BSB-LAN-Adapter

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

Autoren:
Gero Schumacher (gero.schumacher ät gmail.com) (bis v0.16)
Frederik Holst (bsb ät code-it.de) (ab v0.17 aufwärts)
Basierend auf dem Code und der Arbeit von vielen anderen Entwicklern. Vielen Dank!

Übersetzung EN-DE: Ulf Dieckmann

Host System:
Die Software wurde entwickelt, um auf einem Arduino-Mega2560-Board samt Ethernet-Shield zu laufen. Da es unterschiedliche Pinbelegungen bei den verschiedenen Ethernet-Shields gibt, ist es u.U. nötig, den BSB-LPB-Adapter an andere Pins anzuschließen und die entsprechenden Änderungen bzgl. der Pinbelegung in der Software anzupassen.

Die Software wurde mit folgenden Komponenten getestet:

- SainSmart MEGA2560 R3 Development Board
- SainSmart Ethernet Schild für Arduino UNO MEGA Duemilanove Neu Version W5100
- BSB-Interface (siehe BSB_adapter.pdf)
  Für diese Konfiguration wird Pin A14 (68) als RX, und Pin A15 (69) als TX genutzt (s. Parameter weiter unten).

Der Adapter kann auch an einem Raspberry Pi 2 genutzt werden, jedoch muss in dem Fall eine gänzlich andere Software (bsb_gateway) genutzt werden, die hier gefunden werden kann. Für jeglichen Support in Zusammenhang mit der Software kontaktiere bitte direkt den Autor von bsb_gateway. Alle Informationen auf dieser Website betreffen nur die Arduino-Version! Hier kann kein Support für bsb_gateway gegeben werden.

Zielsystem:
Getestet mit verschiedenen Elco- und Brötje-Heizungssystemen (s. README). Die Kommunikation sollte prinzipiell mit allen Systemen möglich sein, die einen BSB aufweisen.

Erste Schritte:

- Verbinde die Anschlüsse CL+ und CL- des Adapters mit den entsprechenden Anschlüssen des Heizungssystems (mögliche Bezeichnungen am Heizungsregler sind BSB, FB (Fernbedienung/remote control), CL+/CL-). Für den LPB sind DB und MB zu nutzen, wobei DB(+) mit CL+ und MB(-) mit CL- zu verbinden sind.
- Downloade und installiere die aktuelle Version der Arduino IDE von https://www.arduino.cc/en/Main/Software (Windows-, Mac- und Linux-Version verfügbar).
- ~~Kopiere die Inhalte des BSB_lan-libraries-Ordners in deinen lokalen Arduino-libraries-Ordner (Eigene Dateien\Arduino\libraries\ unter Windows, ~/Documents/Arduino/libraries auf einem Mac).~~
  UPDATE: Ab v0.34 gilt: Wichtigste Änderunge für alle Neuinstallationen ist, dass die Libraries, die nicht standardmäßig bei der Arduino IDE mit dabei sind, nun einfach im Sketch-Verzeichnis bleiben können und von dort eingelesen werden. Ein Kopieren ist nun nicht mehr nötig, was zum einen gerade für Anfänger die Installation erleichtert und alle anderen bei der Aktualisierung von Libraries diese ebenfalls nicht mehr manuell verschieben müsst. Es *kann* sein, dass das Kompilieren fehl schlägt, wenn die gleiche Library bereits im Standard-Libraries-Verzeichnis der Arduino IDE liegt. In dem Fall müsste die gleichlautende Bibliothek dort (Win: MyDocuments\Arduino\Libraries bzw. Mac: ~/Dokumente/Arduino/Libraries) gelöscht werden.
- 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 Adressen-Kollision kommt).
- 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.
- Lade den Sketch auf den Arduino unter Sketch/Upload bzw. Sketch/Hochladen.
- Öffne die Seite
  `http://<IP-Adresse>/`
  (oder
  `http://<IP-Adresse>/<passkey>/`

wenn die Passkey-Funktion genutzt wird, s.u.) um zu sehen, ob alles korrekt kompiliert und hochgeladen wurde. Ein einfaches Webinterface sollte erscheinen.

Optional können die folgenden Parameter in der Datei **"BSB_lan_config.h"** angepasst werden:

- Konfiguration des Heizungssystems
  `int fixed_device_id = 0`
  Wenn der Wert auf 0 gesetzt ist, ist die automatische Erkennung des angeschlossenen Reglers beim Starten des Arduinos aktiviert. Alternativ kann hier der Wert von Parameter 6225 eingetragen werden.
  Ein fest eingestellter Wert (laut Ausgabe von Parameter 6225) stellt sicher, dass BSB_LAN auch dann noch korrekt arbeitet, wenn die Heizung erst nach dem Starten des Arduinos eingeschaltet wird (da in dem Fall die automatische Erkennung des angeschlossenen Reglers nicht funktionieren kann, da ja keine Rückmeldung vom Regler kommt).
- Die MAC-Adresse des Ethernet-Shields. Üblicherweise (jedoch nicht immer) befindet sie sich auf einem Aufkleber auf dem Ethernet-Shield, eine Änderung ist i.d.R. aber nur nötig, wenn mehr als ein Adapter verwendet wird:
  `byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEA };`
- IP-Adresse:
  `IPAddress ip(192,168,178,88);`
- Ethernet-Port:
  `EthernetServer server(80);`
- 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).
  Wenn bereits ein Raumgerät (RGT1) vorhanden ist, kann bzw. sollte der Adapter als RGT2 angemeldet werden:
  `BSB bus(68,69,7);`
- Bus-Protokoll
  `uint8_t bus_type = bus.setBusType(0);`
  Voreingestellt ist 0 für BSB, für LPB ist 1 einzustellen; mittels der URL-Befehle /P0 und /P1 kann entsprechend umgestellt werden.
- Man kann die Funktion eines Sicherheitsschlüssels (PASSKEY) aktivieren (s. unten):
  `#define PASSKEY "1234"`
- Man kann den Zugriff auf den Adapter auf Lesen beschränken, ein Setzen bzw. Verändern von Parametern der Heizungssteuerung per Adapter ist dann nicht mehr möglich. Dazu muss in der betreffenden Zeile (`#define DEFAULT_FLAG 0`) das Flag auf FL_RONLY gesetzt werden:

`#define DEFAULT_FLAG FL_RONLY;`
UPDATE:
In der Datei *BSB_lan_config.h* ist nun als Voreinstellung DEFAULT_FLAG auf read-only gesetzt, d.h., dass alle Werte (erst einmal) nur lesbar sind!
Wer das ändern will, muss wieder DEFAULT_FLAG auf 0 setzen oder bei den einzelnen Parametern (z.B. 10000 oder 710) in der Datei *BSB_lan_defs.h* den Wert DEFAULT_FLAG durch 0 ersetzen.

- Man kann die Sprache des Webinterfaces des Adapters auf Deutsch einstellen, indem man das entsprechende Definement aktiviert:
  `#define LANG_DE;`

Web-Interface des Adapters:
Eine einfache Webseite wird angezeigt, wenn ohne weitere Parameter auf die URL des Servers zugegriffen wird.

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!).
Falls die PASSKEY-Funktion aktiviert ist (s. unten), muss die URL den definierten Schlüssel als erstes Element enthalten, bspw. `http://<ip-of-server>/<passkey>/` - um die Hilfeseite zu sehen. Bitte nicht den Slash hinter dem Passkey vergesssen!
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 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.
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".

Zusätzlich zum Web-Interface kann auf alle Funktionen direkt via URL-Befehlen zugegriffen werden. Dies ist nützlich, wenn der Adapter in Verbindung mit Heimautomationssystemen wie bspw. FHEM genutzt wird.

Alle Heizungsparameter werden anhand ihrer Zeilennummern abgefragt. Eine nahezu vollständige Übersicht findet sich im „Brötje Systemhandbuch ISR“.

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

Die Parameter sind in Kategorien unterteilt, die den im Display dargestellten Untermenükategorien entsprechen, wenn auf das Heizungssystem vom integrierten Regler aus zugegriffen wird.

Alle Kategorien auflisten:

```
http://<ip-of-server>/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-of-server>/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-of-server>/K<x>
```

Alle Werte von Zeile x abfragen:

```
http://<ip-of-server>/<x>
```

Alle Werte eines Zeilenbereichs abfragen (von Zeile x bis Zeile y):

```
http://<ip-of-server>/<x>-<y>
```

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

```
http://<ip-of-server>/K11/8000/8003/8005/8300/8301/8730-8732/8820
```

Frage den Reset-Wert für Parameter x ab:

```
http://<ip-of-server>/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

```
http://<ip-of-server>/S<x>=<v>
```

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.

Um einen Parameter auf 'abgeschaltet/deaktiviert' zu setzen, muss einfach ein leerer Wert eingefügt werden.

```
http://<ip-of-server>/S<x>=
```

Die Beschreibung der Werteformatierungen wird hier hinzugefügt. Bis dahin sieh dir den Quellcode an (Funktion 'set').

Sende eine INF Nachricht für den Parameter x mit dem Wert v

```
http://<ip-of-server>/I<x>=<v>
```

Setze das Bus-Protokoll auf BSB (x=0) oder LPB (x=1)

```
http://<ip-of-server>/P<x>
```

Einige Werte können nicht direkt gesetzt werden. Das Heizungssystem wird mit einer TYPE_INF-Nachricht informiert, bspw. die Raumtemperatur:

```
http://<ip-of-server>/I10000=19.5
```
// Raumtemperatur beträgt 19.5°C

Setze den Verbositäts-Level auf n

```
http://<ip-of-server>/V<n>
```

Der voreingestellte Verbositäts-Level ist 0. Wenn er auf 1 gesetzt wird, werden der Bus überwacht und alle Daten zusätzlich im Raw-Hex-Format dargestellt.

Der Verbositäts-Level betrifft sowohl die serielle Konsole des Arduino Mega2560 als auch (optional) das Loggen der Bus-Daten auf die microSD-Karte, so dass die Karte u.U. sehr schnell voll wird! Die html-Ausgabe bleibt unverändert.

Bus-Monitor aktivieren:

```
http://<ip-of-server>/M<n>
```

Wenn er 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 Milisekunden dargestellt.

Die Ausgabe der Überwachung betrifft nur die serielle Konsole des Arduino Mega2560. Die html-Ausgabe bleibt unverändert.

Setzen/Abfragen der GPIO Pins

```
http://<ip-of-server>/G<xx>[=<y>]
```

Gibt den momentanen Status von GPIO Pin xx zurück (0 oder 1). Kann ebenfalls benutzt werden, um den Pin auf 0 (LOW) oder 1 (HIGH) zu setzen.

Reservierte Pins, die nicht gesetzt werden dürfen, können in der BSB_lan_config.h unter dem Parameter GPIO_exclude gesperrt werden.

24h Durchschnittswerte von ausgewählten Parametern anzeigen

```
http://<ip-of-server>/A[=parameter1,...,parameter20]
```

Zeigt rollierende 24h Durchschnittswerte ausgewählter Parameter an. Initiale Festlegung dieser Parameter in 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-of-server>/T
```

Gibt die Temperaturwerte von optional angeschlossenen DS18B20-Sensoren aus.

Abfrage von DHT22 Feuchtigkeits-/Temperatursensoren

```
http://<ip-of-server>/H
```

Gibt die Feuchtigkeits-/Temperaturwerte von optional angeschlossenen DHT22-Sensoren aus.

Akkumulierte Brennerlaufzeit

```
http://<ip-of-server>/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 von den Broadcast-Nachrichten ermittelt wurden.
`/B0` setzt den Zähler zurück.

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-of-server>/L=0,0
```

Zum Aktivieren können dann einfach wieder das Intervall und die gewünschten Parameter eingetragen werden (s. Konfiguration des Logfiles).

Konfiguration des Logfiles

```
http://<ip-of-server>/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.

Konfiguration des Loggens von Bus-Telegrammen:

```
http://<ip-of-server>/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-of-server>/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-of-server>/D
```

Zeigt den Inhalt der Datei datalog.txt, die sich auf der microSD-Karte im Slot des Ethernet-Shields befindet.

Mittels `/D0` kann die Datei datalog.txt gelöscht und mitsamt korrekter CSV-Header-Datei neu erstellt werden (dieser Schritt wird für die erste Benutzung empfohlen, bevor das Loggen startet).
UPDATE:
Wer Parameter auf SD-Karte loggt, bekommt nun neben der reinen Textform auch die

Möglichkeit, einen Graphen angezeigt zu bekommen (siehe Screenshot im README-File). Dafür muss bei Javascript-Blockern die Domain d3js.org freigegeben werden. Das hat den Grund, dass der Arduino weiterhin nur die CSV-Datei in den Browser schiebt und diese dann mit dem D3 Framework grafisch aufbereitet wird. Der Befehl lautet `/DG`.

Bus-typ (BSB oder LPB) vorübergehend ändern:

```
http://<ip-of-server>/P<x>
```

Wechselt zwischen BSB (x=0) und LPB (x=1). Um den Bus-typ dauerhaft festzulegen, sollte die Option setBusType config in der Datei *BSB_lan_config.h* entsprechend angepasst werden.

Resetten/Restarten des Arduinos

```
http://<ip-of-server>/N
```

Reset/Restart des Arduinos nach einem Pausieren für 8 Sekunden (#define RESET in *BSB_lan_config.h*). Dabei wird gleichzeitig auch noch das EEPROM des Arduino mit Nullen überschrieben. Dies hat z.Z. 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 einem Druck auf die Anlerntaste neu gegenüber BSB-LAN 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.
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.

Offene Punkte

- Mehr Befehle (command ID) hinzufügen. Nur die bekannten Befehle aus der o.g. Forendiskussion und dem getesteten Heizungssystem (ELCO) sind Bestandteil des Programms. Jeder Nutzer eines anderen Heizungssystems kann den Verbositäts-Level auf 1 setzen und die fehlenden Befehle dekodieren, indem er sie einfach über die integrierte Steuerung aufruft. Unser Ziel ist es, eine generell lauffähiges System zu entwickeln, das herstellerübergreifend mit allen Heizungssystemen verwendet werden kann, die einen BSB aufweisen. Jede Hilfe und jede Rückmeldung ist willkommen!
- 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 hinzgefügt werden.
- Dekodieren der DE-Telegramme. Möglicherweise beinhalten sie Statusinformationen, die ohne Abfragen genutzt werden können.
- Unterstützung der vom Heizungssystem aus gesendeten Fehlerberichte hinzufügen.

Weiterführende Infos:
http://www.mikrocontroller.net/topic/218643
http://blog.dest-unreach.be/2012/12/14/reverse-engineering-the-elco-heating-protocol
http://forum.fhem.de/index.php/topic,29762.0.html
Brötje „Systemhandbuch ISR“