

FiFi-Radio Linux-Treiber

DF9DQ

18. März 2006

Zusammenfassung

Dieses Dokument beschreibt die Installation und Anwendung eines Linuxtreibers für das **FiFi-Radio** des DARC e.V. OV Lennestadt, O28. Eine Übersicht über dieses FM-Radio-Projekt, sowie Schaltplan und Firmware findet man auf der Webseite <http://www.ov-lennestadt.de/Projekte/ukw-radio/>.

Dieses UKW-Radio ist mit einer seriellen Schnittstelle ausgerüstet. Über diese Schnittstelle kann man Kommandos zum Radio senden, um z.B. die Frequenz einzustellen. Das Radio sendet in umgekehrter Richtung zum PC hin Statusinformationen über eingestellte Frequenz, Empfangsfeldstärke, Abstimmung usw. und zusätzlich rohe RDS-Daten vom RDS-Demodulator auf dem FiFi-Radio.

Der hier beschriebene Linux-Treiber ermöglicht es, das Radio als Multimediagerät unter dem Betriebssystem Linux einzusetzen. Damit kann man jede beliebige standardkonforme Radio-Anwendung unter Linux für die Steuerung des Radios einsetzen. Beispiele für solche Anwendungen sind „GRadio“, „Gnomeradio“ oder „kradio“. Es gibt auch viele Anwendungen, die mit einem Plugin zur Radiosteuerung geeignet sind, wie z.B. „XMMS“¹.

Der Treiber darf unter den Bedingungen der GNU General Public License Version 2 benutzt werden.

1 Installation

Wenn der Treiber erstmalig auf einem System installiert werden soll, dann werden die dazu benötigten Schritte ab dem nächsten Kapitel beschrieben.

Wenn aber bereits alles vorinstalliert ist, dann kann man der folgenden Kurzbeschreibung aller notwendigen Befehle zum Starten der Treiber und Anwendungen folgen:

1. Treiber laden (*fifradio*)
2. Schnittstelle aktivieren (*fifiattach*)
3. Anwendung starten (z.B. *gradio*)
4. RDS-Dämon starten (*rdsd*)
5. RDS-Client starten (*rdsquery* oder darauf aufbauende Anwendung, wie z.B. ein *karamba*-Theme.)

Zum Herunterfahren einfach die Schritte in umgekehrter Reihenfolge rückgängig machen. Jede Abweichung davon, z.B. Beenden von *fifiattach*, wenn noch die Radio-Anwendung läuft, kann dazu führen, daß das System instabil wird und abstürzt.

¹xmms-fmradio ist eines der beschissensten Programme die ich kenne :-)

1.1 Voraussetzungen

Getestet wurde der Treiber auf einem Debian unstable mit eigenem Kernel 2.6.10 und dem GNU-Compiler 3.3.6. Auf allen anderen Systemem kann eine Funktion nicht garantiert werden. Im übrigen auch nicht auf dem gerade beschriebenen System ;-)

Da der Treiber selbst compiliert werden muß (es gibt keine Paketversion), müssen neben den dafür nötigen Standardtools auch die Quelltexte des aktuell installierten Kernels verfügbar sein. Die Quellen werden im Verzeichnis `/usr/src/linux` erwartet. Jede anständige Distribution bietet ein Paket an, mit dem sich die Quellen/Header des verwendeten Kernels problemlos installieren lassen. Einige entpacken die Quellen passend nach `/usr/src/linux`, bei anderen (Debian) muß man die Quelltexte in `/usr/src` nach der Installation des entsprechenden Pakets erst entpacken und anschließend einen symbolischen Link `/usr/src/linux` auf dieses Quellenverzeichnis anlegen.

1.2 Kerneltreiber

Das Archiv mit dem Quelltexten des Treibers zunächst entpacken, wie im folgenden Beispiel gezeigt:

```
$ tar -xvjf fifiradio-0.8.3.tar.bz2
$ cd fifiradio-0.8.3
```

Normalerweise ist in `Makefile` keine Änderung vorzunehmen. Aber ein kurzer Blick schadet auch nicht.

Bevor man mit der Übersetzung des Treibers startet, sollte man überprüfen, ob der aktuelle Compiler die gleiche Version wie derjenige hat, der zur Übersetzung des eingesetzten Linux-Kernels verwendet wurde. Um herauszufinden, welche Version von `gcc` für die Übersetzung des Kernels benutzt wurde, kann man einfach

```
$ /sbin/modprobe videodev
```

eingeben. Am Ende der mit `vermagic` bezeichneten Zeile steht dann z.B. `gcc-3.3`, woraus sich ergibt, daß der Compiler mit der Versionsnummer 3.3.x verwendet wurde.

Wenn mehr als nur eine Version des Compilers installiert ist, dann muß gegebenenfalls der symbolische Link `gcc` auf die eigentliche ausführbare Datei des Compilers mit der richtigen Version umgestellt werden, z.B. `gcc-3.3`. Als Beispiel hier die entscheidenden Dateien auf einem Debian-System im Verzeichnis `/usr/bin`:

```
lrwxrwxrwx 1 root root      7 Dec 30 23:45 /usr/bin/gcc -> gcc-3.3
-rwxr-xr-x 1 root root 74104 Oct 29 07:38 /usr/bin/gcc-2.95
-rwxr-xr-x 1 root root 81648 Sep 16 13:59 /usr/bin/gcc-3.3
-rwxr-xr-x 1 root root 84752 Oct 16 01:10 /usr/bin/gcc-3.4
-rwxr-xr-x 1 root root 89136 Dec  2 01:12 /usr/bin/gcc-4.0
lrwxrwxrwx 1 root root     10 Dec 30 23:45 /usr/bin/gccbug -> gccbug-3.3
-rwxr-xr-x 1 root root 15967 Sep 16 13:49 /usr/bin/gccbug-3.3
-rwxr-xr-x 1 root root 16125 Oct 16 01:02 /usr/bin/gccbug-3.4
-rwxr-xr-x 1 root root 16239 Dec  2 01:06 /usr/bin/gccbug-4.0
```

Die Links `gcc` und `gccbug` müssen eventuell umgelenkt werden.

Der Treiber kann nun übersetzt werden:

```
$ make
```

Wenn alles geklappt hat, kann der Treiber nun installiert werden. Dazu bitte das `root`-Paßwort bereithalten :-)

```
$ make install
```

Der Treiber kann nun ganz einfach in den Kernel geladen werden. Dazu bitte als `root` anmelden (mit `su`):

```
$ su
# modprobe fifiradio
```

Das sollte ohne Fehlermeldung geklappt und wieder den Prompt gezeigt haben. Wenn die Compilerversionen nicht wie oben beschrieben übereinstimmen, kommt typischerweise folgende Fehlermeldung:

```
FATAL: Error inserting fifiradio
(/lib/modules/2.6.10.20051030/misc/fifiradio.ko): Invalid module format
```

Wenn alles erfolgreich gelaufen ist, sollte der Treiber in der Liste der geladenen Module auftauchen. Diese Liste kann man nun auf die korrekt geladenen Module hin überprüfen:

```
# lsmod|sort|grep fifiradio
```

Dabei sollte die folgende Liste herauskommen, wobei die Speichergrößen möglicherweise von den hiergezeigten Werten abweichen:

```
fifiradio          11528  0
v4l1_compat        15620  1 fifiradio
v4l2_common         6912  1 fifiradio
videodev           10880  1 fifiradio
```

Wichtig ist, daß alle diese Module auftauchen.

Wenn man einen Blick mit

```
$ dmesg
```

in das Systemlog wirft, sollten dort die folgenden Zeilen hinzugekommen sein:

```
FiFi FM Radio driver, version 0.8.3, supporting Video4Linux 2
fifiradio: successfully registered line discipline #2
```

1.2.1 Modul-Parameter

Beim Laden eines Moduls in den Kernel kann man dem Modul Parameter mitgeben. Für *fifiradio* sind nachfolgend die definierten Parameter mit den jeweils zulässigen Werten aufgeführt.

- Parameter *radio_first_dev*
Normalerweise wird das erste freie Radio-Gerät (`/dev/radioX`) verwendet. Wenn man ein ganz bestimmtes Gerät mit dem FiFi-Radio-Treiber benutzen möchte, kann man dessen Nummer mit diesem Parameter angeben. Der Treiber versucht dann, genau dieses Gerät zu reservieren.

Beispiel: `modprobe radio_first_dev=3` versucht `/dev/radio3` anstelle des ersten freien Gerätes (meist `/dev/radio0`) zu reservieren.

- Parameter *debug*
Dieser Parameter ist wohl nur bei der Entwicklung des Treibers hilfreich. Normalerweise schreibt der Treiber nur wenige Meldungen ins Log. Mit der Angabe von *debug* kann man ihn etwas gesprächiger machen. Dazu wird gibt man als Argument die Summe der unten in der ersten Tabellenspalte gelisteten Zahlen an, wobei man natürlich nur die Zahlen berücksichtigen muß, deren zugeordnete Meldungen man sehen möchte!

1	RDS	Einige Meldungen im Zusammenhang mit RDS
2	mehr RDS	Noch detailliertere Meldungen im Zusammenhang mit RDS
4	Line discipline	Meldungen im Zusammenhang mit dem tty-Layer
8	Video4Linux	Meldungen im Zusammenhang mit dem Video4Linux-Interface
16	Radio Control	Zeigt Details zu Meldungen, die über den Kontroll-Kanal vom und zum Radio abgewickelt werden

Beispiel: `modprobe fifiradio debug=24` zeigt Meldungen von der V4L-Schnittstelle sowie die Kontrolldaten die zwischen Treiber und Radio ausgetauscht werden.

- Parameter *line_discipline_number*

Die standardmäßig verwendete *line discipline* hat die Nummer 2. Nicht auf jedem System scheint diese Nummer noch frei zu sein, deshalb kann man mit diesem Modul-Parameter einen alternativen Wert angeben. Der einzig sinnvolle Wert dafür ist die 9. Eine ausführlichere Erklärung zum Thema *line discipline* findet man im Kapitel 3.1 auf Seite 8.

Beispiel: `modprobe fifiradio line_discipline_number=9` wählt die alternative *line discipline* aus.

1.3 Usermode-Programm *fifiattach*

Der Kerneltreiber stellt nur den Mechanismus bereit, um ein FiFi-Radio ansteuern zu können. Um eine bestimmte serielle Schnittstelle für den Betrieb eines FiFi-Radios auszuwählen, braucht man noch ein normales Anwendungsprogramm. Dieses Programm heißt *fifiattach*. Nach dem Laden des Kerneltreibers muß man dieses Programm mit der Angabe der gewünschten Schnittstelle aufrufen. Anschließend kann dann eine Radioapplikation auf das FiFi-Radio zugreifen.

Zur Installation des Programms wird zunächst das Archiv entpackt:

```
$ tar -xvzf fifiattach-0.2.tar.gz
$ cd fifiattach-0.2
```

Das Programm kann nun übersetzt werden:

```
$ ./configure
$ make
```

Wenn man das Programm permanent installieren will, kann man es folgendermaßen tun:

```
$ su
# make install
```

Angenommen, das FiFi-Radio ist an der ersten seriellen Schnittstelle des Rechners angeschlossen. Dann aktiviert man den Treiber wie folgt:

```
$ fifiattach /dev/ttyS0
```

Für andere Schnittstellen ist der entsprechende Device-Name einzutragen, z.B. `/dev/ttyS1` für die zweite serielle Schnittstelle oder `/dev/ttyUSB0` für die erste serielle Schnittstelle an einem USB-auf-Seriell-Adapter.

Das Programm läuft derzeit nach der Aktivierung der Schnittstelle in einer Endlosschleife. Wenn man später die Schnittstelle mit dem FiFi-Radio wieder freigeben will, muß man das Programm mit *Strg-c* beenden.

Nach erfolgreichem Start von *fifiattach* steht im System nun ein Radio-Device zur Verfügung (`/dev/radio0`, wenn kein anderes Radio im System ist). Wenn man das Systemlog mit

```
$ dmesg
```

abfragt, sollten nun folgende Zeile hinzu gekommen sein:

```
fifiradio: client connected @ ttyUSB0
fifiradio: registered device /dev/radio0
```

Falls schon andere Radios im System eingebaut sind, verwendet der Treiber statt `/dev/radio0` die nächste freie Gerätenummer, also z.B. `/dev/radio1`.

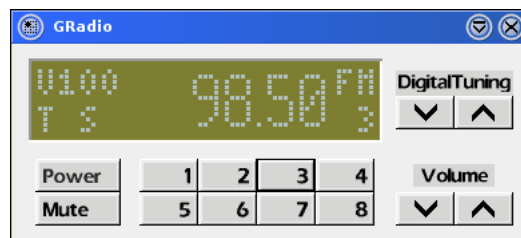
2 Anwendungsprogramme

Um das FiFi-Radio bedienen zu können, gibt es eine ganze Reihe von Applikationen unter Linux. Es sollte jede Applikation funktionieren, die auf das standardisierte Video4Linux-Interface aufsetzt. Der Kerneltreiber für das FiFi-Radio unterstützt die Version 2 von Video4Linux, ist aber auch rückwärts kompatibel mit Anwendungen, die nur die Version 1 von Video4Linux kennen.

Zusätzlich zu den normalen Radio-Programmen unterstützt der Treiber auch solche Programme, die die RDS-Daten einer empfangenen Station auslesen wollen. Beschrieben wird weiter unten das Programm *rdsd*, das zusammen mit der RDS-Bibliothek *librds* ebenfalls die standardisierte Video4Linux-Schnittstelle zum Empfang der RDS-Daten benutzt.

2.1 Radio-Anwendung GRadio

GRadio ist ein recht einfaches Programm, das neben der Frequenzeinstellung auch einige Stationstasten für das Anspeichern von Sendern bietet. Die Funktion ist zuverlässig, allerdings ist die Bedienung sehr gewöhnungsbedürftig.



GRadio beachtet das vom Kerneltreiber mitgeteilte Empfangsband, d.h. der Kerneltreiber kann bestimmen, welche Frequenzen einstellbar sind. Damit ist GRadio konform zur Video4Linux-Spezifikation.

Die Tasten für DigitalTuning ändern die Frequenz im 200-kHz-Raster, wenn man sie mit der linken Maustaste anklickt. Dieses Raster ist sehr unglücklich gewählt, weil die Sender im 100-kHz-Raster arbeiten². Betätigt man die Tasten mit der rechten Maustaste, dann ändert sich die Frequenz um jeweils 10 kHz. Durch Anklicken der Frequenzanzeige kann man die gewünschte Frequenz auch direkt eingeben.

Die aktuelle Frequenz kann man einer der Stationstasten zuweisen, wenn man die gewünschte Taste mit der rechten Maustaste anklickt. Durch Klicken mit der linken Maustaste auf eine der Stationstasten wird der gespeicherte Sender wieder eingestellt. Eine Benennung der Tasten mit Texten ist leider nicht möglich.

Ohne Argumente auf der Kommandozeile verwendet GRadio das Radio-Gerät `/dev/radio0`. Wenn das Radio einem anderen Gerät zugeordnet ist, muß man über einen Parameter auf der Kommandozeile den Gerätenamen angeben. Für das Gerät `/dev/radio1` muß man folgendes Argument verwenden:

```
$ gradio -d /dev/radio1
```

GRadio ist eine steinalte Software und wird nicht mehr weiterentwickelt. Was alt ist muß nicht schlecht sein, aber GRadio läßt in der Originalversion (das ist die 1.0.1) doch einige sinnvolle Eigenschaften vermissen. Es gibt deshalb eine geänderte Version 1.0.2, die man sich von der gleichen Webseite herunterladen kann auf der auch der FiFi-Radio-Treiber steht. Hier wird kurz beschrieben, wie man die Software kompiliert, anschließend werden die neuen Programmeigenschaften beschrieben.

Übersetzt und installiert wird das Programm einfach so:

²In der O28-Version geändert, s.u.

```
$ tar -xvzf gradio_1.0.2-2.tar.gz
$ cd gradio
$ ./configure
$ make
# make install
```

Die wesentlichen Erweiterungen dieser „O28-Version“ sind die folgenden:

1. Großer Frequenzschritt nun 100 kHz (vorher 200 kHz).
Änderbar über Kommandozeilenschalter `-s`. Der Parameter gibt die Schrittweite in kHz an.
2. Das Programm verwendet nun die Video4Linux-API Version 2, und wird damit auch auf den Kernen ab Sommer 2006 laufen. Ab Juni 2006 wird in den neuen Kernen die Unterstützung für die API-Version 1 eingestellt.
3. Die Anzeige „T“ im Display wird aktiviert, wenn die Empfangsfeldstärke oberhalb einer einstellbaren Schaltschwelle liegt. Diese Schaltschwelle kann man mit dem Kommandozeilenschalter `-t` steuern. Der Wertebereich für diesen Parameter ist 0...65535, wobei der Standardwert 16384 ist.
4. Die nutzlose Daueranzeige des Buchstabens „K“ wurde ersetzt durch eine Stereo-Anzeige. Wenn ein Stereo-Signal empfangen wird, wird an dieser Stelle ein „S“ angezeigt.
5. Wenn die aktuell eingestellte Frequenz bereits in einem der Speicherkanäle abgelegt ist, dann wird die Nummer dieses Speicherkanals angezeigt. Dadurch kann man sofort erkennen, ob der Sender schon abgespeichert ist oder nicht.
6. GRadio liest nun in einstellbaren Intervallen den aktuellen Status des Treibers aus. Frequenzänderungen die man am FiFi-Radio direkt vorgenommen hat, werden somit direkt in die Anzeige von GRadio übernommen! Die Standardeinstellung ist hier, daß keine automatische Abfrage stattfindet, aber mit dem Kommandozeilenschalter `-u` läßt sich das Intervall in Millisekunden zwischen zwei Updates einstellen.

Die vollständige Liste aller Kommandozeilenschalter kann man in der Manpage von GRadio nachlesen.

Als Beispiel für die Verwendung der neuen Parameter hier ein Aufruf von GRadio für ein Radiogerät `/dev/radio0`, eine Schaltschwelle für die Feldstärkeanzeige von 50%, ein Update-Intervall von 200 ms für die Anzeige und eine Abstimm-Schrittweite von 100 kHz:

```
$ gradio -t 32768 -u 200 -s 100
```

2.2 Radio-Anwendung Gnomeradio

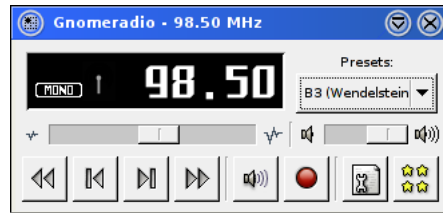
Gnomeradio wird noch aktiv weiterentwickelt. Gespeicherten Stationen kann man einen Namen geben, um sie dann später aus einer Senderliste auswählen zu können.

Die vom Treiber angegebenen Frequenzgrenzen werden von dieser Anwendung ignoriert, der Bereich 87.5 MHz bis 108 MHz ist fest eingestellt.

Leider hat das Programm derzeit einen Fehler bei der Frequenzeinstellung. Zum Treiber wird eine Frequenz gemeldet, die um ca. 30 kHz zu hoch ist!

2.3 RDS-Dämon rdsd

Das FiFi-Radio besitzt einen RDS-Demodulator, dessen Signale vom Kernel-Treiber aufbereitet und über die V4L2-Schnittstelle gelesen werden können. An diese Schnittstelle dockt das hier beschriebene Programm *rdsd* an und dekodiert die RDS-Daten, um dann Informationen wie Sendername, Infotexte und Verkehrsinformationen an einer Schnittstelle (UNIX-Socket) bereitzustellen.



Im Paket mit *rdsd* kommen noch zwei andere Programme, eine Bibliothek, *librds*, und ein Anwendungsprogramm, *rdquery*. Die Bibliothek dient dazu, einem Anwendungsprogramm den Zugriff auf die Daten des *rdsd* zu erleichtern. Das Kommandozeilen-Programm *rdquery* macht von dieser Bibliothek Gebrauch.

2.3.1 Installation

Bevor mit der eigentlichen Installation begonnen werden kann, muß man sicherstellen, daß Programm-Bibliotheken im Verzeichnis */usr/local/lib* vom System erkannt werden. Dazu muß die Datei */etc/ld.so.conf*³ folgende Zeile enthalten:

```
/usr/local/lib
```

Falls man die Zeile erst hinzufügen mußte, muß die Änderung dem System bekannt gemacht werden, indem man folgendes (als *root*) aufruft:

```
# ldconfig
```

Die Quellen des Pakets findet man auf der Seite <http://rdsd.berlios.de/index.html>, zum Download muß man allerdings das Programm *svn* benutzen, das zum Versionsverwaltungs-Tool *subversion* gehört⁴. Nach dem Download sollte man den Instruktionen in der beiliegenden Dokumentation folgen. Die Reihenfolge beim Übersetzen der drei Programme ist: Zuerst *librds*, dann *rdsd* und schließlich *rdquery*.

Der bei früheren Versionen des FiFi-Radio-Treibers (bis V0.5) benötigte Patch von *rdsd* ist nicht mehr nötig! Dem alten Treiber fehlte auf der V4L2-Schnittstelle eine *poll* Funktion. Dadurch konnte ein Client nicht vor Aufruf der *read* Funktion erkennen, ob Daten zur Verfügung standen.

Seine Standard-Einstellungen liest der Dämon beim Start aus der Konfigurations-Datei */etc/rdsd.conf*. Das folgende Listing zeigt eine minimale Version dieser Datei, wie sie für das FiFi-Radio benötigt wird. Wenn keine anderen RDS-Quellen im System vorhanden sind (das ist ganz sicher der Normalfall), ist diese Beispielkonfiguration völlig ausreichend.

```
[global]
unix-socket = "/var/tmp/rdsd.sock"
tcpip-port = 4321
logfile = "/var/tmp/rdsd.log"
pidfile = "/var/tmp/rdsd.pid"
console-log = yes
file-log = yes
loglevel = 5

[source]
name = "FiFi Radio"
path = "/dev/radio0"
type = "radiodev"
tunerfreq = 0
```

³So heißt die Datei auf Debian-Systemen. Eventuell hat sie in anderen Linux-Distributionen einen unterschiedlichen Namen.

⁴Inzwischen scheint es auch Tar-Archive zum Download zu geben

Der letzte Eintrag `tunerfreq = 0` sorgt dafür, daß `rdsd` nicht versucht beim Start die Frequenz des Radios einzustellen. Das sollte schließlich Sache einer der zuvor beschriebenen GUIs für die Bedienung des Radios sein.

2.3.2 Betrieb

Um den Dämonen starten zu können, muß natürlich das Radio-Gerät (z.B. `/dev/radio0`) bereits existieren, d.h. der Kernel-Treiber muß geladen und die Schnittstelle mit `fifiattach` geöffnet worden sein. Dann kann der Dämon einfach gestartet werden, am besten zunächst einmal nicht als eigentlicher Dämon im Hintergrund, sondern als Vordergrund-Anwendung in einer eigenen Konsole, damit man Log-Meldungen direkt sieht. Später dann, wenn alles funktioniert, kann man natürlich `rdsd` auch als normalen Dämonen im Hintergrund betreiben.

```
$ rdsd
```

Dann sollte sich ungefähr folgendes zeigen:

```
... RDS handler initialized.
... Added source definition: FiFi Radio
... RDS sources setup OK.
... Unix domain socket created and listening.
... TCP/IP socket created and listening.
... Using V4L2 for FiFi Radio
... Source opened: FiFi Radio
```

Der Standard-Client für den `rdsd`-Dämon ist das Tool `rdsquery`. Mit diesem Tool kann man den Programm-Namen des eingestellten Senders, den Infotext, Datum/Uhrzeit, Verkehrsmeldungen usw. vom Dämon auslesen. Die Ausgabe der angeforderten Informationen erfolgt auf der Kommandozeile. Andere Tools können dadurch `rdsquery` verwenden, dessen Ausgabe filtern und auf eigene Art und Weise anzeigen. Im Anschluß wird ein Beispiel mit dem Tool `karamba` gezeigt.

Um den Namen des eingestellten Radiosenders abzufragen, ruft man `rdsquery` so auf:

```
$ rdsquery -s localhost -c 0 - t pname
```

Die Ausgabe wird ständig fortgesetzt, bis das Programm mit `Strg-c` beendet wird. Ein Beispiel für die Ausgabe:

```
pname: BAYERN 3
```

3 Funktionsprinzip

Die Angaben in diesem Kapitel sind nur für denjenigen interessant, der mehr über die Funktionsweise des Treibers erfahren will.

3.1 tty und die Line Discipline

Zwischen dem Treiber für den physikalischen Schnittstellenbaustein im PC und dem Anwendungsprogramm liegt bei Linux die sogenannte *line discipline*. Die *line discipline* kann die Daten einfach unverändert durchreichen, so wie es die Standard-Variante `N_TTY` im wesentlichen nur macht. Sie kann aber auch den Datenverkehr auf der seriellen Schnittstelle vollständig übernehmen und die Daten überhaupt nicht mehr über den zugeordneten `tty`-Kanal (z.B. `/dev/ttyS0`) leiten. Stattdessen kann der Treiber der eine solche *line discipline* implementiert, die Daten auf einer vollständig anderen Gerätedatei bereistellen. Der FiFi-Radio-Treiber benutzt die *line discipline* um damit das Protokoll auf der seriellen Schnittstelle zum Radio abzuwickeln, bietet aber zum Anwendungsprogramm ein Radio-Gerät

/dev/radioX über die Video4Linux-Umgebung an. Ein standardisierter Steuerbefehl über den *ioctl*-Kanal der Video4Linux-Schnittstelle wird im Treiber umgewandelt in das entsprechende Steuerkommando auf der seriellen Schnittstelle.

Eine *line discipline* muß einer geöffneten seriellen Schnittstelle zugewiesen werden. Nach dem Öffnen einer seriellen Gerätedatei ist die Standard-Einstellung für die *line discipline* N_TTY, d.h. eine transparente Verbindung von Gerätedatei /dev/ttyX und Schnittstelle. Über einen *ioctl*-Aufruf muß der seriellen Schnittstelle nach dem Öffnen mitgeteilt werden, daß ein FiFi-Radio angeschlossen ist. Der Treiber verwendet die *line discipline* N_MOUSE (Zahlenwert 2), die von keinem anderen Systemtreiber verwendet wird. Leider gibt es keine bessere Möglichkeit eine eigene Nummer für eine *line discipline* zu vergeben. Sollte man einmal Probleme bekommen, weil man einen weiteren Treiber verwenden will, der auf diese Art arbeitet, könnte man noch auf eine andere sehr wahrscheinlich unbenutzte *line discipline* ausweichen: N_R3964 (Zahlenwert 9). Es sei denn, man würde eine Siemens-SPS an seinem Rechner betreiben :-)

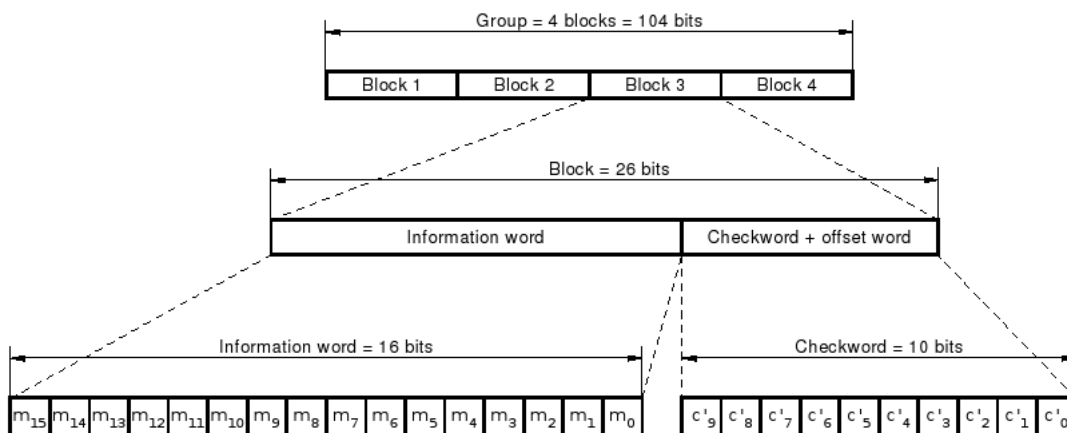
Wenn die *line discipline* von N_MOUSE auf N_R3964 geändert werden soll, muß man lediglich einen Parameter beim Laden des Treibers angeben. Der Parameter *line_discipline_number* hat dabei nur einen sinnvollen Parameter, nämlich 9 (N_R3964). Der Wert wird allerdings nicht überprüft, also Vorsicht bei der Eingabe!

In der Modul-Initialisierung wird dem Kernel über einen Aufruf von *tty_register_ldisc* mitgeteilt, daß der FiFi-Treiber ab jetzt für die *line discipline* N_MOUSE zuständig ist. Beim Entfernen des Moduls aus dem Speicher wird bei Kernen vor 2.6.12 die gleiche Funktion auch zur Abmeldung verwendet, ab Kernel 2.6.12 gibt es dafür die neue Funktion *tty_unregister_ldisc*.

Wenn eine serielle Schnittstelle auf die *line discipline* N_MOUSE umgestellt wird (das geschieht mit dem Tool *fifiattach*), dann ruft der Kernel die Funktion *lfifi_open* auf. Dort wird dann ein neues Radio-Gerät, z.B. /dev/radio0 angelegt. Vor dem Schließen des seriellen Kanals ruft der Kernel die Funktion *lfifi_close* auf, sodaß der Treiber das zuvor angelegte Radio-Gerät wieder entfernen kann. Beim Empfang von Zeichen auf der seriellen Schnittstelle informiert der Kernel den Treiber durch den Aufruf von *lfifi_receive_buf*. In dieser Funktion werden die Datenströme für Kontrollmeldungen und RDS-Daten getrennt und an die zuständigen Funktionen zur Dekodierung übergeben.

3.2 RDS-Synchronisation und -Fehlerkorrektur

Im nachfolgenden Bild (aus [5]) ist die Struktur der RDS-Übertragung dargestellt. Die größte Einheit ist die Gruppe, eine Folge von 104 Bits. Eine Gruppe besteht aus vier Blocks von je 26 Bits, die sich jeweils zusammensetzen aus 16 Nutzbits und 10 Prüfbits. Nutzbits und Prüfbits bilden zusammen einen zyklischen Code mit speziellen Eigenschaften für die Korrektur von Übertragungsfehlern [4, S.96ff].



3.2.1 Blocksynchronisation

Wenn der Treiber noch keine Ahnung hat, wo der Beginn eines Blocks oder einer Gruppe im empfangenen Datenstrom ist, berechnet er einfach mit jedem hereinkommenden Bit die Fehleranzahl im letzten Block unter der Annahme, daß die letzten 26 Bits zum Block A gehörten. Wenn er dabei null Fehler feststellt, waren die letzten 26 Bits mit sehr hoher Wahrscheinlichkeit ein gültiger Block A. Um sicher zu sein, gilt die Synchronisation erst dann als erreicht, wenn ein zweiter fehlerfreier Block A im Abstand von Vielfachen von 104 Bits erkannt wird.

Da die Codewörter in allen Blöcken nach der gleichen Rechenvorschrift gebildet werden, würde ohne weiteres ein gültiges Codewort aus Block A nicht von gültigen Codewörtern aus anderen Blöcken unterscheidbar sein. Deshalb wird vom Sender auf die Codewörter je nach Block ein unterschiedlicher Offset addiert. Dieser Offset ermöglicht es, bei der anfänglichen Synchronisation den genauen Block erkennen zu können.

3.2.2 Fehlerkorrektur

In der RBDS-Spezifikation [5] (die US-amerikanische Variante von RDS) steht, daß dieser verwendete zyklische Code in der Lage ist, sogenannte *Bündelfehler* mit einer Länge von fünf Bits zu korrigieren. Somit können alle Übertragungsfehler die innerhalb eines Abschnitts von fünf Bit Länge in einem Block auftreten korrigiert werden. Treten dagegen Fehler auf, die so weit auseinander liegen, daß sie nicht innerhalb eines Bündels von fünf Bits liegen, sind sie nicht korrigierbar.

Eine Realisierung einer entsprechenden Korrekturschaltung basierend auf einem Schieberegister ist in der Spezifikation angegeben. Die Fehlerkorrektur im Treiber folgt genau diesem Schema. Dabei muß man bedenken, daß verschiedene Fehlerbilder zum gleichen Fehlersyndrom führen können. Ein Syndrom ist deshalb nicht eindeutig, und eine Korrektur wird nur dann gelingen, wenn das Syndrom aus einem Bündelfehler hervorgegangen ist.

Ein gültiges Codewort (Informationswort + Prüfwort) kann man daran erkennen, daß das Fehlersyndrom null ist. Nach der versuchten Fehlerkorrektur wird deshalb noch einmal das Syndrom für das resultierende Codewort bestimmt. Zeigt das Syndrom an, daß durch die Korrektur kein gültiges Codewort entstanden ist, wird das entstandene Codewort verworfen.

3.2.3 RDS-Schnittstelle zum Anwendungsprogramm

Ein Anwendungsprogramm kann die RDS-Daten in standardisierter Form vom Treiber abholen, siehe dazu [1]. Über die ganz normale Lesefunktion *read* des Radio-Geräts */dev/radioX* liefert der Treiber die RDS-Daten mit jeweils drei Bytes pro RDS-Block. Das erste Byte ist das LSB der 16 Informationsbits eines Blocks, das zweite Byte enthält das MSB, im dritten Byte ist neben der Blocknummer auch das Ergebnis der Fehlerkorrektur codiert. Dabei hat das dritte Byte folgenden Aufbau:

Bit 7	Wenn dieses Bit gesetzt ist, konnte der Block nicht dekodiert werden, d.h. es sind nicht-korrigierbare Fehler vorhanden. Ist es null, ist der Block entweder fehlerfrei oder enthält nur korrigierbare Fehler.
Bit 6	Wenn dieses Bit gesetzt ist, wurden korrigierbare Fehler entdeckt (die selbstverständlich auch bereits korrigiert sind). Ist das Bit null und das Bit 7 ebenfalls, dann wurde der Block fehlerfrei empfangen.
Bit 5..3	Hier steht die Blocknummer des empfangenen RDS-Blocks. Dabei gilt folgende Zuordnung von RDS-Block zu Blocknummer: A→0, B→1, C→2, D→3, C'→4.
Bit 2..0	Hier steht die gleiche Zahl wie in den Bits 5..3. Leider ist die V4L2-Spezifikation nicht sehr klar, was in den Feldern 5..3 und 2..0 nun genau stehen soll. Im Original heißt es für die Bits 5..3: „ <i>Received Offset. Indicates the offset received by the sync system.</i> “, für die Bits 2..0 heißt es „ <i>Offset Name. Indicates the offset applied to this data.</i> “. Wer da Licht rein bringen kann, möge sich melden...

3.3 Übertragungsformat auf der seriellen Schnittstelle

Die Schnittstelle arbeitet mit 2400 8N1, ohne Handshake.

In der Übertragungsrichtung Radio→PC laufen zwei Übertragungen parallel: RDS-Rohdaten vom RDS-Demodulator, und der Austausch von Kontrollmeldungen für Frequenzwahl und Empfangsstatus. Beide Datenströme lassen sich im Treiber auf der PC-Seite einfach unterscheiden, denn die Bytes mit RDS-Daten haben immer Bit 7 auf 1 gesetzt, die gesamte andere Kommunikation läuft mit Bit7=0. Die Bits 6..0 in den RDS-Bytes entsprechen den empfangenen Rohdaten, wobei Bit 0 zuerst empfangen wurde, Bit 6 zuletzt. In der Gegenrichtung PC→Radio laufen nur Kontrollmeldungen, bei denen immer das Bit 7 gelöscht sein soll.

3.3.1 Kontrollmeldungen

Das <ESC>-Zeichen (0x1B) kennzeichnet immer den Beginn einer Kontrollmeldung. Wenn das Radio <ESC> empfängt, wird eine evtl. bereits laufende Übertragung abgebrochen und eine neue Übertragung begonnen. Der PC sollte auf <ESC> ebenso reagieren, wenn es vom Radio gesendet wird. Es sind die nachfolgend aufgeführten Kontrollmeldungen definiert.

Beim Empfang muß der PC anhand des Kennbuchstabens wissen welche und wieviele Daten folgen. Wenn er einen Kennbuchstaben nicht zuordnen kann, dann muß er einfach auf das nächste <ESC> warten und in der Zwischenzeit alle Kontrollbytes ignorieren..

- Um die Frequenz einzustellen, muß der PC folgendes senden:

<ESC>Q<a><c><d><RETURN>

wobei a...d für die gewünschte Frequenz stehen. Von den 4 Bytes a...d ist nur das untere Nibble von Bedeutung, die Bits 7...4 werden nicht beachtet. (Bit 7 sollte aber immer auf null gesetzt werden). <a> ist das MSB, <d> ist das LSB der 4-stelligen HEX-Zahl. Der Wert errechnet sich folgendermaßen:

$$a...d = \frac{[Frequenz\ in\ Hz] - 225000}{8192} \quad (1)$$

Beispiel: Frequenz 102.7 MHz \Rightarrow $a...d = 12509 = 0x30DD$ Der PC muß dann folgendes senden:

0x1B 0x51 0x33 0x30 0x3D 0x3D 0x0D

Die beliebig wählbaren Bits 7...4 wurden hier zu '3' gewählt, weil das der Wert ist, den auch das Radio bei der Rückmeldung einsetzt. Das Radio antwortet mit dem fast gleichen Paket, es wird allerdings kein <RETURN> gesendet. Also sieht die Kontrollmeldung des Radios so aus:

0x1B 0x51 0x33 0x30 0x3D 0x3D

Das Radio kann diese Meldung nicht nur als Reaktion auf eine Frequenzeinstellung durch den PC senden, sondern sendet sie auch dann, wenn sich die Frequenz des Radios aus anderen Gründen geändert hat. Das kann in zwei Fällen passieren: Einmal, wenn man die Suchlauf-Taste drückt, das andere mal, wenn man den Vorzugssender über die Speichertaste abrufen.

- Um den Empfangsstatus abzufragen, muß der PC folgendes senden:
<ESC>R

Die Antwort des FiFi-Radios sieht so aus:

<ESC>R<a><c><d>

wobei von den vier empfangenen Bytes a...d nur das jeweils untere Nibble von Bedeutung ist, und die oberen Bits 7...4 ignoriert werden können (Sie sind vom Radio fest auf '3' gesetzt). Die 16 Nutzbits sind wie folgt zu interpretieren:

<a>	Empfangsfeldstärke. Kann die Werte 0...15 annehmen.
	Keine Bedeutung. Diese Bits können ignoriert werden.
<c>	Im obersten Bit (d.h. Bit 3!) steht das Stereo-Bit. Ist es 1, wird gerade in Stereo empfangen. Die unteren drei Bits enthalten die oberen drei Bits der Mittenfrequenz. Siehe Wort d.
<d>	Enthält die unteren vier Bits der Mittenfrequenz. Zusammen mit den drei Bits aus Wort <c> ergibt das ein 7-Bit-Wort, das die Werte 0...127 annehmen kann. Ein Wert von ca. 55 entspricht einer Abstimmung auf Sendermitte.

- Um den Audio-Ausgang des Radio-Bausteins TEA5768 ein- und auszuschalten („Muting“), muß der PC folgendes senden:
<ESC>M → schaltet den Audio-Ausgang stumm.
<ESC>N → schaltet den Audio-Ausgang aktiv.
Diese Funktion ist bis einschließlich zur Firmware V0.6 auf dem FiFi-Radio nicht implementiert.

4 Sonstiges

Dieses Kapitel enthält weitere Hinweise oder Erfahrungsberichte

4.1 Akkus vs. Batterien

Der RDS-Demodulator auf dem FiFi-Radio benötigt mindestens 3.6V für eine einwandfreie Funktion. Wenn das Radio mit 3 Akkus anstelle von Batterien betrieben wird, wird diese Spannung möglicherweise unterschritten, weil auch noch ein Spannungsabfall am Schalttransistor auf dem Radio hinzu kommt. Eine Funktion des RDS-Teils ist dann nicht mehr gewährleistet. Ein einwandfreier RDS-Empfang ist nur möglich bei Batteriebetrieb.

Die Radio-Funktion selbst ist allerdings auch noch bei weniger als 3V möglich, d.h. die Batterien/Akkus können wirklich bis zum allerletzten Rest ausgelutscht werden!

4.2 Anzeige der RDS-Daten mit dem Tool „karamba“

Mit dem Tool *karamba* kann man Texte oder Bilder transparent auf dem Desktophintergrund anzeigen⁵. Was und wo angezeigt werden soll, legt man in einer Textdatei („Theme“) fest, die die Endung *.theme* haben muß. Der Trick für die Anzeige der RDS-Daten mit diesem Tool besteht darin, das Kommandozeilen-Tool *rdsquery* für die Abfrage des gewünschten Texts zu

⁵Die Icons werden leider nicht transparent behandelt, also muß man einen freien Platz auf dem Desktop verwenden.

verwenden, und aus der Ausgabe dieses Programms die gewünschte Information mit dem Tool *awk* herauszufiltern.

Um das schöne Ergebnis aus Abbildung 1 zu erhalten, braucht man eine *.theme*-Datei mit folgendem Inhalt:

```
# RDS-Daten vom FiFi-Radio anzeigen
karamba x=770 y=0 w=500 h=95 ontop=false locked=true interval=1000

<group> x=0 y=0
  #Bild im Hintergrund anzeigen
  image x=0 y=0 path="bground.png"

  #Abfrage des Sendernamens mit rdsquery
  text x=5 y=-5 sensor=program program="rdsquery -s localhost -c 1 -t pname
    | awk '{sub(/^pname:/,'x'); print}'" align=left
    fontsize=48 font="urw gothic l" color=248,224,196 interval=2000

  #Abfrage des Infotexts mit rdsquery
  text x=8 y=65 sensor=program program="rdsquery -s localhost -c 1 -t lrtxt
    | awk '{sub(/^lrtxt:/,'x'); print}'" align=left
    fontsize=16 font="bitstream vera sans" color=254,145,89 interval=1000
</group>
```

4.3 Was man beim Anschluß an den PC beachten sollte

Der Prozessor auf dem FiFi-Radio ist mit seinen Port-Pins direkt an den Stecker für die serielle Schnittstelle angeschlossen. Die Treiberleistung eines solchen Port-Pins ist eigentlich nicht geeignet, um damit eine V24-Schnittstelle zu betreiben. Außerdem ist der erzeugte Spannungspegel, der ja auch im Idealfall nur zwischen 0 V und der Batteriespannung liegen kann, nicht normgerecht: Die V24-Norm verlangt, daß der Bereich -3 V bis +3 V nicht verwendet wird. In der Realität sind allerdings die meisten V24-Pegelwandler durchaus in der Lage, mit den Pegeln des FiFi-Radios klar zu kommen. Mit etwas Pech kann es aber sein, daß der Pegelwandler im PC den Empfang verweigert. Dann hilft nur ein Versuch mit einem USB-Seriell-Wandler.

Falls durch ein besonders langes Kabel zwischen FiFi-Radio und PC noch eine hohe kapazitive Last hinzu kommt, dann kann es sein, daß dadurch eine saubere Kommunikation nicht mehr möglich ist. Es sollte also ein möglichst kurzes Kabel benutzt werden. Am besten geeignet sind die gängigen USB-Seriell-Wandler, denn die kann man direkt auf's FiFi-Radio aufstecken und vermeidet damit die schädlichen Kabeleinflüsse.

Falls beim Anschluß der seriellen Schnittstelle der Radioempfang deutlich gestört wird, kann man versuchen, das serielle Kabel (oder die USB-Zuleitung eines Wandlers) mit einem Ferritkern mundtot zu machen. Geeignete Ferrite zum Aufschnappen auf Kabel verschiedener Größe gibt's bei *Reichelt*⁶. Auch der USB-Seriell-Wandler selbst kann stören. Ein serielles Verlängerungskabel zum Absetzen des Wandlers vom FiFi-Radio kann helfen.

⁶<http://www.reichelt.de>, Suche nach *Ferritring*

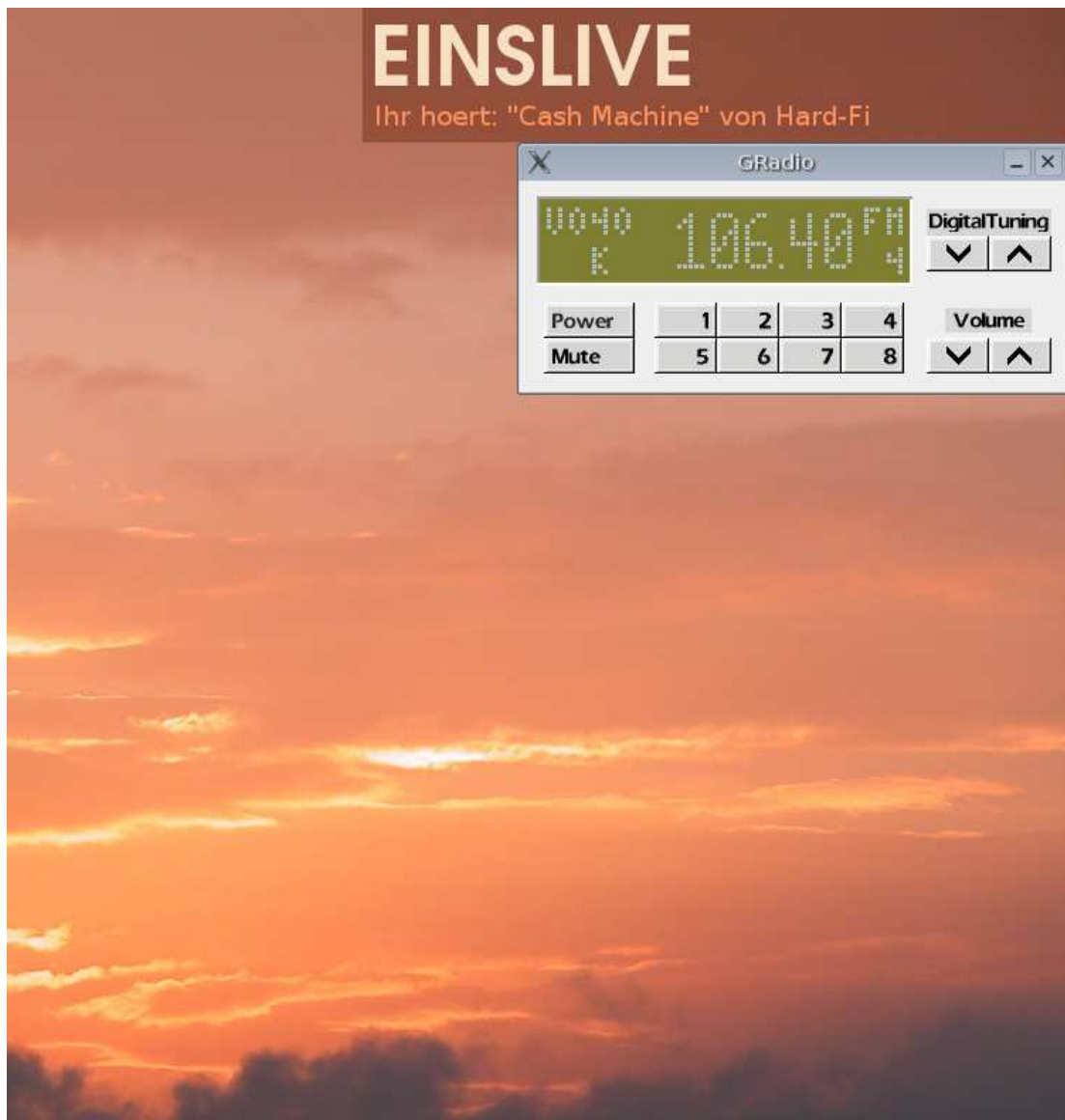


Abbildung 1: RDS-Anzeige mit Karamba

Inhaltsverzeichnis

1	Installation	1
1.1	Voraussetzungen	2
1.2	Kerneltreiber	2
1.2.1	Modul-Parameter	3
1.3	Usermode-Programm fifiattach	4
2	Anwendungsprogramme	5
2.1	Radio-Anwendung GRadio	5
2.2	Radio-Anwendung Gnomeradio	6
2.3	RDS-Dämon rdsd	6
2.3.1	Installation	7
2.3.2	Betrieb	8
3	Funktionsprinzip	8
3.1	tty und die Line Discipline	8
3.2	RDS-Synchronisation und -Fehlerkorrektur	9
3.2.1	Blocksynchrisation	10
3.2.2	Fehlerkorrektur	10
3.2.3	RDS-Schnittstelle zum Anwendungsprogramm	10
3.3	Übertragungsformat auf der seriellen Schnittstelle	11
3.3.1	Kontrollmeldungen	11
4	Sonstiges	12
4.1	Akkus vs. Batterien	12
4.2	Anzeige der RDS-Daten mit dem Tool „karamba“	12
4.3	Was man beim Anschluß an den PC beachten sollte	13

Literatur

- [1] M. H. Schimek, B. Dirks, H. Verkuil, „Video for Linux Two API Specification“, Draft 0.9, 2005
- [2] A. Rubini, J. Corbet, „Linux Device Drivers“, Second Edition, ISBN 0-596-00008-1, O’Reilly, June 2005
- [3] J. Quade, E-K. Kunst, „Linux-Treiber entwickeln“, 1. Auflage, ISBN 3-89864-238-0, dpunkt-Verlag, 2004
- [4] H. Rohling, „Einführung in die Informations- und Codierungstheorie“, 1. Auflage, ISBN 3-519-06174-0, Teubner, Stuttgart, 1995
- [5] National Radio Systems Committee, „United States RBDS Standard“, National Association of Broadcasters, April 1998