Hochschule Augsburg | Fakultät für Informatik | Sommer 2020
Prof. Dr. Högl | Embedded Linux | 2020-04-21

Memo-#3: **Versuche auf der Kommandozeile**

Inhalt

1 Schnittstellen des BBB **2**

2 Serielle Schnittstelle **4**

2.1 Welche serielle Schnittstelle ist an welchem Device? 4

2.2 Zugriffsrechte 6

2.3 Udev 6

3 Terminal Emulatoren **7**

4 Micro-SD Karte mit Debian beschreiben **7**

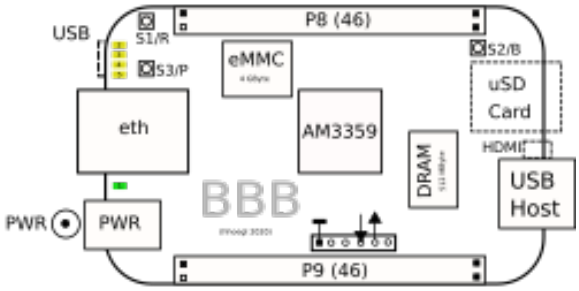
5 eMMC im BBB flashen **8**

6 Auf dem BBB umsehen **8**

Updates

```
2020-04-28 - udev rules, Python serial
```

1 Schnittstellen des BBB



- Mini USB Schnittstelle (oben links bei den gelben LEDs)

  Das ist ein "USB Gadget" Device. Die Funktion dieser Schnittstelle ist durch Software steuerbar. In unserem Fall verwaltet der laufende Linux Kernel dieses "Gadget Device". Das heisst natürlich auch, dass der Kernel laufen muss, damit ich diese Schnittstelle ansprechen kann. Dadurch ist es auch klar, dass man z.B. nicht die Kernel Bootmeldungen über diese Schnittstelle anzeigen kann, da ja der Kernel beim Booten noch nicht läuft. Folgende Funktionen liegen auf dieser USB Schnittstelle:

  - Serielle Schnittstelle (Terminal). Nur nach dem Booten!
  - Netzwerkverbindung
  - Massenspeicher
  - Stromversorgung des BBB

  Wer sich näher für das Linux Gadget Device interessiert, kann z.B. diesen Foliensatz ansehen:

  https://elinux.org/images/8/81/Useful_USB_Gadgets_on_Linux.pdf

- UART (Universal Asynchronous Receiver/Transmitter)

  Der UART wird über den 6-poligen Steckverbinder nahe am Stecker P9 angeschlossen. Es gibt nur zwei wichtige Signale: **RX** (Receive - Pfeil zum Pin) und **TX** (Transmit - Pfeil aus dem Pin). Ausserdem braucht man noch einen **Masse** Pin (ganz links der erste Pin).

  Über den UART kann man mit dem Board auf unterster Ebene kommunizieren, auch wenn noch kein Betriebssystem läuft. Er dient in unserem Fall vor allem dazu, den U-Boot Bootloader zu steuern und den Bootvorgang des Linux Kernel zu überwachen.

  Technisch wird diese Schnittstelle vom PC zum Targetrechner heutzutage meist als USB Kabel ausgeführt, das an einem Ende die Signale des UART an einen

Stecker herausführt. Von den 6 Kabeln braucht man aber wie schon gesagt nur drei. Im USB Anschluss ist auch ein Controller verbaut, der die Umwandlung von USB nach UART macht. In der folgenden Abbildung ist ein USB-zu-UART Kabel von FTDIChip gezeigt (mit Controller FT232 oder ähnlich), die Kabel haben meist die gezeigten Farben:

Der Stecker passt auch bereits auf den BBB UART Steckverbinder. Das schwarze Kabel ist die Masse. Der `lsusb` Befehl listet dieses Kabel als

```
Bus 003 Device 007: ID 0403:6001 Future Technology
Devices International, Ltd FT232 Serial (UART) IC
```

Es gibt auch UART Kabel von anderen Herstellern, verbreitet ist auch der Controller von Prolific PL2303HXA. Er hat vier Kabel mit den Farben schwarz (Masse), rot (5V -- nicht verwenden!), grün (TX) und weiss (RX).

Achtung Die Signale TX und RX sollen höchstens eine Spannung von 3,3 Volt haben. Es gibt auch Adapter, die für 5 Volt Logik vorgesehen sind. Diese sind **nicht** für das Beagleboard geeignet.

- USB Host

  An diese Schnittstelle würde man z.B. einen WiFi Dongle stecken. Ich verwende hier z.B. einen TP-LINK TL-WN725N.

- Ethernet (eth)

  Hier kann man ein Ethernetkabel anstecken um das BBB ans Netz zu bekommen. Es gibt also nun insgesamt zwei Möglichkeiten, um per Netzwerk auf das BBB zuzugreifen:

  1. Mini-USB Buchse mit Ethernet "Gadget"
  2. Ethernet Buchse (eth)

- Zentralstecker für 5V Spannungsversorgung

  Dieser Anschluss ist optional, da die Stromversorgung auch über den Mini-USB Anschluss möglich ist. Wenn man ein Gerät mit dem BBB baut, das dauerhaft im Betrieb mit Spannung versorgt werden muss, dann ist die Versorgung über den Zentralstecker die bessere Wahl.

2 Serielle Schnittstelle

Man sollte sich ansehen:

- Das `lsusb` Kommando
- Das `udevadm` Kommando
- Vendor- und Product-ID: VID:PID
- Gerätedateien
- Welcher serielle Adapter ist an welchem `/dev/tty*`?
- Terminal-Emulatoren: `picocom`, `putty`, `kermit`

2.1 Welche serielle Schnittstelle ist an welchem Device?

Wenn ich mir auf meinem Rechner alle `/dev/tty*` Gerätedateien anschaue, dann sieht es so aus:

```
tty        tty21      tty35      tty49      tty62      ttyS12     ttyS26
tty0       tty22      tty36      tty5       tty63      ttyS13     ttyS27
tty1       tty23      tty37      tty50      tty7       ttyS14     ttyS28
tty10      tty24      tty38      tty51      tty8       ttyS15     ttyS29
tty11      tty25      tty39      tty52      tty9       ttyS16     ttyS3
tty12      tty26      tty4       tty53      ttyACM0    ttyS17     ttyS30
tty13      tty27      tty40      tty54      ttyACM1    ttyS18     ttyS31
tty14      tty28      tty41      tty55      ttyACM2    ttyS19     ttyS4
tty15      tty29      tty42      tty56      ttyACM3    ttyS2      ttyS5
tty16      tty3       tty43      tty57      ttyACM4    ttyS20     ttyS6
tty17      tty30      tty44      tty58      ttyprintk  ttyS21     ttyS7
tty18      tty31      tty45      tty59      ttyS0      ttyS22     ttyS8
tty19      tty32      tty46      tty6       ttyS1      ttyS23     ttyS9
tty2       tty33      tty47      tty60      ttyS10     ttyS24     ttyUSB0
tty20      tty34      tty48      tty61      ttyS11     ttyS25
```

Die für uns wichtigsten Einträge sind `/dev/ttyACM*` und `/dev/USB*`. Je nachdem wie viele serielle Geräte man in welcher Reihenfolge eingesteckt hat, kann sich der Gerätename auch ändern.

Über `lsusb` findet man schon mal den VID:PID Eintrag heraus (VID = Vendor ID, PID = Product ID). Der USB Anschluss des BBB ist hier:

```
Bus 003 Device 010: ID 1d6b:0104 Linux Foundation Multifunction
Composite Gadget
```

Der USB UART für die Konsole sieht so aus:

```
Bus 003 Device 011: ID 0403:6001 Future Technology Devices
International, Ltd FT232 Serial (UART) IC
```

Mit `udevadm` kann man nun die Properties eines Gerätes ausgeben und nachschauen, ob VID:PID identisch sind:

```
udevadm info -q property -n /dev/ttyACM4
```

Bei mir kommt heraus, dass ttyACM4 der USB Anschluss des BBB ist. Der Konsolen-UART ist ttyUSB0.

Nun kann man den Terminal-Emulator mit dem richtigen Device und der richtigen Geschwindigkeit aufrufen:

```
picocom -b 115200 /dev/ttyUSB0    # Konsolen-UART (USB Kabel von FTDIChip)
picocom -b 115200 /dev/ttyACM4    # USB Anschluss des BBB
```

Aufpassen: Es kann gut sein, dass eine andere Linux Distribution nicht die angegebenen Geschwindigkeit von 115200 Baud hat sondern z.B. nur 9600 Baud!

Nachtrag (2020-04-28) Das Python Modul `serial` kann man auch wie folgt als Skript aufrufen, dann berichtet es, welche seriellen Schnittstellen gefunden wurden:

```
python -m serial.tools.list_ports -v
```

Die Ausgabe sieht in etwa wie folgt aus (ein paar Geräte habe ich weggelassen):

```
...
/dev/ttyACM3
  desc: BeagleBoneBlack - CDC Abstract Control Model (ACM)
  hwid: USB VID:PID=1D6B:0104 SER=3614BBBK0076 LOCATION=3-2.4.1:1.5
...
/dev/ttyUSB0
  desc: TTL232R-3V3
  hwid: USB VID:PID=0403:6001 SER=FTE4080F LOCATION=3-2.4.4
```

2.2 Zugriffsrechte

Auf dem Hostrechner sollte der Zugriff auf die serielle Schnittstelle immer als gewöhnlicher User funktionieren. Es sollte also nicht nötig sein, dass man dazu in die Administrator-Rolle (root) gehen muss.

Es genügt, wenn der Mitglied der `dialout` Gruppe ist. Mit dem `id` Kommando findet man heraus, welchen Gruppen ein User angehört. Mit `useradd` kann man den User in eine neue Gruppe aufnehmen:

```
usermod -a -G dialout <user>
```

Mit `cat /etc/group` sieht man alle Gruppen. Bei mir sieht der Eintrag mit Dialout wie folgt aus:

```
dialout:x:20:hhoegl
```

Wenn man nur diese Zeile sehen will, dann gibt man ein:

```
cat /etc/group | grep dialout
```

2.3 Udev

Über eine Konfigurationsdatei kann man genau einstellen, mit welchen Rechten und unter welchem Namen ein USB Gerät beim Einstecken eingebunden werden soll. Die Konfiguration liegt in Form einer "Regel" (z.B. `59-my-cp2102.rule`) im Verzeichnis `/etc/udev/rules.d/`.

```
# - 59-my-cp2102.rule -
# USB-to-UART bridge with CP2102 (www.silabs.com)
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", \
ATTRS{idProduct}=="ea60", MODE="0666", SYMLINK+="my_cp2102_port"
```

Die Zahl am Anfang des Namens der Regeldatei bestimmt die Reihenfolge, mit der mehrere Dateien eingebunden werden. Die Regel muss mit den folgenden Kommandos aktiviert werden:

```
$ udevadm control --reload-rules
$ udevadm trigger
```

Ein anderes Beispiel mit einem FTDIChip USB-to-Serial Adapter

```
# /etc/udev/rules.d/81-ftdi-usb-to-serial.rules
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", \
GROUP="users", MODE="0666", SYMLINK+="ftdi_uart"
```

Die Attribute eines seriellen Gerätes kann man mit dem `udevadm` Kommando ausgeben. Diese Attribute kann man auch in den Regeln verwenden:

```
udevadm info -a -n /dev/ttyUSB0 | grep '{serial}' | head -n1
```

Lit.: https://wiki.ubuntuusers.de/udev

3 Terminal Emulatoren

- picocom
- putty
- ckermit

4 Micro-SD Karte mit Debian beschreiben

Alte Mikro-SD Karte (4 GByte) in den Kartenleser gesteckt, um zu sehen was drauf ist:

```
$ lsblk
...
sdd      8:48   1   3,7G  0 disk
|-sdd1   8:49   1   100M  0 part /media/hhoegl/BOOT
|-sdd2   8:50   1   3,6G  0 part /media/hhoegl/ROOT
...
```

Das sieht nach einer bereits für das BBB beschriebenen Karte aus. Wir wollen diese Karte jedoch mit einem neuen Image überschreiben, deshalb aktuelles Debian Image holen von https://beagleboard.org/latest-images:

```
bone-debian-10.3-iot-armhf-2020-04-06-4gb.img.xz
```

Man muss das Image nicht entpacken.

Den "Balena Etcher" von https://www.balena.io/etcher holen. Im ZIP Archiv befindet sich ein "AppImage" für 64-Bit Linux, das man einfach mit `chmod 755 ...` ausführbar macht. Danach kann man es mit `./balenaEtcher-1.5.81-x64.AppImage` starten. Das Tool ist mit etwa 100 MByte ziemlich gross, Gründe dafür sind (a) es ist mit dem "Electron" Framework gemacht, und (b) es ist ein AppImage.

Im Etcher wählt man das Image und die Mikro-SD Karte aus, dann wird die Karte geschrieben. Nachdem die Karte fertig geschrieben wurde, wirft man sie aus und steckt

sie in das BBB. Vorher die Stromversorgung des BBB abschalten. Danach wieder einschalten, nun sollte das BBB von der Karte booten.

Auch empfohlen: Die Karte mit üblichen Kommandozeilenwerkzeugen beschreiben. Es geht ungefähr so:

1. Mit `fdisk` oder `cfdisk` zwei Partitionen anlegen, vom Typ eine (klein) für FAT16 Dateisystem, eine für EXT4 (gross).
2. Die Dateisysteme erstellen (den konkreten Wert /dev/sdX müssen Sie für Ihren Rechner einsetzen):

```
sudo mkfs.vfat -F 16 -n "boot" /dev/sdb1
sudo mkfs.ext4 -j -L "root" /dev/sdb2
```

3. Die Dateien zum Booten müssen in "boot" kopiert werden.
4. Das Root Filesystem muss in "root" kopiert werden.

Eine detaillierte Anleitung ist hier: http://www.armhf.com/boards/beaglebone-black/bbb-sd-install/

http://www.armhf.com/boards/beaglebone-black/bbb-sd-install/

5 eMMC im BBB flashen

Das geht im Prinzip wie im vorherigen Abschnitt "Mikro-SD Karte mit Debian beschreiben", ausser dass man nun das "Flasher Image" nimmt. Das Flasher Image wird auf die Mikro-SD Karte geschrieben, nach dem Booten erfüllt es nur den alleinige Zweck, dass es den eMMC Speicher mit einem Debian Linux beschreibt. Während dem Flashen sieht man viele Logmeldungen auf der Konsole:

http://hhoegl.informatik.hs-augsburg.de/elinux/ss20/flasher-log.txt

Nach dem Flashen muss man die Karte entfernen und gut beschriften, nicht dass sie versehentlich zum Booten verwendet wird. Sollte das passieren, wir sofort wieder der eMMC Speicher überschrieben.

6 Auf dem BBB umsehen

Booten von eMMC oder SD-Karte. Debian 10.3 (2020-04-20)

Verbinden vom Hostrechner auf dem BBB mit

```
picocom -b 115200 /dev/ttyACM3
```

(Serielle Schnittstelle über USB Kabel)

oder

```
ssh debian@192.168.7.2
```

(Netzwerkverbindung über USB Kabel - "Gadget Ethernet")

- Terminalfenster 80x25 (wenn man über USB-UART Konsole auf dem BBB ist)
- Es gibt den Terminal Multiplexer `tmux` auf dem BBB!

  Damit kann man über eine einzige serielle Verbindung zum BBB mehrere Terminals auf dem BBB starten, ohne dass man eine grafische Oberfläche benötigt.

  Mehr Infos dazu: http://hhoegl.informatik.hs-augsburg.de/elinux/tmux

  Meist möchte man die tmux Einstellungen ein wenig ändern und macht dies in einer Konfigurationsdatei `~/.tmux.conf`. Ich verwende z.B. statt des voreingestellten Kommando-Prefix `C-b` immer `C-z`. Meine `.tmux.conf` ist in

  https://r-n-d.informatik.hs-augsburg.de:8080/elinux/material/-/tree/master/tmux
- Editoren: `pico`, `nano`, `vi`, `vim`
- Pager: `less`
- `ip addr`
- Eingesteckter TL-WN725 WiFi Adapter wird sofort erkannt und ist betriebsbereit.

  Wie konfiguriert man WiFi auf der Kommandozeile?

  Es gibt die Kommandos `ip`, `iw`, `wpa_supplicant`, `wpa_cli`, ...
- Debian Version / Kernel

  ```
  debian@beaglebone:~$ cat /etc/debian_version
  10.3

  debian@beaglebone:~$ uname -a
  Linux beaglebone 4.19.94-ti-r42 #1buster SMP PREEMPT ...
    ... 2020 armv7l GNU/Linux
  ```
- Netzwerkverbindung vom Host zum BBB klappt.

  ```
  host $ ping 192.168.7.2
  ```
- Netzwerkverbindung vom BBB zum Host

  Auf dem Hostrechner gibt es eine "Wired Connection" im Network Manager, die per USB Kabel vom Host zum Target geht. Über "Edit Connections" diese Verbindung auf "Shared to Other Computers" setzen. Dadurch wird auf dem Host im Hintergrund der `dnsmasq` Server gestartet. Der Hostrechner kann entweder über ein Netzwerkkabel am Internet hängen oder über WiFi:

```
4 S nobody    2227  1108  0  80   0 -  3761 -  21:08 ?  00:00:00 \
/usr/sbin/dnsmasq --conf-file=/dev/null --no-hosts --keep-in-foreground
--bind-interfaces --except-interface=lo --clear-on-reload --strict-order
--listen-address=10.42.0.1 --dhcp-range=10.42.0.10,10.42.0.254,60m \
--dhcp-lease-max=50 \
--dhcp-leasefile=/var/lib/NetworkManager/dnsmasq-enxd05fb8e8e3db.leases
--pid-file=/run/nm-dnsmasq-enxd05fb8e8e3db.pid \
--conf-dir=/etc/NetworkManager/dnsmasq-shared.d
```

Danach sollte man sofort vom BBB aus das Internet sehen:

```
ping www.hs-augsburg.de
```

- Debian updaten

  ```
  debian@beaglebone:~$ sudo apt-get update
  ```

- Welche Prozesse laufen auf dem BBB? Sie sollten das mit dem Debian im eMMC Speicher testen!

  ```
  ps -e | less
  ```

  Kernel Threads interessieren nicht, sie sollten sich auf die Prozesse im User-Space beschränken.

- Im Web-Browser auf dem Hostrechner zur "Cloud 9" IDE gehen:

  http://192.168.7.2/ide.html

  Diese IDE ist aber für diesen Kurs nicht von Bedeutung.