Inhalt

Gibt es einen einfachen Weg, um Parameter zu loggen?

Ja, den gibt es - zum Einen als standalone-Variante, zum Anderen als remote-Variante.

Um den Adapter als standalone-Logger zu nutzen, gehe wie folgt vor:

Stecke eine FAT32-formatierte microSD-Karte in den Speicherkartenplatz des Ethernet-Shields, bevor du den Arduino einschaltest. Einige Geräte erkennen u.U. keine Speicherkarten, die größer als 2GB sind, nutze in dem Fall eine kleinere Karte und formatiere sie mit FAT16.
Ändere dann die Datei BSB_lan_config.h und ent-kommentiere das Definement #define LOGGER. Füge dann die zu loggenden Parameter zur Variable log_parameters hinzu und bestimme das Logintervall mit der Variable log_interval. Später können während der Laufzeit sowohl das Intervall als auch die Logging-Parameter mittels des Befehls "/L=[Intervall],[Parameter1],...,[Parameter20]" geändert werden.

Wenn du das Setup fertig hast, schalte das System ein und warte auf ankommende Daten. Sämtliche Daten werden auf der Karte in der Datei datalog.txt im CSV-Format gespeichert, und können somit leicht in Excel oder OpenOffice importiert werden.
Den Dateiinhalt kannst du mit dem URL-Befehl "/D" einsehen. Um die Datei zurückzusetzen, 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-Interval entsprechend anpassen, und bspw. 59 Sekunden anstatt 60 Sekunden einstellen.

Für das Loggen in der remote-Variante gehe nach folgenden Schritten vor:

Führe diesen Befehl periodisch aus (z.B. per cron job):

```
DATE=`date +%Y%m%d%H%M%S`; wget -qO- http://192.168.1.50/1234/8310/720/710 |
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. Ändere einfach sowohl die Parameternummern in der http-Abfrage, als auch die des egrep-Befehls (und selbstverständlich die IP-Adresse sowie ggf. den Sicherheitscode (hier: 1234)). Später kannst du das Logfile basierend auf den Parameternummern sortieren, nutze hierfür den Befehl 'sort':

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

Ich nutze FHEM und möchte die Daten meines Heizungssystems darin weiter verarbeiten. Wie kann ich dies tun?

Bitte beachte, dass FHEM eine komplexe Software ist, und dementsprechend dies nicht der Ort ist, an dem grundsätzliche Informationen oder Anleitungen zu FHEM bereitgestellt werden. Solltest du jedoch bereits andere Geräte in FHEM konfiguriert haben, so hilft dir das Folgende hoffentlich weiter:

Um auf die Webschnittstelle des Adapters zuzugreifen, kann das Modul HTTPMOD in FHEM genutzt werden. Die folgende Beispielkonfiguration kann einfach an deine eigenen Bedürfnisse und die entsprechenden Parameter angepasst werden. Selbstverständlich müssen sowohl die IP, als auch der 'passkey' (falls der optionale Sicherheitsschlüssel aktiviert wurde) entsprechend angepasst werden. Der u.g. Code fragt die Parameter 8700, 8743 und 8314 alle 300 Sekunden ab und weist diese dem Gerät "THISION" (der Name meines 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.

```
define THISION HTTPMOD http://192.168.1.50/1234/8700/8743/8314 300
attr THISION userattr reading0Name reading0Regex reading1Name reading1Regex
reading2Name reading2Regex readingOExpr 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 readingOExpr $val=~s/[\r\n]//g;;$val
attr THISION set0Name Istwert
attr THISION set0URL http://192.168.1.50/1234/I10000=$val
attr THISION timeout 5
attr THISION userReadings Spreizung
{ sprintf("%.1f",ReadingsVal("THISION","Vorlauftemperatur",0)-
ReadingsVal("THISION","Ruecklauftemperatur",0));; }
```

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

Ich habe ein Relaisboard an dem Arduino Mega angeschlossen, wie kann ich die einzelnen Relais ansteuern bzw. deren Zustand abfragen?

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 (Achtung: Ändere entsprechend deiner Konfiguration sowohl die IP als auch den optional aktivierbaren Sicherheitsschlüssel 'passkey'!):

```
define EthRelais HTTPMOD http://192.168.1.50/1234/G05/G06/G07 30
attr EthRelais userattr reading0Name reading0Regex reading1Name reading1Regex
reading2Name reading2Regex readingOExpr readingOMap 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://192.168.1.50/1234/G07=$val
attr EthRelais set1Name Fan
attr EthRelais set1URL http://192.168.1.50/1234/G06=$val
attr EthRelais set2Name Bell
attr EthRelais set2URL http://192.168.1.50/1234/G05=$val
attr EthRelais setParseResponse 1
attr EthRelais setRegex GPIO[0-9]+:[ \t](\d)
attr EthRelais timeout 5
```

Mein Heizungssystem verfügt über Parameter, die von der Software bisher nicht unterstützt werden. Kann ich behilflich sein, diese Parameter hinzuzufügen?

Yes, you can :)! Du brauchst nur deinen mit der Heizungssteuerung verbundenen Arduino mit einem Laptop/PC über USB zu verbinden und den folgenden Schritten zu folgen:

1. Starte die Arduino IDE und öffne den seriellen Monitor (Menü Werkzeuge -> Serieller Monitor bzw. drücke Strg+Umschalt+M).
2. Aktiviere das Loggen zur seriellen Konsole und die Verbositäts-Ausgabe mit dem URL-Parametern /V1 auf dem Arduino, bspw. http://192.168.1.50/1234/V1. Alternativ dazu kannst du Bus-Telegramme auf die microSD-Karte loggen: Logge dazu (als einzigen!) Parameter 30000 (s. Logging-Abschnitt oben), setze die Variable log_unknown_only auf 1 (URL-Befehl /LU=1) und beobachte die Logeinträge mit dem URL-Befehl /D.
3. Schalte bei dem Heizungssystem über die integrierte Steuerung zu dem Parameter, den du analysieren möchtest (mittels des Drehrades, der Pfeiltasten bzw. der spezifischen Eingabemöglichkeiten deiner Heizungssteuerung).
4. Warte auf 'Ruhe' auf dem Bus, dann schalte einen Parameter weiter vor und gleich wieder zurück zu dem Parameter, den du analysieren möchtest. Nun solltest du etwas im Log sehen, was in etwa so aussehen sollte:

```
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 sind von dem Parameter, zu dem hingeschaltet wurde. Die letzten vier Zeilen stammen von 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). Anstelle von DISP wird eventuell RGT1 angezeigt, dies ist abhängig vom jeweiligen Gerät, mit dem du die Eingaben am Heizungssystem tätigst (integrierte Steuerung oder angeschlossenes Raumgerät/Fernbedienung). Jedes Datentelegramm hat die folgende Struktur:

Byte 1: immer 0xDC (Beginn des Telegramms)
Byte 2: Quellgerät (0x00 = Heizungssystem, 0x06 = Raumgerät 1, 0x07 = Raumgerät 2, 0x0A = Display, 0x7F = Broadcast) plus 0x80
Byte 3: Zielgerät (gleiche Werte wie bei Quellgerät)
Byte 4: Telegrammlänge (Startbyte des Telegramms (0xDC) wird nicht mitgezählt)
Byte 5: Nachrichtentyp (0x02 = Info, 0x03 = Setzen, 0x04 = ack, 0x05 = nack, 0x06 = Abfragen, 0x07 = Antworten, 0x08 = Fehler)
Byte 6-9: Command ID (-> diese ist es, die wir brauchen!)
Byte 10...: Payload data (optional)
Letzten zwei Bytes: CRC-Checksumme

5. Das untere Datentelegramm im obigen Beispiel hat die Command ID 0x113D3063. Bitte beachte, dass die ersten beiden Bytes der Command ID beim Nachrichtentyp "Abfragen" (0x06) vertauscht sind. Stelle daher

bitte sicher, dass du auf das richtige Telegramm siehst (Typ "Antwort" (0x07), die letzte Zeile des o.g. Beispiels).

6. 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 darstelt). Fals es ihn bereits gibt, fahre fort mit Schritt 8.
7. 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 8, aber anstatt CMD_UNKNOWN zu ersetzen, müssen logischerweise die Command ID und der Wertetyp der kopierten Zeile ersetzt werden.

8. 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 meherere Optionen bereitstellen, musst du eine entsprechende Zeile im Abschnitt "ENUM tables" hinzufügen.
9. Wenn der Adapter den gleichen Wert ausgibt, wie er auf dem Display der integrierten Heizungssteuerung dargestellt wird, 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!
10. Wenn du fertig bist, überprüfe nochmals, ob die neue Command ID nicht bereits irgendwo anders in der Definitionsdatei verwendet wird (wenn du beispielsweise nach der Command ID suchst, sollte sie nur einmal gefunden werden). Da das BSB-Protokoll 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 nämlich möglich, dass 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. Wir werden dann bedingungsabhängige Compiler-Flags hinzufügen, so dass das Heizungssystem X anders kompiliert wird als Heizungssystem Y, so dass letztlich beide Systeme die mehrdeutigen Command IDs für die korrekten Parameter nutzen können.
11. Bitte sende nur die neuen bzw. aktualisierten Zeilen an bsb (ät) code-it.de - solltest du eine Diff-Datei nutzen, überprüfe bitte vor dem Erstellen nochmals, ob du wirklich die aktuelle BSB_lan_defs.h des Repositoriums heruntergeladen hast, da manchmal die Dateien aktualisiert werden, ohne dass direkt eine neue Version veröffentlicht wird.