relais hat in dem Fall lediglich die Aufgabe, den Kontakt zu schließen (oder zu öffnen).

Das Relais wiederum kann jedoch unter bestimmten Umständen vom Arduino gesteuert werden (bspw. mittels eines Relaisboards).

Entsprechende Relais findest du im Internet, bei Unsicherheiten solltest du deinen Elektriker und/oder Heizungsinstallateur zu Rate ziehen. Eine falsche Belegung und/oder Parametrierung kann den Regler u.U. zerstören!

15.5 Ist zusätzlich ein Relaisboard am Arduino anschließ- und steuerbar?

Ja. Siehe diesbezüglich den entsprechenden Punkt in Kapitel 8.1 Auflistung und Beschreibung der URL-Befehle.

15.6 Kann ich bspw. den Zustand eines angeschlossenen Koppelrelais abfragen?

Ja. Siehe diesbezüglich den entsprechenden Punkt in Kapitel 8.1 Auflistung und Beschreibung der URL-Befehle.

15.7 Kann ich behilflich sein, um bisher nicht unterstützte Parameter hinzuzufügen?

Ja! Wenn dein Heizungssystem über Parameter verfügt, die von der Software bisher nicht unterstützt werden, würden wir uns sehr freuen, wenn du uns unterstützt!

15.8 Warum erscheinen bei einer Komplettabfrage einige Parameter doppelt?

Wenn du eine Komplettabfrage aller Parameter via URL-Befehl machst (`http://<IP-Adresse>/0-10000`) kann es sein, dass sich einige Parameter bzw. Programmnummern in der Auflistung wiederholen. Dies kommt daher, dass es es zwar unterschiedliche Parameter sind, diese aber die gleiche Command ID haben. Dies stellt nur einen ‚optischen Mangel' dar, der die Funktionalität nicht negativ beeinflusst.

15.9 Warum werden manchmal bestimmte Parameter nicht angezeigt?

Wenn der Regler nach erfolgtem Adapteranschluss angeschaltet wird und der Arduino zu diesem Zeitpunkt bereits lief, funktioniert die automatische Reglererkennung nicht. Der Arduino muss dann lediglich resettet bzw. aus- und wieder angeschaltet werden.

Sollten dann bestimmte Parameter noch immer nicht erscheinen, so sollte bitte einmal `/Q` ausgeführt und die Webausgabe zusammen mit der Ausgabe von
`http://<IP-Adresse>/6220-6228`
gemeldet werden.

15.10 Warum ist kein Zugriff auf angeschlossene Sensoren möglich?

Wenn du DHT22- und/oder DS18B20-Sensoren korrekt am Arduino/Adapter angeschlossen hast, die entsprechenden Menüs im Webinterface jedoch nicht anwählbar sind, hast du vermutlich die betreffenden Einträge in der Datei *BSB_lan_config.h* nicht entsprechend angepasst. Siehe Kap. 5, 11 & 13.

15.11 Ich nutze ein W5500-LAN-Shield, was muss ich tun?

Prinzipiell ist die Nutzung eines Shields mit dem Chip-Typ W5100 zu empfehlen, da die entsprechenden Bibliotheken bereits eingebunden sind.
Sollte dennoch ein W5500 zum Einsatz kommen, ist
1. die Datei *BSB_lan_config.h* entsprechend anzupassen (s. Kap. 5) und
2. die Datei *Ethernet2.zip* im Unterverzeichnis *src* entpacken.

15.12 Können Stati oder Werte als Push-Mitteilungen abgesetzt werden?

Nein, nicht ohne weitere Software wie z.B. FHEM. Dafür müsste ansonsten die Therme ständig abgefragt werden, was den Bus (und die Erreichbarkeit des Arduino) stark belasten würde. Die sinnvollere Variante wäre, bestimmte Werte z.B. alle 60 Sekunden abzurufen und dann anhand bestimmter Kriterien weitere Aktionen auszulösen.
Bei FHEM wäre das mit `DOIF` oder `NOTIFY` möglich.

15.13 Kann bspw. FHEM auf bestimmte Broadcasts ‚lauschen'?

FHEM kann zwar lauschen, aber BSB-LAN kann bisher keine eigenständigen Nachrichten absetzen. Dazu müsste ein Hintergrundprozess die auflaufenden Broadcast-Meldungen anhand konfigurierbarer Schwellenwerte auswerten und über einen HTTP-Client-Aufruf an eine definierbare Zieladresse absetzen. Ob dies mit dem begrenzten Speicherplatz des Arduino noch umsetzbar ist, wäre fraglich. Wer sich aber daran probieren möchte, ist herzlich eingeladen, dies zu tun!

15.14 Warum kommt es manchmal zu timeout-Problemen bei FHEM?

Das könnte an der Dauer des Sende-/Empfangsvorgangs liegen. Man sollte den timeout-Wert in FHEM so bemessen, dass für jeden Parameter pro Setzbefehl zwei und pro Abfragebefehl eine Sekunde angesetzt werden.

Sind mehrere einzelne (BSB-LAN-spezifische) HTTPMOD-Abfragen definiert und werden diese zum gleichen Zeitpunkt ausgeführt, kann es außerdem vorkommen, dass sie sich somit gegenseitig ‚behindern', da der Bus bereits von einer Abfrage belegt ist. Als Abhilfe können hier entweder unterschiedliche Abfrageintervalle gewählt oder alle Abfragen in eine HTTPMOD-Abfrage gelegt werden, die dann intern nacheinander abgearbeitet wird.

15.15 Gibt es ein Modul für FHEM?

Jein. Ein Modul wird gerade vom FHEM-Forumsmitglied „justme1968" entwickelt:
https://forum.fhem.de/index.php/topic,84381.0.html
Die Entwicklung ist jedoch noch nicht abgeschlossen, so dass ein zuverlässiger und problemloser Einsatz bisher noch nicht garantiert werden kann.

15.16 Warum werden unter /B bei Stufe 2 keine Werte angezeigt?

Wenn du einen Gasbrenner hast, so wird dieser höchstwahrscheinlich modulieren und generell nicht über ein zweistufiges Brennersystem verfügen; zweistufige Brenner kommen meist nur bei Ölbrennern zum Einsatz. Die Unterscheidung der Brennerstufen wird mittels spezifischer Broadcasts vorgenommen, die jedoch nicht jeder Regler sendet. In dem Fall werden die Brennerstarts und -laufzeiten kumuliert unter Stufe 1 dargestellt. Bitte beachte diesbezüglich auch den Hinweis unter „`/B`" in Kapitel 8.1 Auflistung und Beschreibung der URL-Befehle.

15.17 Ich habe den Eindruck, die angezeigten Werte bei /B sind nicht korrekt.

Das kann durchaus sein. Die jeweiligen Starts und Laufzeiten werden anhand von Broadcasts ermittelt, die automatisch vom Regler gesendet werden. Manchmal kann es vorkommen, dass einzelne BCs nicht ankommen, bspw. wenn zeitgleich eine Abfrage gestartet wird oder der Arduino das Logfile lädt.

15.18 Was ist der genaue Unterschied zwischen /M1 und /V1?

Mit dem URL-Befehl `/M1` aktivierst du den Monitor-Modus, mit `/V1` den Verbositäts-Modus.

Mit aktivierter Monitor-Funktion (`/M1`) werden alle Daten, die über den Bus gehen und nicht von BSB-LAN aus initiiert wurden, „roh“ auf dem seriellen Monitor ausgegeben.
Dies kann sinnvoll sein, um Fehlfunktionen in der Datenübertragung ausfindig zu machen, da ansonsten nur Meldungen von BSB-LAN verarbeitet werden, die von ihrem Aufbau her korrekt sind. Das schließt auch die Verarbeitung von Broadcast-Nachrichten ein, d.h. mit aktivierter Monitor-Funktion findet keine Auswertung dieser Nachrichten statt.

Die Monitor-Funktion erlaubt es z.B. bei Fehlermeldungen genauer zu sehen, ob eine Nachricht schlichtweg nicht auf dem Bus angekommen ist oder ob BSB-LAN sie wegen fehlerhafter Übertragung verworfen hat. Die volle Kontrolle hätte man mit einem zweiten BSB-LAN-Adapter, der auf dem Bus lauscht und dann alle ein- und ausgehenden Nachrichten protokolliert.

Mit (seit v0.41 per default) aktiviertem Verbositäts-Modus (`/V1`) werden zu jedem von BSB-LAN initiierten Aufruf und der entsprechenden Antwort neben dem Klartext auch die entsprechenden Rohdaten auf dem seriellen Monitor ausgegeben, wenn die Nachricht von ihrem Aufbau her korrekt sind und fehlerfrei übertragen wurden.
Eine Auswertung von (fehlerfreien) Broadcasts findet hier weiterhin statt. Es werden hier beim Senden aber nur die Daten ausgegeben, die BSB-LAN vorbereitet hat. Dies muss nicht bedeuten, dass diese Daten - z.B. bei Hardwarefehlern - auch auf dem Bus ankommen. Umgekehrt werden beim Auswerten der Rückmeldung auf einen Befehl zwar die Daten ausgegeben, die auf dem Bus zurück gekommen sind, aber nur dann, wenn die Nachricht auch korrekt aufgebaut war.

Eine Kombination aus beiden Parametern ist möglich und führt dazu, dass im Monitor-Modus auch bei von BSB-LAN initiierten Nachrichten die Rohdaten ausgegeben werden - mit den bereits erwähnten Einschränkungen des Verbositäts-Modus bezüglich des Verwerfens von nicht korrekt aufgebauten Nachrichten.

15.19 Kann ich eigenen Code in BSB-LAN einbinden?

Ja, dafür gibt es die Datei *BSB_lan_custom.h*. Hier können eigene Programmteile geschrieben werden, die bei jedem Schleifendurchlauf (loop) aufgerufen werden. Damit kann man z.B. unabhängig von externen Home-Automations-Systemen Sensoren auswerten und/oder Relais schalten.
Die Beispieldatei wertet z.B. zwei DHT22-Feuchtigkeits-/Temperatursensoren aus und schaltet beim Unter- bzw. Überschreiten ein Relais, das an einem digitalen Ausgang angeschlossen ist.

15.20 Kann ich MAX!-Thermostate einbinden?

Ja, das ist möglich. Dazu musst du das entsprechende Definement in der Datei *BSB_lan_config.h* aktivieren und anpassen. Mittels entsprechender Modifikationen in der Datei *BSB_lan_custom.h* können weitere Funktionen realisiert werden, mit der derzeitigen Programmierung ist eine eigenständige Raum-Ist-Wert-Übermittlung (ohne FHEM) möglich. Siehe auch die jeweiligen Punkte in Kapitel 5. Einstellungsrelevante Parameter der BSB-LAN-Software und 8.1 Auflistung und Beschreibung der URL-Befehle.

15.21 Warum ist der Adapter nach einem Stromausfall nicht mehr erreichbar?

Dieses Verhalten wurde des Öfteren bei den günstigen LAN-Shield-Clones beobachtet, mit einem originalen Arduino-LAN-Shield scheint dieses Problem nicht aufzutreten. Nach Drücken des Reset-Knopfes am Arduino ist der Adapter wieder wie gewohnt erreichbar. Abhilfe könnte eine kleine USV für den Arduino schaffen, so dass der Arduino nicht stromlos wird. Andere Lösungen sind bisher nicht bekannt.

15.22 Warum ist der Adapter (ohne Stromausfall) manchmal nicht mehr erreichbar?

Dieses Problem ist bisher nur vereinzelt aufgetreten, eine eindeutige Lösung oder Erklärung für dieses Verhalten gibt es bisher noch nicht. Bei dem Nutzer, der davon berichtete, kam ebenfalls ein günstiger LAN-Shield-Clone zum Einsatz. Laut Mitschnitt des Seriellen Monitors lief der BSB-LAN-Sketch ohne Probleme weiter, lediglich das LAN-Shield war nicht mehr erreichbar. Abhilfe schaffte nur ein Reset. Sollte dieses Verhalten auftreten, ist das Testen eines weiteren LAN-Shields zu empfehlen, da Hardwareprobleme des betroffenen LAN-Shields nicht auszuschließen sind. Der Einsatz eines originalen Arduino-LAN-Shields ist selbstverständlich eine weitere Option.

15.23 Warum kommen beim Senden manchmal ‚query failed'-Meldungen?

Wenn Befehle, die in der Regel problemlos gesendet werden können, plötzlich ‚query failed'-Fehlermeldungen auslösen, könnte dies in der eingesetzten Hardware begründet sein. Es scheint, als wenn einige günstige Arduino-Mega-Clones zeitweise unzuverlässig arbeiten und diffuse Probleme verursachen. Abhilfe könnte ein Austausch des Arduino schaffen, der Einsatz eines originalen Arduino ist selbstverständlich eine weitere Option.

Ein Nutzer berichtete von erfolgreichen Änderungen an der Adapter-Hardware selbst, die er zur Eingrenzung des Problems vornahm.

Dieses Problem wird aktuell verfolgt und es wird aktiv nach einer Lösung gesucht. Sollte sich der Austausch von Hardwarekomponenten des Adapters bei solchen ‚Problem-Clones' als dauerhaft erfolgreich zeigen, so wird dies kommuniziert und im Platinenlayout berücksichtigt werden.

15.24 Ich finde keinen LPB- oder BSB-Anschluss, nur L-BUS und R-BUS?!

In diesem Fall schließe bitte den Adapter NICHT an und beachte das Kapitel 3.3 Hinweis: Neue Modellgeneration - NICHT unterstützter Regler von Brötje.

15.24 Ich habe weitere Fragen, an wen kann ich mich wenden?

Das Beste wäre, wenn du dich dafür im FHEM-Forum (https://forum.fhem.de/) anmelden würdest, da dort speziell für diesen Adapter ein eigener Thread existiert und sich dort eine nette und hilfsbereite Community findet. Hier findet ein reger Austausch über die Hard- und Software statt, Fragen werden meist zügig beantwortet und auf Updates wird hingewiesen.

Hier findest du den entsprechenden Thread:
https://forum.fhem.de/index.php/topic,29762.0.html

Wenn du dich mit deinen Fragen vorstellst, gib uns bitte zuerst genaue Informationen bzgl. des von dir verwendeten Heizungstyps, des Reglers, des verwendeten Bus-Typs etc.

Wenn du den Adapter bereits erfolgreich angeschlossen und in Verwendung hast, frage bitte außerdem die Parameter 6220-6228 (`http://<IP-Adresse>/6220-6228`) ab und schreibe die Ausgaben zusätzlich mit in deine Beschreibung.

Prinzipiell kann man sagen: Lieber erst einmal zu viele Informationen, als zu wenige.

Fragen, deren Antworten sich aus dem gründlichen Lesen dieses Handbuchs ergeben, werden ab einem gewissen Punkt lediglich mit einem Verweis hierauf beantwortet. Bitte bedenke, dass dies für jeden von uns nur ein Hobby-Projekt ist und wir keinerlei Geld damit verdienen.

16. Offene Punkte

- Mehr Befehle (Command IDs) hinzufügen.
  Nur die bekannten Befehle aus der genannten FHEM-Forendiskussion und den getesteten Heizungssystemen sind Bestandteil des Programms. Jeder Nutzer eines anderen Heizungssystems kann fehlende Parameter dekodieren und zur Weiterentwicklung der Software beitragen (siehe 15.7 Kann ich behilflich sein, um bisher nicht unterstützte Parameter hinzuzufügen?).
  Unser Ziel ist es, ein generell lauffähiges System zu entwickeln, das herstellerübergreifend mit allen Heizungssystemen verwendet werden kann, die einen BSB/LPB aufweisen.
  Jede Hilfe und jede Rückmeldung ist willkommen!

  Aufruf:
  Aktuell sind speziell bei dem Regler des Typs RVS43.325 etliche Parameter noch nicht dekodiert und werden somit von BSB-LAN auch noch nicht unterstützt. Da es anscheinend der aktuellste RVS-Reglertyp ist, gibt es dort etliche neue Parameter und erweiterte Funktionsoptionen.
  Jeder Benutzer eines solchen Reglertyps (bisher bekannte Heizungsmodelle: Brötje BOB und z.T. WOB) kann hier also aktiv dazu beitragen, BSB-LAN weiter zu vervollständigen!
- Testen und Vervollständigen der Funktionalität.
  Mit der gegenwärtigen Implementierung können bereits viele Werte gesetzt werden. Jedoch sind noch immer Tests nötig und einige Parameter müssen hinzugefügt werden.
- Dekodieren der DE-Telegramme.
  Möglicherweise beinhalten sie Statusinformationen, die ohne Abfragen genutzt werden können.
- Unterstützung für heizungsseitige Fehlermeldungen hinzufügen.

17. Weiterführende Informationen und Quellen

Ein reger Austausch bzgl. der hier vorgestellten Hard- und Software findet in folgendem Forum statt: https://forum.fhem.de/index.php/topic,29762.0.html
Dies ist auch eine gute Anlaufstelle für Fragen, Erfahrungsaustausch und Support, wo auch regelmäßig über Neuerungen und erfolgte Updates informiert wird.

Sämtliche Dokumentationen zur hier vorgestellten Hard- und Software sowie die verschiedenen Software-Versionen sind unter https://github.com/fredlcore/bsb_lan zu finden.
Dieses Handbuch ist zudem unter https://github.com/1coderookie/BSB-LPB-LAN erhältlich.

Die Software und die dazugehörigen Dokumentationen für den Einsatz des hier vorgestellten Adapters in Verbindung mit einem Raspberry Pi 2 ist unter https://github.com/loehnertj/bsbgateway zu finden.

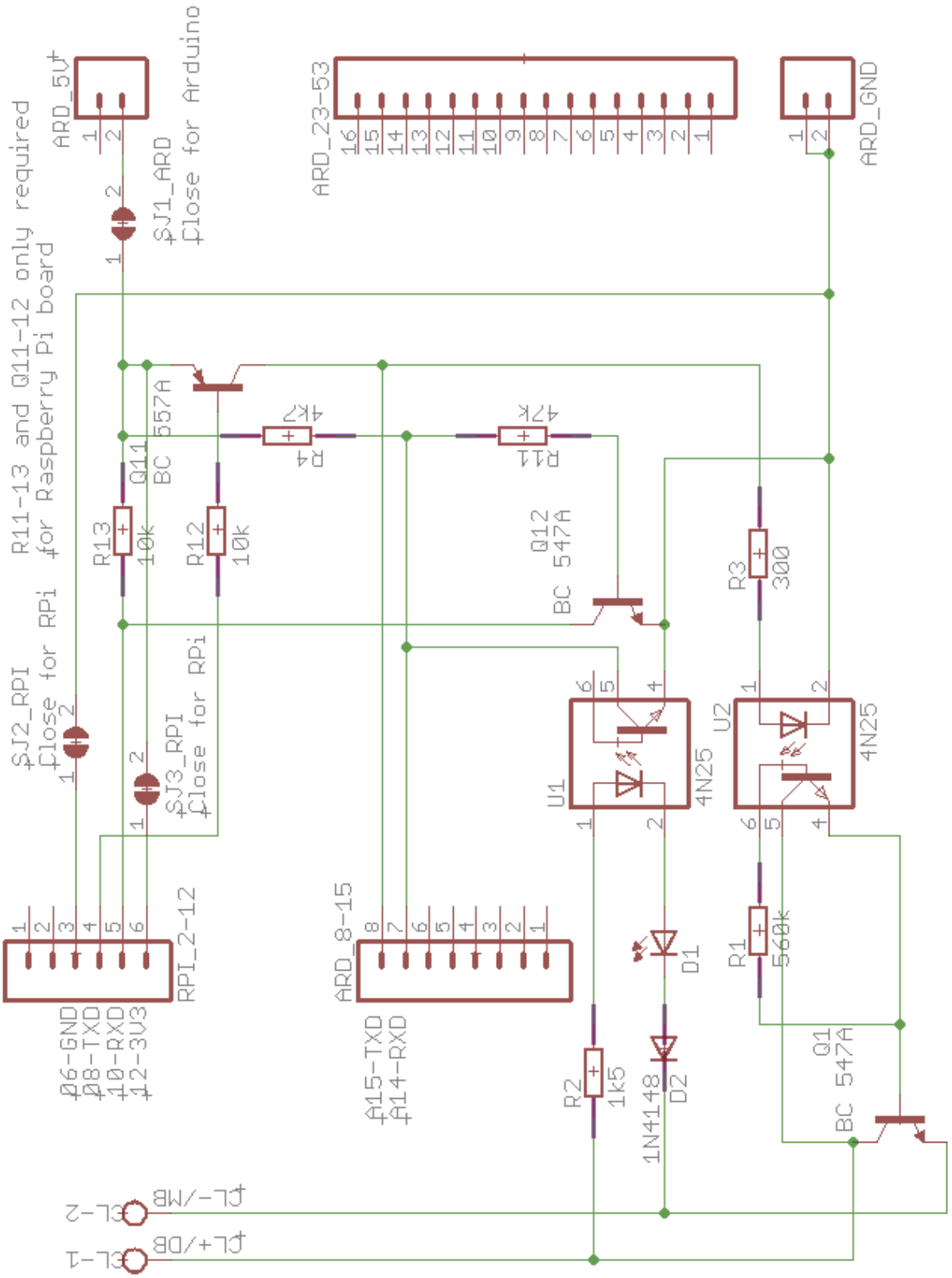
Die initiale Idee der Regleranbindung via BSB/LPB kann hier nachvollzogen werden:
http://www.mikrocontroller.net/topic/218643
http://blog.dest-unreach.be/2012/12/14/reverse-engineering-the-elco-heating-protocol

Als relativ umfangreiche Quelle mit vielen Parameterbeschreibungen sei das „Brötje Systemhandbuch ISR Plus“ empfohlen. Es stellt neben zahlreichen anderen und modellspezifischen Anleitungen die zugrunde liegende ‚Referenz‘ für die Parameterdefinitionen des hier vorgestellten Projekts dar.

Tiefergehende Informationen wie Spezifikationen und technische Anforderungen der Bus-Typen sind den jeweiligen Dokumenten der Hersteller zu entnehmen.
Speziell hinsichtlich des LPB seien zwei Dokumente von „Siemens Building Technologies - Landis & Staefa Division“ empfohlen:
- CE1N2030D Local Process Bus LPB Systemgrundlagen
- CE1N2032D Local Process Bus LPB Projektierungsgrundlagen

Hinsichtlich der Installation und Verwendung von DHT22- und OneWire-Sensoren wie dem DS18B20 gibt es zahlreiche Informationsquellen. Im Internet finden sich etliche kostenlose Anleitungen, Beispielinstallationen und Skripte.
Bei Problemen mit einer umfangreicheren OneWire-Installation sei ein Blick auf die technischen Anforderungen von OneWire (speziell hinsichtlich der Bus-Typologie und der Leitungslängen) und auf die Hinweise im entsprechenden Kapitel empfohlen.

Anhang A1: Schaltplan BSB-LPB-LAN-Adapter v2



Anhang A2: Anmerkungen zum Schaltplan

A2.1 Kurze Legende zum Schaltplan

D1 = Leuchtdiode
D2 = Diode
Q(x) = Transistor
R(x) = Widerstand
U(x) = Optokoppler
ARD = Arduino
RPI = Raspberry Pi
CL+/- = BSB-Anschluss
DB/MB = LPB-Anschluss
TXD = Digitalpin Senden
RXD = Digitalpin Empfangen

A2.2 Teileliste

Für den Einsatz des Adapters an einem *Arduino*:
1x LED (rot) (Betriebsspannung max. 2,8V, Sperrspannung 5V) (→ D1)
1x Diode 1N4148 (→ D2)
1x Transistor BC547A (→ Q1)
je 1x Widerstand 560kΩ (→ R1), 1,5kΩ (→ R2), 300Ω (→ R3), 4,7kΩ (→ R4)
2x Optokoppler 4N25 (→ U1, U2) (plus optional 2x Sockel)

Für den Einsatz des Adapters an einem *Raspberry* zusätzlich:
1x Widerstand 4,7kΩ (→ R11)
2x Widerstand 10kΩ (→ R12, R13)
1x Transistor BC557A (→ Q11)
1x Transistor BC547A (→ Q12)

Optional:
Anschlussklemmen, Pinheader etc. für die Leitungen.

A2.3 Generelle Hinweise

Vor dem Löten gilt: Bitte den Schaltplan aufmerksam studieren!

Bei Verwendung der vorgefertigten Platine ist für den Einsatz an einem ***Arduino*** zwingend die Verbindung SJ1 herzustellen!
Die Bestückung von R11-13 sowie Q11&12 ist nicht nötig!

Für den Einsatz an einem ***Raspberry*** sind jedoch SJ2 und SJ3 zu setzen!
Ebenso sind in diesem Verwendungsfall die Komponenten R11-13 und Q11&12 zwingend erforderlich!

Die folgenden Hinweise ersetzen kein grundsätzliches Elektronik-Vorwissen, könnten aber vielleicht doch dem einen oder anderen Elektronik-Anfänger eine kleine Hilfe sein.

Generell ist es u.U. hilfreich, die Bauteile erst einmal zu positionieren und erst am Schluss zu verlöten, so dass ein Vertauschen der Komponenten verhindert wird.
Ein vorheriger Breadboard-Testaufbau ist natürlich eine Option, aber aufgrund nicht auszuschließender Problemquellen (Nutzung einer falschen Steckreihe, eventuelle Wackelkontakte o.ä.) nicht unbedingt empfehlenswert.

Bitte achte darauf, dass die Bauteile beim Löten nicht zu heiß werden, da sie u.U. Schaden nehmen können. Für den Einsatz der Optokoppler-ICs U1 und U2 bietet es sich daher an, einen entsprechenden IC-Sockel zu verwenden und die Optokoppler erst nach Fertigstellung der Lötarbeiten in die Sockel zu stecken. Achte dabei auf die korrekte Ausrichtung der Sockel/Optokoppler; ebenso ist auf die korrekte Ausrichtung von bspw. Dioden und Transistoren zu achten!

Vor der Inbetriebnahme des Adapters ist es ratsam, die komplette Bestückung nochmals gründlich zu überprüfen und (falls möglich) durchzumessen. Kalte Lötstellen, versehentlich überbrückte Kontakte, nicht-gesetzte Lötverbindungen („SJ“ bei der fertigen Platine) etc. können ein unerklärliches und schwer zu diagnostizierendes Fehlverhalten des Adpaters bis hin zu einem eventuellen Reglerdefekt nach sich ziehen!

Viel Erfolg!

Anhang B: Cheatsheet URL-Befehle

URL-Befehl	Auswirkung
/<x>	Wert/Einstellung von Parameter <x> anzeigen
/<x>,<y>,<z>	Werte/Einstellungen der Parameter <x>, <y> und <z> anzeigen
/<x>-<y>	Werte/Einstellungen der Parameter <x> bis <y> anzeigen
/A	Anzeigen der 24h-Durchschnittswerte
/A=<x>,<y>	Ändern der 24h-Durchschnittswertberechnung in Parameter <x>, <y>
/B	Anzeige akkumulierter Brennerlaufzeiten (in Sek.) & -takte (plus TWW)
/B0	Zurücksetzen des Zählers Brennerlaufzeiten & -takte
/C	Anzeige der Konfiguration von BSB-LAN
/D	Anzeige der Logdatei der microSD-Karte
/DG	Grafische Anzeige der Logdatei der microSD-Karte
/D0	Zurücksetzen der Logdatei & neue Generierung des Headers
/E<x>	ENUM-Werte für Parameter <x> anzeigen
/G<x>	GPIO: Abfragen des Pins <x>
/G<x>,<y>	GPIO: Setzen des Pins <x> auf high (<y> = 1) oder low (<y> = 0)
/G<x>,I	GPIO: Abfragen des Pins <x> mit gleichzeitigem Setzen auf INPUT
/H	Abfrage optional angeschlossener DHT22-Sensoren
/I<x>=<y>	INF-Nachricht an Parameter <x> mit Wert <y> senden
/K	Alle Regler-Kategorien auflisten
/K<x>	Alle Parameter und Werte von Regler-Kategorie <x> abfragen
/L=0,0	Loggen auf microSD-Karte deaktivieren
/L=<x>,<y>	Log-Intervall auf <x> Sekunden setzen, mit (optional) Log-Parameter <y>
/LB=<x>	Loggen von Bus-Telegrammen: Nur Broadcasts (<x>=1) oder alle (<x>=0)
/LU=<x>	Loggen von Bus-Telegrammen: Nur unbekannte (<x>=1) oder alle (<x>=0)
/M<x>	Monitor-Modus aktivieren (<x> = 1) oder deaktivieren (<x> = 0)
/N	Reset und Neustart des Arduino (Dauer ca. 15Sek)
/O	Übersicht der URL-Befehle
/P<x>	Busprotokoll / Bustyp setzen: <x> = 0 → BSB | 1 → LPB | 2 → PPS
/P<x>,<s>,<d>	Busprotokoll/-typ <x>, eigene Adresse <s>, Zieladresse <d> setzen
/Q	Test auf nicht-freigegebene reglerspezifische Parameter
/R<x>	Abfrage des Reset-Werts für Parameter <x>
/S<x>=<y>	Wert <y> für Parameter <x> setzen
/T	Abfrage optional angeschlossener DS18B20-Sensoren
/V<x>	Verbositäts-Modus aktivieren (<x> = 1) oder deaktivieren (<x> = 0)
/X	Abfrage optional eingebundener MAX!-Thermostate

Anhang C: Changelog BSB-LAN-Software

Version 0.41 – 18.02.2018
- Improved graph legend when plotting several parameters
- Added JSON export; query with /J=a,b,c,d... or push queries to /JQ or push set commands to /JS
- Logging of MAX! parameters now possible with logging parameter 20007
- Added unit to log file as well as average output
- Rewrote device matching in cmd_tbl to accomodate also device variant (Gerätevariante). Run /Q with activated "#definde DEBUG" to see if transition has worked for your device!
- Bugfix ENUM memory adressing

Version 0.40 – 21.01.2018
- Implemented polling of MAX! heating thermostats, display with URL command /X.
See BSB_lan_custom.h for an example to transmit average room temperature to heating system.
- Added new category "22 - Energiezähler" - please note that all subsequent categories move one up!
- New virtual parameter 1601 (manual TWW push)
- Added Fujitsu Waterstage WSYP100DG6 device family (211)
- Added CTC device family (103)
- New definement "#define TRUSTED_IP2" to grant access to a second local IP address
- Added optional definement "#define GatewayIP" in BSB_lan_config.h to enable setting router address different from x.x.x.1
- Removed parameter 10109 because it is the same as 10000
- Added function to check all known CommandIDs on your own heating system. Use /Q after enabling definement "#define DEBUG" in BSB_lan_config.h
- Added parameter numbers to category menu
- Updated analyze.sh
- hopefully fixing the memory issue
- Moved HTML strings to html_strings.h

Version 0.39 – 02.01.2018
- Implementation of PPS-Bus protocol. See /K40 for the limited commands available for this bus. Use setBusType(2) to set to PPS upon boot or /P2 to switch temporarily.
- Set GPIOs to input by using /Gxx,I
- Definement "#define CUSTOM_COMMANDS" added. Use this in your configuration to include individual code from "BSB_lan_custom.h" (needs to be created by you!) which is executed at the end of each main loop. Variables "custom_timer" and "custom_timer_compare" have been added to execute code at arbitrary intervals.
- Added LogoBloc Unit L-UB 25C device family (95)
- several new parameters added
- Bugfix for logging Brennerlaufzeit Stufe 2

Version 0.38 – 22.11.2017
ATTENTION: New BSB_lan_config.h configurations! You need to adjust your configuration when upgrading to this version!
- Webserver port is now defined in #define Port xx
- IP address is now defined in #define IPAddr 88,88,88,88 form - note the commas instead of dots!
- Special log parameters 20002 to 20006 have changed, see BSB_lan_config.h for their new meaning
- Added new virtual parameter 701 (Präsenztaste) which enters reduced temperature mode until next timed switch
- Added Brötje BOB device family (138), including many new parameters!
- Added Brötje SOB26 device family (28)
- Added Elco Aquatop 8es device family (85)
- Added Elco Thision 13 Plus device family (203)

- Added Weishaupt WTU 25-G familiy (50)
- Added output for absolute humidity (g/m3) for DHT22 sensors
- New schematics for Arduino/Raspberry board layout
- Included support for W5500 Ethernet2 shields. Activate definement ETHERNET_W5500 in BSB_lan_config.h
- Including two-stage oil furnaces BC-counters and logging - please note that logging parameters have been adjusted, see BSB_lan_config.h for new values!
- Added new options for commands /P and /S to allow specifying a different destination device during runtime
- Added new configuration definement CUSTOM_COMMANDS which includes BSB_lan_custom.h at the end of each main loop. You may use custom_timer (set to current millis()) and custom_timer_compare to execute only every x milliseconds.
- Bugfixing SD-card logging in monitor mode
- Bugfix for setting hour:time parameters via webinterface

Version 0.37 – 08.09.2017
- LPB implementation! More than 450 parameters supported! Switch temporarily between LPB and BSB with the Px command (0=BSB, 1=LPB) or use the setBusType config option to set bus-type at boot-time. Parameter numbers are the same as for BSB.

Version 0.36 – 23.08.2017
- bugfix: brought back VT_BIT list of options which were erroneously deleted :(, fixed/freed several memory issues

Version 0.35 – 25.06.2017
- new category "Sitherm Pro"; caution: category numbers all move up by one, starting from category "Wärmepumpe" (from 20 to 21) onwards.
- graph display of logging data now comes with crosshair and shows detailed values as tooltip
- improved SD-card output by factor 3 (from 16 to 45 kbps), switching SD-card library from from SD.h to SdFat.h (https://github.com/greiman/SdFat) brings another 10% performance boost
- adjusted paths and directory layout of SdFat to enable compiling from sketch directory.
- new data type vt_sint for signed int data, currently only used in some Sitherm Pro parameters

Version 0.34 – 29.05.2017
- Log data can now be displayed as graph
- Webinterface can now display and set vt_bit type parameters in human-readable form
- added KonfigRGx descriptions; caution: various sources used, no guarantee that descriptions match your individual heating system!
- vt_bit is generally read-only in the webinterface. To set, use URL command /S with decimal representation of value
- fixed a bug with vt_seconds_short5, affecting parameters 9500 and 9540.
- fixed bug regarding Fujitsu's device family (from 127 to 170)
- moved libraries from folder libraries to src so they can be included without copying them to the Arduino libraries folder
- modified DallasTemperature.h's include path for OneWire.h

Version 0.33 – 09.05.2017
- no more heating system definements anymore due to new autodetect function based on device family (parameter 6225), or set device_id variable to parameter value directly
- two more security options: TRUSTED_IP to limit access to one IP address only, and HTTP authentication with username and password
- Average values are saved on SD-card if present and LOGGER definement is activated
- deactivate logging by setting /L0=0 - this way you can enable LOGGER definement without filling up SD card but still save average values
- new error codes for THISION

- added dump of data payload on website for commands of unknown type, greyed out unsopported parameters
- enable logging of telegrams (log parameter 30000) also in monitor mode (bsb.cpp and bsb.h updated)
- time from heating system is now retreived periodically from broadcast telegrams, further reducing bus activity
- new data type vt_bit for parameters that set individual bits. Display as binary digits, setting still using decimal representation
- new data type vt_temp_short5_us for unsigned one byte temperatures divided by 2 (so far only 887 Vorlaufsoll NormAussentemp)
- new data type vt_percent5 for unsigned one byte temperatures divided by 2 (so far only 885 Pumpe-PWM Minimum)
- new data type vt_seconds_word5 for two byte seconds divided by 2 (so far only 2232, 9500 and 9540)
- new data type vt_seconds_short4 for (signed?) one byte seconds divided by 4 (so far only 2235)
- new data type vt_seconds_short5 for (signed?) one byte seconds divided by 5 (so far only 9500, 9540)
- new data type vt_speed2 for two byte rpm (so far only 7050)
- cleaned up set() function from apparent duplicate cases
- added cases for vt_temp_word, vt_seconds_word5, vt_temp_short, vt_temp_short5, vt_seconds_short4 to set() function

Version 0.32 – 18.04.2017
- lots of new parameters suppoerted
- newly designed webinterface allows control over heating system without any additional software or cryptic URL commands. URL commands of course are still available, so no need to change anything when using FHEM etc.
- German webinterface available with definement LANG_DE
- new URL-command /LB=x to log only broadcast messages (x=1) or all bus messages (x=0)
- new URL-command /X to reset the Arduino (need to enable RESET definement in BSB_lan_config.h)
- new logging parameters 20002 and 20003 for hot water loading times and cycles
- moved DS18B20 logging parameters from 20010-20019 to 20200-20299 and DHT22 logging parameters from 20020-20029 to 20100 to 20199
- moved average logging parameter from 20002 to 20004
- set numerous parameters to read-only because that's what they obviously are (K33-36)
- various bugfixes

Version 0.31 – 10.04.2017
- increased dumping of logfile by factor 5 / as long as we still have memory left, you can increase logbuflen from 100 to 1000 to increase transfer speed from approx. 16 to 18 kB/s
- adjusted burner activity monitoring based on broadcast messages for Brötje systems
- removed definement PROGNR_5895 because so far, it has only disabled an ENUM definition.
- removed definement PROGNR_6030 because double command ID could be resolved via BROETJE / non-BROETJE definements
- renamed BROETJE_SOB to BROETJE in order to allow for fine-grained distinction between different BROETJE cases (e.g. 6800ff). This means you have to activate TWO definements when using a Brötje system now: The general BROETJE as well as BROETJE_SOB or BROETJE_BSW. Have a look at your serial log for parameters 6800 to see which command IDs fit your system and activate one of both accordingly.
- changed 16-Bit addressing of flash memory to 32-Bit to address crashes due to ever growing PROGMEM tables - now we have lots of air to breathe again for new command IDs :)
- removed trailing \0 string from several ENUMs that led to wrong ENUM listings. Please keep in mind not to end ENUMs with a trailing \0 !

Version 0.30 – 22.03.2017
- Time library by Paul Stoffregen (https://github.com/PaulStoffregen/Time) is now required and included in the library folder.
- adds logging of raw telegram data to SD card with logging parameter 30000. Logging telegram data is affected by commands /V and /LU
- adds command /LU=x to log only known (x=0) or unknown (x=1) command IDs when logging telegram data
- removed define USE_BROADCAST, broadcast data is now always processed
- new internal functions GetDateTime, TranslateAddr, TranslateType

Version 0.29 – 07.03.2017
- adds command /C to display current configuration
- adds command /L to configure logging interval and parameters
- adds option for command /A to set 24h average parameters during runtime
- adds special parameter 20002 for logging /A command (24h averages, only makes sense for long logging intervals)
- bugfixes for logging DS18B20 sensors

Version 0.28 – 05.03.2017
- adds special parameters 20000++ for SD card logging of /B, /T and /H commands (see BSB_lan_config.h for examples)
- adds version info to BSB_LAN web interface

Version 0.27 – 01.03.2017
- adds date field to log file (requires exact time to be sent by heating system)
- /D0 recreates datalog.txt file with table header
- added "flags" field to command table structure. Currently, only FL_RONLY is supported to make a parameter read-only
- added DEFAULT_FLAG in config. Defaults to NULL, i.e. all fields are read/writeable. Setting it to FL_RONLY makes all parameters read-only, e.g. for added level of security. Individual parameters can be set to NULL/FL_RONLY to make only these parameters writable/read-only.

Version 0.26 – 27.02.2017
- added functionality for logging on micro SD card, using the slot of the w5100 Ethernet shield
- more parameters added (e.g. 8009)

Version 0.25 – 21.02.2017
- more FUJITSU parameters added

Version 0.24 – 14.02.2017
- updated README with added functions
- added German translations of FAQ and README, courtesy of Ulf Dieckmann

Version 0.23 – 12.02.2017
- minor bugfix

Version 0.22 – 07.02.2017
- more FUJITSU parameters
- (hopefully) correct implementation of VT_VOLTAGE readings
- minor bugfixes

Version 0.21 – 06.02.2017
- added numerous parameters for Fujitsu Wärmepumpe, including new #define FUJITSU directive to activate these parameters due to different parameter numbers
- minor bugfixes

Version 0.20 – 27.01.2017
- added more parameters for Feststoffkessel
- minor bugfixes

Version 0.19 – 01.01.2017
- added humidity command "H", currently for DHT22 sensors
- added 24h average command "A", define parameters in BSB_lan_config.h
- removed trailing whitespace from menu strings
- fixed command id 0x053D04A2 for THISION heaters
- included Rob Tillaart's DHT library because there are various libraries implementing the protocol and this one is used in the code for its ability to address multiple sensors with one object.
- removed /temp URL parameter as it is a duplicate of /T
- included loop to display DHT22 sensors in IPWE
- making compiling IPWE extensions optional (#define IPWE)

Version 0.18 – 22.12.2016
- split off configuration into bsb_lan_config.h
- split off command definitions into bsb_lan_defs.h
- changed GPIO return values from LOW/HIGH to 1/0
- reactivated and updated IPWE (define parameters in config)
- check for protected pins when accessing GPIO (define in config)
- added schematics and PCB files to new subfolder "schematics"

Version 0.17a – 20.12.2016
- minor errors corrected

Version 0.17 – 20.12.2016
- merged v0.16 with FHEM user miwi's changes

Version 0.16 – 20.11.2016
- removed IPWE and EthRly interface
- added GPIO interface
- merged parameters from J.Weber
- resolved duplicate command IDs

Version 0.15a – 25.07.2016
- collated the commands from a Python project and this project, merged the two versions, corrected obvious errors. Inserted hypothetical numerical values in ENUM definitions where Broetje manuals documented only the message texts.
- added information from traces in a Broetje installation with an ISR-SSR controller and a WOB 25C oil furnace.

Version 0.15 – 21.04.2016
- added Solar and Pufferspeicher from Elco Logon B & Logon B MM

Version 0.14 – 04.04.2016
- minor bugfixes for Broetje SOB
- extended broadcast handling (experimental)

Version 0.13 – 31.03.2016
- change resistor value in receiving path from 4k7 to 1k5
- added values 0x0f and 0x10 to Enum8005
- fixed strings for Zeitprogramme
- added timeout for sending a message (1 second)

- added option T for querying one wire temperature sensors in mixed querys
- added special handling for Broetje SOB
- simplified settings

Version 0.12 – 09.04.2015
- added ONEWIRE_SENSORS to ipwe
- fixed parameter decoding for ELCO Thision heating system

Version 0.11 – 07.04.2015
- fixed parameter decoding for ELCO Thision heating system

Version 0.10 – 15.03.2015
- added more parameters for ELCO Thision heating system

Version 0.9 – 09.03.2015
- added more parameters for ELCO Thision heating system
- printTelegramm returns value string for further processing

Version 0.8 – 05.03.2015
- added parameters for ELCO Thision heating system
- added IPWE extension
- minor bugfixes

Version 0.7 – 06.02.2015
- added bus monitor functionality

Version 0.6 – 02.02.2015
- renamed SoftwareSerial to BSBSoftwareSerial
- changed folder structure to enable simple build with arduino sdk

Version 0.5 – 02.02.2015
- bugfixes
- added documentation (README)
- added passkey feature
- added R feature (query reset value)
- added E feature (list enum values)
- added setter for almost all value types
- fixed indentation
- added V feature to set verbosity for serial output
- set baudrate to 115200 for serial output
- redirecting favicon request
- added some images of the BSB adapter

Version 0.1 – 21.01.2015 – initial version