BiDiB-Broker
Entwicklerhandbuch

Ergänzung zum BiDiB-Broker Anwenderhandbuch

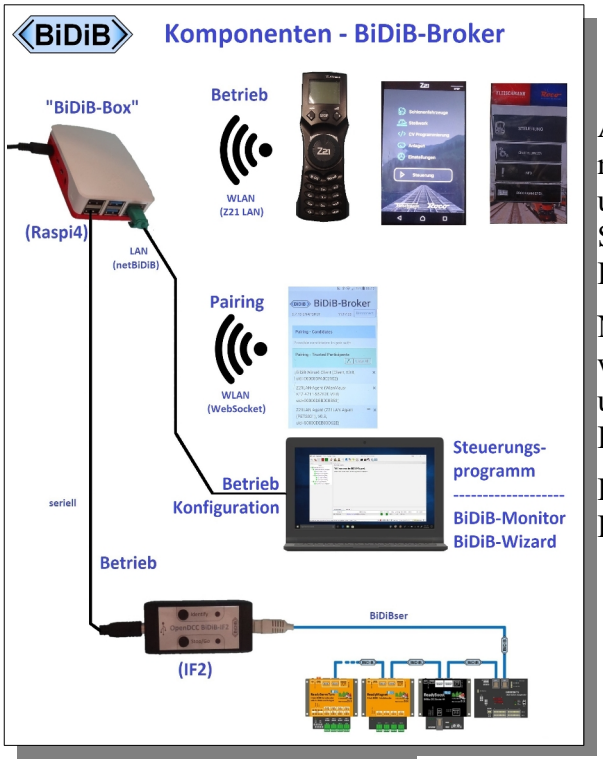
Inhaltsverzeichnis

Historie

Datum	Name	Version	Bemerkung
2020-04-25	Michael Schäfer	0.1	initial version (english)
2022-01-22	Michael Schäfer	0.2	parameter “overwrite” is dropped, now master data always written. Restrictions for Mac OS X 10.11 (Java 16).
2022-12-09	Michael Schäfer	0.3	Übersetzung ins Deutsche und Aktualisierung
2022-12-13	Michael Schäfer	0.4	Fehlerfall „*libpi4j.so nicht gefunden*“ aufgenommen
2022-12-14	Michael Schäfer	0.5	Review von Andreas Bergmaier eingearbeitet
2022-12-17	Michael Schäfer	0.6	Review von Alexander Reich u. Robert Kölz eingearbeitet, neues Kapitel „Verbindungsaufbau im TCP-Netzwerk“
2023-01-02	Michael Schäfer	0.7	Kap. 12.3. Unable to load [libpi4j.so]: angepasst
2023-01-14	Michael Schäfer	0.8	Kap. 4 BiDiBser neu, bevorzugte Ports ergänzt
2023-01-30	Michael Schäfer	0.9	Kap. 12.2 Hinweis auf Handbuch Embedded Systems
2023-02-05	Michael Schäfer	0.10	Kap. 9: bidib-serial.host-adapter neu, Fußzeile
2023-06-03	Michael Schäfer	0.11	Kap. 9.+12.2. Fehlerteufel beseitigt
2023-06-07	Michael Schäfer	0.12	Kap. 7+10 Änderung der Protokoll-Stufe
2023-06-09	Michael Schäfer	0.13	Kap.13 Abhilfe bei Fehler: Port in Use.
2023-06-28	Michael Schäfer	0.14	Kap. 7+10 spezielle Protokoll-Definitionsdateien
2023-97-26	Michael Schäfer	0.15	Kap. 12.1 Connect-Button obsolet

1 Einführung

BiDiB-Broker (Broker) ist eine auf Java 17 basierende Anwendung, um einen oder mehrere BiDiB-Sub-Knoten (Knoten) über TCP/IP sowie USB/seriellen BiDiB-Interface-Knoten (BiDiBser/Master) mit einem Host-System (Host) zu verbinden.

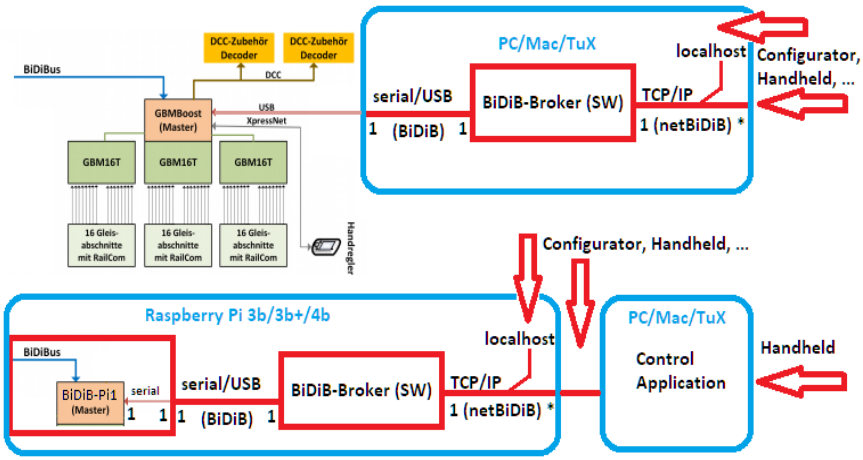


Auf der TCP/IP-Seite unterstützt Broker das netBiDiB-Protokoll. Die Knoten können unterschiedlicher Art sein, wie z.B. Steuerungsanwendungen, Handheld-Terminals, Konfigurationstools usw.

Nachdem eine Verbindung erfolgreich hergestellt wurde, leitet der Broker die Daten zwischen Host und Knoten weiter. Die Daten müssen dem BiDiB-Protokoll folgen.

Diese Betriebsweise wird im Folgenden als Hub-Rolle bezeichnet.

In der anderen, der Avatar-Rolle, stellt Broker für genau einen Master genau einen TCP/IP-Zugang bereit und tritt selber nicht nach Außen auf.



2 Betriebssystem

Broker kann auf verschiedenen Betriebssystemen wie Windows10 (WIN), Mac OS X1[1] (MAC) und einigen Linux Distributionen (TuX) laufen. Auf diesen Systemen kann man Broker mit Standardwerten, also ohne weitere Attribute, starten.

Auf einem BiDiB-Pi mit Pairing-LED und -Taster wird ein zusätzlicher Startparameter benötigt.

1 Java 17 erfordert Mac OS X 10.12 oder höher, Mac OS X 10.11 läuft nur bis Java 16.

3 Verbindungsaufbau im TCP-Netzwerk

Der Verbindungsaufbau in einem TCP-Netzwerk geschieht über eine Anfrage eines Clients an einen Server, um dessen Dienste in Anspruch nehmen zu können.

Standardmäßig ist Broker der Server, der dem Client (hier: Host) seine Dienste anbietet. Broker unterstützt die automatische Verbindung (via mDns). Dadurch kann die TCP-Adresse via Bonjour-Verfahren bekannt gegeben werden (siehe auch Anwenderhandbuch zum BiDiB-Broker). Das Port für den Host ist standardmäßig 62875 (Details siehe auch BiDiB-Protokoll, Transport > Netzwerk).

Knoten, die das Bonjour-Verfahren beherrschen, werden von Broker ebenfalls automatisch erkannt und in den Knotenbaum eingebunden. Zu dieser „Seite“ hin nimmt Broker als Client die Dienste des Knotens in Anspruch.

Sollte das Bonjour-Verfahren von einem Knoten nicht unterstützt werden, kann auch manuell eine Verbindung über das Port 62874 aufgebaut werden. Der Knoten verbindet sich dann über diesen so genannten Nebeneingang als Client mit Broker als Server.

Verbindungen mit netBiDiB-Knoten unterstützt Broker ausschließlich in der Rolle als Hub.

4 BiDiBser (Serielle Schnittstelle/USB)

Sowohl in der Avatar- als auch in der Hub-Rolle ermöglicht Broker die Verbindung zum BiDiBus über eine oder beim Hub mehrere serielle Schnittstellen.

Zur automatischen Erkennung ist ein Hot-Plug-Verfahren integriert mit dem die gefundenen BiDiBser-Gegenstellen automatisch eingebunden werden können.

Für den Fall, dass nicht alle gefundenen BiDiBser-Gegenstellen an Broker angebunden werden sollen, ist eine Liste „bevorzugter Ports“ als Startparameter zu übergeben. Es werden dann nur die aufgeführten Ports ausgewählt, sofern sie gefunden werden und lesbar sind.

5 Profile

Broker als SpringBoot-Anwendung erlaubt es, mit Profilen zu starten, die die Attribute des Standardprofils überschreiben. Mit solchen Profilen können die Benutzerattribute in einer Datei, im Ausführungspfad, definiert werden, anstatt sie in der Kommandozeile hinzuzufügen.

Das Standardprofil definiert die Attribute, die der Broker für WIN/MAC/TuX-Systeme benötigt. Es ist in einem Java-Archiv (jar-Datei) enthalten. Wichtige Attribute:

```
master-data:
  tcp-port-number: 62875
  pairing-wait-timeout-sec: 30
  pairing-wait-timeout-with-request: true
  pairing-button-long-pressed-time-sec: 3
  usb-delay-detect-sec: 0,1,2
  signature: "BiDiB-Broker-Hub"
```

Ein vom Anwender definiertes Profil, z.B. **application-klingon.yml** könnte wie folgt aussehen:

```
---
spring.config.activate.on-profile: "klingon"
master-data:
  tcp-port-number: 62800
  signature: "BiDiB-Win-Broker"
```

Im Beispiel werden Signatur und Port für die netBiDiB-Schnittstelle überschrieben, die restlichen Attribute werden dem Standardprofil entnommen.

Aufruf: **java -jar bidib-broker.jar --spring.profiles.active=klingon**

Die neuen Attribute werden in den Stammdaten (master-data) gespeichert.

6 Vorgefertigte Profile

Es gibt insgesamt vier vorgefertigte Profile für den Betrieb und eins für JUnit-Tests.

Für die Rollen gibt es kein eigenes Profil. Standard für WIN/MAC/TuX ist die Hub-Rolle und für den Raspi die Avatar-Rolle. Soll die Rolle geändert werden, muss sie explizit angegeben werden.

6.1 Gemeinsame Attribute

```
master-data:
  tcp-port-number: 62875
  pairing-wait-timeout-sec: 30
  pairing-button-long-pressed-time-sec: 3
base-data:
  file-base: ".bidib/broker-family"
  pairing-receipt: Dialog
```

6.2 WIN/MAC/TuX (Standardprofil)

Aufruf: **java -jar bidib-broker.jar**

```
spring.config.activate.on-profile: "bidib-hub"
master-data:
  signature: "BiDiB-Broker-Hub"
descriptor:
  prod-string-default: Broker-Hub
```

6.3 Raspi

Aufruf: **java -jar bidib-broker.jar --spring.profiles.active=bidib-pi-hat**

```
spring.config.activate.on-profile: "bidib-pi-hat"
bidib-system:
  default-role: Avatar
master-data:
  signature: "BiDiB-Broker-Pi-Hat"
serial-device:
  preferred-ports: "/dev/serial0"
base-data:
  pairing-receipt: Button
descriptor:
  prod-string-default: "Broker-Pi-Hat"
```

6.4 Test-Profile

Die beiden Test-Profile für WIN/MAC/TuX sowie Raspi unterscheiden sich nur durch die verwendeten Protokoll-Definitionsdateien von ihren Standardprofilen:

```
Aufruf:  java -jar bidib-broker.jar --spring.profiles.active=test
Aufruf:  java -jar bidib-broker.jar --spring.profiles.active=test-pi-hat
```

6.5 JUnit-Test

Das JUnit-Test-Profil wird nur beim Bauen der Applikation verwendet. Die benötigten Dateien landen in einem eigenen Ordner: **<home>/<temp>/broker-temp/**.

```
Aufruf:  java -jar bidib-broker.jar --spring.profiles.active=bidib-mac-x
```

```
spring:
  profiles: "bidib-mac-x"
master-data:
  signature: "BiDiB-Mac-Broker"
  serial-port-uri: "/dev/cu.usbserial-AI3AWIVY"
base-data:
  pairing-receipt: Dialog
```

7 Protokolle

Die Protokolle für das Standardprofile landen im Ordner **<home>/<temp>/.BiDiBBrokerFamily**. Die Dateinamen sind: **broker.log** sowie Archiv-Versionen.

Die Protokolldateien für die Standardprofile sind kumulierend, werden also auch beim Broker-Start fortgeschrieben. Bei einer Größe von 10 MB wird die Datei gesichert und eine neue angefangen. Das wird dreimal wiederholt, dann erst wird die erste Datei gelöscht.

Die Protokolle für die Testprofile landen im gleichen Ordner wie die der Standardprofile. Der Dateiname ist: **brokerTestProfile.log**.

Die Protokolldatei für das Testprofil wird bei jedem Broker-Start überschrieben!

Standardmäßig ist die Protokoll-Stufe derzeit auf DEBUG eingestellt. Zur Reduzierung der Ausgabe, z.B. nur auf INFO-Stufe, kann man Broker mit einem Parameter starten:

```
Aufruf:  java -jar bidib-broker.jar --logging.level.org.bidib.broker=INFO
```

Für eine minimale Protokollierung auf der Konsole steht ein eigene Protokolldefinition zur Verfügung:

```
Aufruf:
```

```
java -jar bidib-broker.jar --logging.config=classpath:logback-spring-min-con.xml
```

Auch im laufenden Betrieb ist eine Umstellung möglich. Siehe dazu Kapitel 10 Actuator.

8 Stammdaten

BiDiB-Monitor und -Wizard sowie BiDiB-Broker speichern ihre Stammdaten vereinbarungsgemäß im Verzeichnis: **<home>/.bidib** ab. Für Monitor und Wizard kann ein Anwender diese Vorgaben individuell ändern, für Broker ist keine Änderung des Stammdaten-Verzeichnisses vorgesehen.

Broker nutzt ein gemeinsames Verzeichnis mit anderen „Familienmitgliedern“: **<home>/.bidib/broker-family/data**. Die für Broker relevanten Dateien:

- **DescriptorBroker.yml** Enthält den „Descriptor“ für Hub oder Avatar
- **MasterData.yml** Enthält die Stammdaten aus dem aktuellen Profil
- **TrustedParticipants.yml** Enthält alle durch das „Pairing“ vertraute Gegenstellen
- **FeatureServiceBroker.yml** Enthält die von Broker bereitgestellten Features

9 Konfigurationsattribute

Alle Attribute können beim Start angegeben werden und überschreiben die Vorgabewerte.

Die Attribute in **DescriptorBroker.yml** werden beim Broker-Start automatisch aktualisiert, wenn es einen Rollenwechsel gegeben hat.

Attribut	Vorgabe	Werte	Erklärung
server.port	62876	49152...65535	Port für Pairing-GUI
connection.host.address	"localhost"	URL	URL für Discovery (n.c.)
connection.host.port	0	0, 49152...65535	Port für Discovery (0 → frei vom OS vergeben)
connection.side-entrance	62874	49152...65535	Port für den Nebeneingang
bidib-system.default-role	Hub	Avatar, Host, Hub	Rolle der Applikation
master-data.tcp-port-number	62875	49152...65535	Port für BiDiB-Host-System
master-data.pairing-wait-timeout-sec	30	0 … 255	Wartezeit für das „Pairing“ in Sekunden
master-data.pairing-wait-timeout-with-request	true	true / false	PAIRING_REQUEST mit oder ohne Wartezeit erwartet
master-data.pairing-button-long-pressed-time-sec	3	1 … 5	Drück-Zeit für das Löschen aller Vertrauten
master-data.signature	BiDiB-Broker-Hub	max. 24 Zeichen	Signatur beim Verbindungsaufbau
serial-device.host-adapter	rxtx	rxtx, scm, jsc	Adapter für serielle Schnittstellen
serial-device.usb-delay-detect-sec	0,1,2	Vektor	Versuche für USB-Ports mit Verzögerung in Sekunden

Attribut	Vorgabe	Werte	Erklärung
serial-device.preferred-ports	/dev/serial0	bei BiDiB-Pi (sonst leer)	Bevorzugte Ports/Symbolic Links, Komma separiert
descriptor.uid-host-base	80010DEA00	5 Byte	Basis-UID des Knotens
descriptor.p-version-default	0.9	2 Byte	BiDiB-Protokoll Version
descriptor.prod-string-default	BiDiB-Broker	max. 24 Zeichen	Produktname
descriptor.suffix	Broker		Suffix für Dateinamen
bidib.magic	AFFE	AFFE, B00D, FACE, C0DE	Systemkennung
discovery.type	_bidib._tcp.	_bidib._tcp.	Discovery Typ
discovery.bidib	node	node, interface	Discovery Teilnehmer
discovery.weight	0		Gewichtung
discovery.priority	0		Priorität
discovery.persistence	true		Speicherverhalten
base-data.file-base	.bidib/broker-family		Basisverzeichnis für die Stammdaten (wie BiDiB-Monitor und -Wizard)
base-data.pairing-receipt	Dialog	Button, Dialog, None	Verhalten beim „Pairing“ mit Dialog oder zusätzlich mit Button
netbidib-test-mode	false	false, directly	Test-Modus ohne „Pairing“ --netbidib-test-mode=directly
logging.config	classpath:logback-spring.xml	Verschiedene	Explizite Verwendung einer Protokolldefinitionsdatei

Beispiele für den Broker-Start von der Kommandozeile mit überschriebenen Attributen:

```
java -jar bidib-broker.jar --master-data.tcp-port-number=62800 --master-data.signature=BiDiB-Broker-New --master-data.pairing-wait-timeout-sec=10
```

10 Actuator

Broker basiert auf Spring-Boot-Technologie und unterstützt einige „Actuator Actions“, z.B.:

```
curl -X "GET" "http://127.0.0.1:62876/actuator/health"
```

```
curl -X "GET" "localhost/actuator/loggers/org.bidib.broker"
```

```
curl -X "POST" "http://127.0.0.1:62876/actuator/loggers/org.bidib.broker" -H
"Content-Type: application/json; charset=utf-8" --data
"{\"configuredLevel\": \"INFO\"}"
```

z.B. vermindere Protokollausgabe vom Wizard:

```
curl -X "POST" "http://127.0.0.1:62876/actuator/loggers/RAW" -H "Content-Type:
application/json; charset=utf-8" --data "{\"configuredLevel\": \"ERROR\"}"
```

11 Build

Eine Möglichkeit, Broker zu bauen ist von der Kommandozeile mit Maven 3 vom Pfad mit der Konfigurationsdatei *pom.xml:*

`mvn clean install`

Eine andere Möglichkeit besteht mit Eclipse aus dem Package Explorer heraus:

`Run as > Maven build … > Goals=clean install > Run.`

12 Pairing

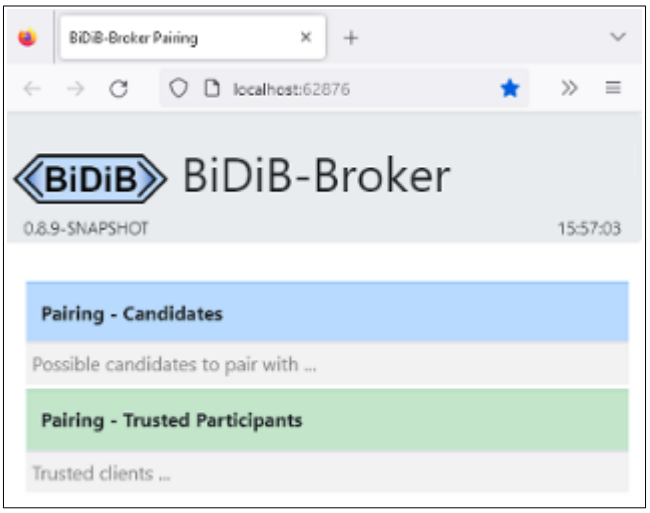
Das netBiDiB-Protokoll verlangt ein „Pairing“ zwischen den verschiedenen Teilnehmern[2].

Das Pairing-Verfahren ist Teil des Verbindungsaufbaus und arbeitet mit mehreren Teilnehmern parallel. Ansonsten ist es aber unabhängig vom Nachrichtenaustausch mit dem BiDiB-System.

12.1 Pairing mit Browser

Auf allen Systemen, besonders denen ohne Pairing-Taster und -LED, unterstützt Broker das „Pairing“ via Browser mit einer WebSocket-Schnittstelle.

Um sich mit Broker zu verbinden, reicht die Eingabe: http://localhost:62876 , der Rechnername (wenn ein DHCP-Server im Hintergrund läuft) oder die IP-Adresse jeweils mit Portnummer. Momentan nutzen wir das Port 62876.

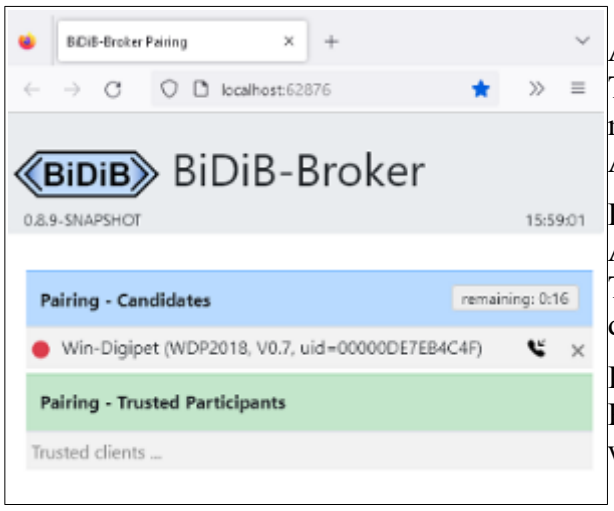


Die Verbindung zwischen Browser und Broker wird mit dem Start der Seite automatisch aufgebaut. Es erscheinen Broker-Version und die aktuelle Uhrzeit mit den möglichen Verbindungskandidaten und bereits vertrauten Partnern.

Bei bestehender Verbindung wird die Uhrzeit sekündlich aktualisiert.

Wie schon erwähnt muss das Vertrauen nur einmal für ein Zielsystem und jeden netBiDiB-Teilnehmer ausgesprochen werden und gilt solange, bis es explizit entzogen wird.

Sind Browser und Broker getrennt, was z.B. nach einem Neustart Brokers der Fall ist, wird die Uhrzeit nicht mehr weiter gezählt! Dann ist die Verbindung durch den Nachladen-Button des Browsers wieder herzustellen.



Auf dem folgenden Bild ist ein netBiDiB-Teilnehmer verbunden. Er erwartet das „Pairing“ mit Broker, das durch eine roten und blinkenden Alarm gemeldet wird.

Der Telefonhörer signalisiert eine ankommende Anfrage. Durch seine Betätigung wird dem Teilnehmer vertraut, mit dem eXit-Symbols wird die Annahme dagegen abgelehnt.

In der Rubrik „Pairing – Candidates“ werden alle Pairing-Teilnehmer angezeigt und können abgelehnt werden.

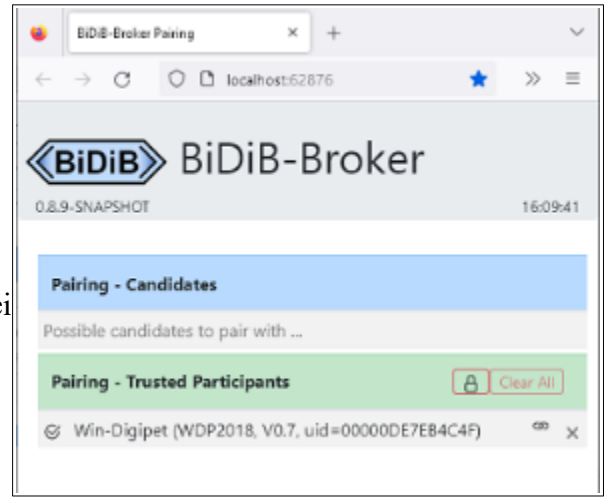
2 Die serielle Schnittstelle (BiDiBser) ist historisch bedingt vom „Pairing“ ausgenommen.

Nach erfolgreichem „Pairing“ wird der Teilnehmer durch das Kettenglied (∞) am rechten Rand angezeigt.

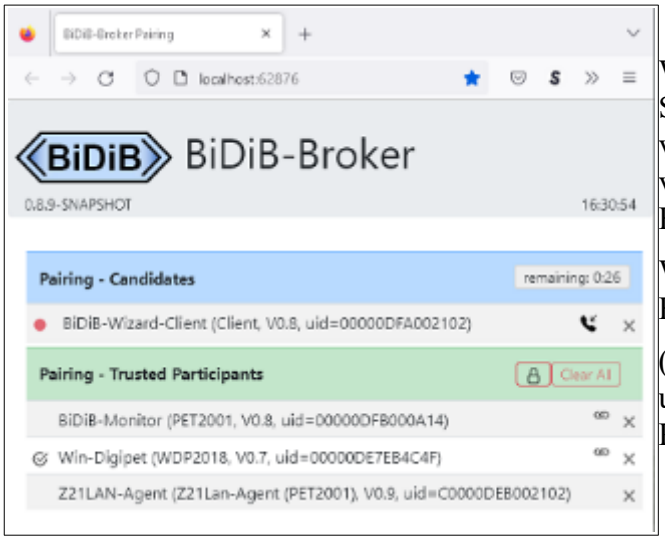
Der Kreis mit Haken am linken Rand zeigt, dass der Teilnehmer die Kontrolle über das BiDiB-System erhalten hat.

Mit dem eXit-Button rechts wird dem Teilnehmer das Vertrauen entzogen. Dann muss das „Pairing“ bei Bedarf erneut durchgeführt werden.

Um allen Teilnehmern gleichzeitig das Vertrauen zu entziehen, muss erst das rote Schloss geöffnet und dann der „Clear all“-Button betätigt werden.



Auf dem nächsten Bild sind vier netBiDiB-Teilnehmer mit unterschiedlichem Status zu sehen.



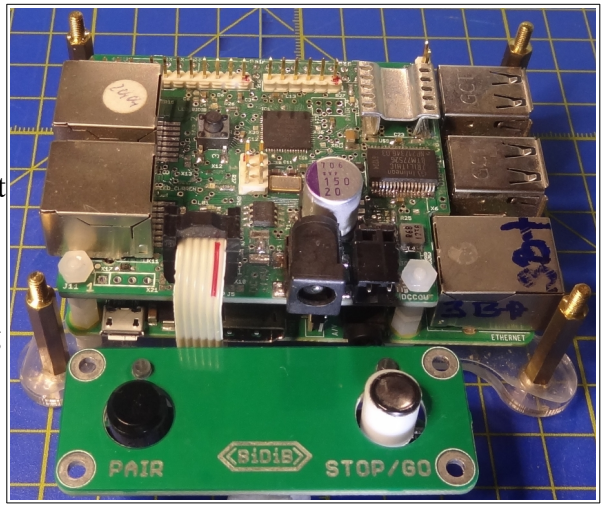
Win-Digipet hat die Kontrolle über das BiDiB-System, der BiDiB-Monitor ist vertraut und verbunden, Z21Lan-Agent ist vertraut aber nicht verbunden. Der BiDiB-Wizard wartet auf Bestätigung.

Weitere mit Broker über Port 62876 verbundene Browser, zeigen das Bild synchron.

(Die Pairing-API auf Port 62876 arbeitet unabhängig von der netBiDiB-Verbindung auf Port 62875.)

12.2 Pairing mit Raspberry Pi

Broker unterstützt zusätzlich das „Pairing“ mit Taster und LED[3] auf dem Raspberry Pi.



Auf dem Bild ist ein erstes Labormuster BiDiB-Pi1 auf einem Raspberry Pi3 B+ montiert. Im Vordergrund die Zusatzplatine mit der Pairing-Taste links unten und direkt darüber die Pairing-LED.

Meldet sich ein Teilnehmer zum „Pairing“ an, blinkt die Pairing-LED. Soll dem Teilnehmer vertraut werden, muss der Pairing-Taster innerhalb der konfigurierten Wartezeit (Vorgabe: 30 Sekunden) betätigt werden. Mit erfolgreichem Pairing-Vorgang leuchtet die Pairing-LED kontinuierlich.

Ist kein Vertrauter mehr verbunden, erlischt die Pairing-LED.

Soll Broker mit einem verbundenen, aber neuen Teilnehmer vertraut gemacht werden, kann der Pairing-Taster initial betätigt werden.

Soll allen bisherigen Vertrauten das Vertrauen entzogen werden, muss der Pairing-Taster mindestens die konfigurierte Wartezeit (Vorgabe: 3 Sekunden) betätigt werden.

3 Pairing mit Taster und LED auf anderen Raspberry Pi Systemen wird im Handbuch Embedded Systems beschrieben.

13 Fehlerbehandlung

13.1 GUI – Incompatibile SockJS

Die Verwendung der Pairing-GUI über die WebSocket-Schnittstelle bedingt die gleiche WebSocket-Version in Broker und Browser.

Wenn nicht, zeigt die Debug-Konsole im Browser (erreichbar z.B. mit F12) einen Fehler in der Art:

```
Error: Incompatibile SockJS! Main site uses: "1.1.2", the iframe: "1.0.0"
```

Dann muss das Betriebssystem, auf dem Broker läuft, aufgefrischt werden, z.B. mit:

```
sudo apt-get update && sudo apt-get dist-upgrade
```

von der Kommandozeile aus.

13.2 scm.dll/usb.dll: Can't find dependent libraries

Diese Exception kann auftreten, wenn es mindestens ein nicht funktionierendes Paket gibt: *Microsoft Redistributable Packages.*

Für die Installation einer funktionierenden dll-Datei, hilft die Suche nach *„vcredist_x64.exe 2013“* oder *https://support.microsoft.com/de-de/help/4032938/update-for-visual-c-2013-redistributable-package).*

(Siehe auch OpenDCC-Forum: https://forum.opendcc.de/wiki/doku.php?id=wizard#wizard_113)

Let akuhtz and tom_holzwurm speak (2020-08-12):

akuhtz: Could you download this tool: https://github.com/lucasg/Dependencies
Then select the file *C:\Users\<login>\AppData\Local\Temp\jbidibc-netbidib\scm.dll* and see which dependent library is missing?

tom_holzwurm: Of course: due to the "dependencies tool" MSVCR120.dll is missing.

akuhtz: Strange ... This is exactly what is installed with the *Visual C++ 2013 Redistributable Package*: https://support.microsoft.com/de-de/hel ... le-package (scroll to *More Information* --> x64 operating system: %WinDir%\System32)

In addition: With 64-bit-Version of Windows, you have to install both files: "vcredist_x86.exe" and "vcredist_x64.exe". With 32-bit-Version of Windows you have to install "vcredist_x86.exe" only, but not "vcredist_x64.exe". (see also in german: https://forum.opendcc.de/viewtopic.php?p=77372#p77372)

After installing the corresponding file(s) the SCM interface runs reliably again.

Siehe auch https://forum.opendcc.de/wiki/doku.php?id=wizard#download

13.3 Unable to load [libpi4j.so]

"*Unable to load [libpi4j.so] using path: [/lib/raspberrypi/dynamic/libpi4j.so]*"

Diese Exception tritt auf, wenn Broker mit einem Profil für den BiDiB-Pi gestartet wurde, z.B. **`bidib-pi-hat`** oder **`test-pi-hat`**. Die neueste OS enthält kein wiringPi mehr!

```
Upgrade:    cd /tmp
            wget https://project-downloads.drogon.net/wiringpi-latest.deb
            sudo dpkg -i wiringpi-latest.deb
Check:      gpio -v
```

(siehe auch: WiringPi updated to 2.52 (Gordon Henderson))

13.4 APPLICATION FAILED TO START

`ERROR: Web server failed to start. Port 62876 was already in use.`

In diesem Falle verwendet eine andere Anwendung das Port 62786, der für die Pairing-GUI zuständig ist.

Tipps zur Behebung des Problems siehe Wiki-FAQ.

14 Glossar

Begriff	Erklärung
Avatar/Avatarmode/ Avatar-Rolle	Stellvertreterbetrieb für physisch eng gekoppelte Baugruppen im BiDiB-System. Der Avatar tritt nicht nach Außen auf.
BiDiB	**BiDi**rektionaler **B**us (siehe www.bidib.org)
DCC-Mapping	Zuordnung von DCC-Adressen zu BiDiB-Accessorys und -Aspekten
Nebeneingang	Anschlussmöglichkeit für netBiDiB-Teilnehmer ohne Discovery
netBiDiB	Protokollergänzung für eine Verbindung via TCP/IP
Verteilte Steuerung	Protokollerweiterung zur Unterstützung teilautonomer BiDiB-Geräte