Diplomarbeit

Fachhochschule Münster University of Applied Sciences

Fachbereich Elektrotechnik und Informatik

Entwicklung und Realisierung eines PDA-Testers zur Analyse von IR-Fernbedienungen

von
Thorsten Krautstrunk

Referent:	Prof. Dr.-Ing. Peter Richert
Korreferent:	Prof. Dr.-Ing. Dirk Fischer
Kolloquium:	22. April 2003
Durchgeführt bei:	Fachhochschule Münster, Fachbereich Elektrotechnik und Informatik, eLKaTe — Labor Kommunikationstechnik Stegerwaldstraße 39, 48 565 Steinfurt

Kurzfassung

Der Schwerpunkt dieser Diplomarbeit ist die Realisierung eines Testgerätes für Infrarot-Fernbedienungen. Dieses Gerät überprüft die Funktionen der Fernbedienung als solche. Darüber hinaus wird das Infrarot-Signal als Puls- und Pausenlänge grafisch angezeigt. So können verbreitete Fernbedienungs-Protokolle unterschieden werden und es ist zu erkennen, ob auch fehlerfrei übertragen wurde. Somit ist es beispielsweise möglich zu überprüfen, ob bei der Übertragung ein Bit fehlt.

Außerdem werden Programme zur Dekodierung der empfangenen Infrarot-Signale vorgestellt und eingesetzt, damit die gedrückte Taste einer Fernbedienung erkannt und am Display dargestellt werden kann.

Ein weiterer Schwerpunkt ist die Beschreibung geeigneter Hard- und Software. Die interne IrDA-Schnittstelle des Sharp Zaurus SL-5500G wird genutzt, um infrarotes Licht, welches für das menschliche Auge nicht sichtbar ist, auf dem Display des Linux-PDA's darzustellen. Diese grafische Ausgabe wird mit den Qt/Embedded-Bibliotheken entwickelt. Darüber hinaus wird beschrieben, wie die Entwicklungsumgebung für das Embedded-Linux System des Zaurus unter SuSE 8.0 eingerichtet wird. Mit dieser Entwicklungsumgebung ist es dann möglich, Softwareentwicklungen unter X11 zu testen. Diese Software kann anschließend, nachdem sie für den PDA kompiliert wurde, auf den Sharp Zaurus SL-5500G kopiert werden.

Vorwort

Hiermit danke ich allen, die mich bei der Ausarbeitung dieser Diplomarbeit unterstützt und mit Rat und Tat zur Seite gestanden haben. Besonderen Dank gilt:

- Herrn Prof. Dr.-Ing. Peter Richert für die Bereitstellung des Diplomthemas und die freundliche und fachliche Betreuung bei der Realisierung,
- Herrn Prof. Dr.-Ing. Dirk Fischer für die Übernahme des Korreferates,
- Herrn Dipl.-Ing. Peter Furchert für seine stets freundliche Unterstützung,
- Eltern und Familie, ohne deren Unterstützung ein Studium kaum möglich gewesen wäre.

Erklärung

Hiermit wird versichert, dass diese Diplomarbeit selbstständig angefertigt wurde und die benutzten Hilfsmittel angegeben wurden.

Steinfurt, April 2003

Thorsten Krautstrunk

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Nahezu jedes Gerät der Unterhaltungselektronik wird mit seiner eigenen Fernbedienung angeboten. Aus Gründen des Patentrechts und der technischen Unabhängigkeit haben die meisten Hersteller ihre eigenen Protokolle zur Infrarot-Datenübertragung entwickelt. Damit diese Protokolle analysiert, bzw. die Funktionen der Fernbedienungen überprüft werden können, werden besondere Meßgeräte benötigt, da das infrarote Licht im nicht sichtbaren Bereich liegt.

In dieser Diplomarbeit soll gezeigt werden, dass so ein Meßgerät mit einem PDA [1] realisiert werden kann und noch einige Funktionen zusätzlich möglich sind. Die üblichen Aufgaben des PDA's, wie Adressbuch, Terminplaner, Taschenrechner, Video- und mp3-Player bleiben erhalten. In Kapitel 2 werden verschiedene PDA's vorgestellt und verglichen.

Wird das Signal einer Infrarot-Fernbedienung an der internen IR-Schnittstelle [2] des PDA's empfangen, soll es möglich sein, dieses grafisch auf dem Display darzustellen. Wird dieses Signal analysiert, kann das verwendete Protokoll des Herstellers erkannt und die jeweilige Bitfolge abgelesen werden. Mit der Bitfolge ist es möglich, sowohl die gedrückte Taste als auch das dazugehörige Gerät herauszufinden. Zusätzlich soll erkannt werden, ob die Übertragung fehlerfrei ist. In Kapitel 3 werden verbreitete Protokolle vorgestellt.

In Kapitel 4 werden einige relevante Programme und Kernelmodule aus dem LIRC-Projekt beschrieben. Das Programm "irw" beispielsweise erkennt automatisch die Tasten und Geräte verschiedener Fernbedienungen, wenn diese am PC mit Hilfe von "irrecord" angelernt wurden. Die dazu notwendige Schaltung des IR-Empfängers wird ebenfalls erklärt. Mit dem Programm "rc" und einer Senderschaltung ist es möglich, Infrarotsignale auszusenden.

Die erforderliche Hardware, mit der IR-Signal mit einem PC empfangen bzw. gesendet werden kann, ist in Kapitel 5 vorgestellt und beschrieben.

Da Software für Sharp Zaurus SL-5500G entwickelt wurde, wird in Kapitel 6 erklärt, wie auf einem SuSE-Linux-PC die Entwicklungsumgebung installiert und eingerichtet wird, damit die Programme auf dem PC entwickelt und kompiliert werden können. Diese Umgebung simuliert eine ARM-Architektur, da der StrongARM SA-1100 Prozessor des Zaurus im Gegensatz zum x86-Prozessor des PC's eine spezielle Architektur besitzt. Die kompilierten Dateien werden auf den Sharp PDA kopiert und sind

[1] Personal Digital Assistant
[2] Infrarot

sofort ausführbar. Auch die Systemänderungen im PC, die notwendig sind, um den Sharp Zaurus mit einem Linux-PC zu synchronisieren, werden in diesem Kapitel erklärt. Dies ist erforderlich, da die mitgelieferte Synchronisations-, Datenssicherungs- und -bearbeitungssoftware des Linux-PDA's nur mit Microsoft Windows Betriebssystemen zusammenarbeitet. Zusätzlich wird die Erstellung eines selbst konfigurierten und kompilierten Kernels für den SL-5500G beschrieben.

Im Rahmen dieser Diplomarbeit ist das Programm "zmode2" entwickelt worden. In Kapitel 7 wird der Aufbau des Programms erklärt. Außerdem werden Details erläutert, die bei der Qt/Embedded-Programmierung zwingend zu beachten sind.

2 Voraussetzung und Ziele

2.1 Allgemeines

In Werkstätten gibt es spezielle Meßgeräte, die ausschließlich anzeigen, ob eine Fernbedienung infrarotes Licht aussendet. Ebenfalls ist es möglich, das Infrarot-Signal mit Hilfe einer Fotodiode auf einem Oszilloskop anzuschauen. Allerdings ist auch diese Möglichkeit nicht sehr komfortabel, da ein Oszilloskop aufgrund der Abmessungen und des Gewichts nur schwer beim Kunden eingesetzt werden kann.

In dieser Arbeit ist ein mobiles Testgerät für Infrarot-Fernbedienungen realisiert worden. Das zu entwickelnde Meßgerät soll folgende Spezifikation erfüllen:

- geringe Größe und Gewicht
- integrierte grafische Signalanzeige
- Erkennung unterschiedlicher Fernbedienungen
- Kontrolle von IR-Empfängern durch Senden von IR-Signalen

Daher bietet sich ein PDA bei der Realisierung all dieser Anforderungen an. Wenn die interne IR-Schnittstelle verwendet werden kann, ist keine zusätzliche Hardware erforderlich.

2.2 Auswahl des Betriebssystems

Infrarot-Fernbedienungen werden, falls spezielle Treiber und Programme eingesetzt werden, unter den Betriebssystemen Linux und Microsoft Windows 'verstanden'. Dies bedeutet, der Computer läßt sich mit Hilfe einer IR-Fernbedienung steuern. So kann z.B. der mp3-Player gestartet, die Lautstärke verändert oder der nächste Musiktitel gewählt werden.

Unter Linux ist dieses mit der LIRC [1] Software möglich. Dabei werden die Linux-Treiber, die in den Kernelmodulen implementiert sind, zur Unterstützung der Hardware genutzt. Fernbedienungen lassen sich mit einem speziellen Programm anlernen.

[1] Linux Infrared Remote Control [LP02]

Ebenso ist es möglich, empfangene IR-Signale grafisch darzustellen oder IR-Signale zu senden.

Linux ist ein freies Unix-System und unterliegt der GNU GPL [2][Fre91]. Kurz zusammengefaßt sichert diese Lizenz, dass ein frei zur Verfügung gestellter Quellcode auch immer und für jedermann frei bleibt. Sie erlaubt es, die Software zu ändern oder Teile davon in neuen *freien* Programmen zu verwenden. Mit dem Medium Internet kombiniert, kann eine schnelle und globale Verbreitung der offenen Quellen und hierdurch eine nahezu synchrone Weiterentwicklung erfolgen. Das Betriebssystem Linux ist ein Musterbeispiel für diese Philosophie [JK99]. So werden auch Entwicklungstools weiterentwickelt und sind in der Regel für Linux *frei*. Auch LIRC unterliegt der GNU GPL und die Entwicklungen dieser Diplomarbeit sollen ebenfalls frei sein. So darf der Quellcode verwendet und weiterentwickelt werden.

Für die Windows Betriebssysteme wurde die LIRC-Software portiert und ist als winLIRC [wP01] erhältlich. winLIRC hat mit gewissen Einschränkungen die selben Funktionen, als Grundlage dient allerdings eine ältere Version von LIRC. Das Programm setzt eine deutlich leistungsfähigere Hardware voraus als das Linux-Pendant. Der 'Lern'-Algorithmus zum Anlernen von neuen Fernbedienungen ist weit von dem Stand der Linux-Version entfernt [Bar00]. Außerdem ergaben sich große Schwierigkeiten beim Testen von winLIRC unter dem Betriebssystem Windows 98. Ebenfalls sind gute, aktuelle Entwicklungstools unter Windows oft nicht kostenlos erhältlich.

Auf Grund der Vorteile die Linux und LIRC im Gegensatz zu Windows und winLIRC bieten, soll diese Diplomarbeit unter dem Betriebssystem Linux realisiert werden.

2.3 Auswahl des PDA's

Da sich Linux als geeignetes Betriebssystem herausstellte, muß ein PDA gewählt werden, der unter diesem Betriebssystem arbeitet. Es sind folgende Linux-PDA's verfügbar:

- Gmate Yopy
- Palm
- Compaq iPaq
- Sharp Zaurus

Für die weitere Auswahl sind die folgenden Kriterien ausschlaggebend:

- Preis
- Systemausstattung

[2] General Public License

- verfügbare Entwicklungsumgebung
- Dokumentation im Internet
- Forum mit Entwicklern und Benutzern

2.3.1 Compaq iPaq und Palm

Da die verschiedenen Compaq iPaq und Palm PDA's nicht mit dem Betriebssystem Linux ausgeliefert werden, muß dieses durch den User gewechselt werden. Durch diesen Wechsel erlischt die Herstellergarantie.

Daher werden im Folgenden nur der Gmate Yopy und der Sharp Zaurus näher vorgestellt und berücksichtigt.

2.3.2 Gmate Yopy YP-3000

Der Linux-PDA Yopy wurde zum ersten Mal auf der CeBIT 2000 von Samsung präsentiert. Samsung zog sich später aus dem Vertrieb des Yopy zurück und 2002 stellte Gmate [Gma00] den überarbeiteten Yopy auf der CeBIT vor. Gekauft werden kann der Gmate Yopy YP-3000 in Deutschland über den österreichischen Distributor Xact [Xac02]. Dieser PDA besitzt die folgenden Merkmale [FD02]:

- Prozessor: Intel StrongARM (SA-1100, 206 MHz)
- Speicher: 64 MByte RAM, 16 MByte Flash
- Display: TFT, 3,6 Zoll, 240 x 320 Pixel, 16 Bit Farbtiefe, OHNE Hintergrundbeleuchtung
- Anschlüsse: Cradle, IrDA, SD/MMC-Card, Kopfhörer, USB, serieller Port
- Tastatur: Geräte- (QWERTY) und Bildschirmtastatur, Schrifterkennung
- grafische Oberfläche: X11, Desktop Gnome
- Applikation: Kalender, Terminplaner, Aufgaben und Kontaktdatenbank
- Entwicklerplattform [Yop03]
- Synchronisationssoftware für Microsoft Windows
- Preis: 600 Euro (Oktober 2002)

2.3.3 Sharp Zaurus SL-5500G

Der Sharp Zaurus SL-5500G wird ebenfalls mit Linux ausgeliefert. Er wird in Deutschland durch mehrere Kaufhausketten und Internet-Shops vertrieben. Dieser PDA besitzt die folgenden Merkmale [Fae02]:

- Prozessor: Intel StrongARM (SA-1100, 206 MHz)
- Speicher: 64 MByte RAM, 16 MByte Flash
- Display: TFT, 3,5 Zoll, 240 x 320 Pixel, 16 Bit Farbtiefe, Hintergrundbeleuchtung
- Anschlüsse: Cradle, IrDA, SD/MMC-Card, CF II, Kopfhörer, USB, serieller Port
- Tastatur: Geräte- (QWERTZ) und Bildschirmtastatur, Schrifterkennung
- grafische Oberfläche: Qt/Embedded, Qtopia
- Applikation: Kalender, Terminplaner, Aufgaben und Kontaktdatenbank
- Entwicklerplattform [Zau03a], [Zau03b]
- Synchronisationssoftware für Microsoft Windows
- Preis: 400 Euro (Oktober 2002) / Entwicklerpreis: 336 Euro

2.3.4 Entscheidung

Die wichtigen technischen Merkmale sind bei beiden Linux-PDA's vergleichbar. Allerdings unterscheiden sie sich durch folgende Details, die abschließend kaufentscheidend waren.

Für den **Yopy YP-3000** spricht, dass für die grafische Oberfläche X11 und für den Desktop Gnome verwendet wird. Dehalb sollten auf diesem System auch die X-Anwendungen von LIRC funktionieren. Allerdings ist er ausschließlich über den österreichischen Distributor erhältlich. Ein großer Nachteil ist die fehlende Hintergrundbeleuchtung, da nur bei optimaler Raumausleuchtung das Farbdisplay ablesbar ist.

Für den **Sharp Zaurus SL-5500G** sprechen die folgenden Gründe. Im Internet sind mehrere Entwicklerplattformen und Foren vorhanden. Er besitzt eine Hintergrundbeleuchtung, so dass auch bei nicht optimalen Lichtverhältnissen das Farbdisplay ablesbar ist. Der PDA kann zum günstigeren Entwicklerpreis bestellt werden. Dann wird auch direkt eine Entwickler-CD für SuSE 8.0 mitgeliefert. Allerdings verwendet der Zaurus für die grafische Oberfläche Qtopia bzw. Qt/Embedded. Deshalb ist eine Softwareentwicklung für die grafische Ausgabe des empfangenen IR-Signals notwendig.

Der Sharp Zaurus SL-5500G stellt sich beim Vergleich als optimaler PDA für diese Diplomarbeit heraus.

3 Infrarot-Fernbedienungen

3.1 Ziel

In diesem Kapitel soll die infrarote Strahlung, die verwendeten Protokolle und die Modulationsfrequenz vorgestellt bzw. erklärt werden. Mit diesem Wissen ist es dann möglich bei einer grafischen Ausgabe des IR-Signals festzustellen, ob das übertragene Protokoll fehlerfrei ist. Außerdem ist es möglich aus der Bitfolge den gesendeten Befehl zu dekodieren.

3.2 Infrarote Strahlung

Der Infrarotbereich ist nach DIN 5031 dem Bereich der optischen Strahlung zugeordnet, der auch die sichtbare und ultraviolette Strahlung umfaßt. Dabei handelt es sich um elektromagnetische Strahlung im Spektralbereich von 780 nm bis 1 mm. Hierbei wird zwischen dem nahen, dem mittleren und dem fernen IR-Bereich unterschieden [Jan02].

Wellenlänge	optische Strahlung
100 nm ... 380 nm	ultravioletter Bereich
380 nm ... 780 nm	sichtbarer Bereich
780 nm ... 1,4 μm	naher infraroter Bereich
1,4 μm ... 3,0 μm	mittlerer infraroter Bereich
3,0 μm ... 1 mm	ferner infraroter Bereich

Tabelle 3.1: Optische Strahlung

Die Infrarot-Sendedioden strahlen mit einer Wellenlänge von ca. 900 nm. Das emittierte Licht wird dabei gebündelt. Allerdings muß die Signalübertragung bei Fernbedienungen nicht direkt vom Sender zum Empfänger erfolgen, sondern in geschlossenen Räumen kann das Licht auch von Wänden reflektiert werden.

3.3 Modulationsfrequenz des Infrarot Signals

Die meisten Infrarot-Fernbedienungen arbeiten im Frequenzbereich zwischen 30 und 40 kHz. Die Fernbedienungen müssen auch bei hohen Störimpulsen, wie z.B. Lampen, Sonneneinstrahlung noch einwandfrei funktionieren. Um dies zu erreichen, wird das Signal bei Highpegel moduliert. Zur Überprüfung der Modulationsfrequenz wird eine schnelle Fotodiode benötigt. So ist zu erkennen, dass beim Highpegel eines Bits ein Signal mit einer Sinusschwingung von z.B. 36 kHz gesendet wird. Beim Lowpegel wird kein Signal gesendet. In Abbildung 3.3.1 ist das gesendete Signal eines Bits des RC5-Codes dargestellt.

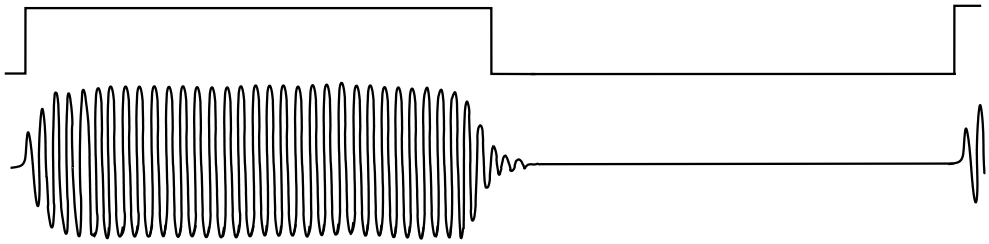


Abbildung 3.3.1: Modulationsfrequenz

3.4 Fernbedienungs-Protokolle

Nahezu jedes Gerät in der Unterhaltungselektronik hat seine eigene Fernbedienung. Viele Hersteller nutzen aus Gründen des Patentrechts und der technischen Unabhängigkeit ein eigenes Protokoll. Bei der großen Anzahl der Fernbedienungen ist es nicht möglich, alle Codes darzustellen. In Tabelle 3.2 werden verbreitete Protokolle vorgestellt. Die Zuordnung der Hersteller muss nicht auf alle Geräte zutreffen [Woh02].

Verwendetes Protokoll	Hersteller
RC5	Philips, Loewe, Grundig, Marantz
NEC	Yamaha, Canon, Tevion, Harman/Kardon
DENON	Denon
SIRCS	Sony
RECS80	Thomson, Nordmende, Telefunken, Saba
MOTOROLA	Grundig, Kathrein
JAPAN	Panasonic, Loewe
SAMSUNG	Samsung
FERNOST	Daewoo, Wisi

Tabelle 3.2: Verwendete Protokolle der Hersteller

3.4.1 Allgemeines

In den folgenden Kapiteln werden die verschiedenen Infrarot-Datenübertragungs-Protokolle vorgestellt. Als Quelle diente [Woh02], allerdings sind die Protokolle sowohl mit einem Oszilloskop, als auch mit Programmen aus dem LIRC-Projekt überprüft worden. Die dargestellten Abbildungen und die Bitfolgen in den Tabellen sind Befehle von vorhandendenen Fernbedienungen. Im Kapitel 3.4.9 sind Protokolle vorgestellt, von denen keine Fernbedienungen vorhanden sind.

Die Fernbedienungs-Protokolle unterscheiden sich im wesentlichen durch:

- Anzahl der Bits
- Start-Bit
- Stop-Bit
- Toggle-Bit
- Kodierung
- Dauer der Übertragung
- Modulationsfrequenz

3.4.2 SIRCS- bzw. CNTRL-S-Code

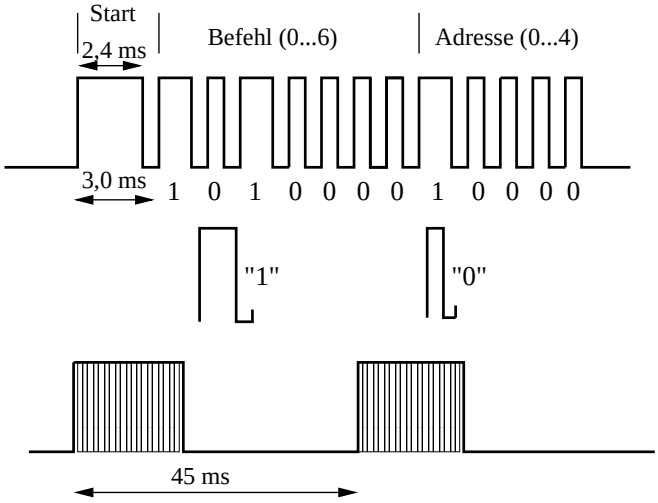


Abbildung 3.4.1: Protokollaufbau SIRCS

Das SIRCS- bzw. CNTRL-S-Protokoll wird von Sony eingesetzt. Es beginnt mit einem Start-Bit, darauf folgen sieben Befehls-Bits für die Tasten und endet mit fünf Adress-Bits. Das Start-Bit hat eine Länge von 2,4 ms, darauf folgt eine Pause von

0,6 ms. Im Anschluß werden die Daten gesendet, wobei eine logische 1 durch 1,2 ms High- 0,6 ms Lowpegel und eine logische 0 durch 0,6 ms High- 0,6 ms Lowpegel definiert werden [Woh02]. Diese Kodierungsart wird als Pulsweitenmodulation bezeichnet [Fis02], allerdings mit variabler 'Abtastzeit'. Bei der Auswertung des Codes ist zu beachten, dass das Bit 0 des Befehls- bzw. Adressfeldes zuerst, Bit 6 bzw. Bit 4 als letztes gesendet wird.

Das Datenwort wird alle 45 ms gesendet. Es muß allerdings mindestens zweimal wiederholt werden, ansonsten wird von einem Übertragungsfehler ausgegangen. Der SIRCS-Code und der CTRL-S-Code verwenden den selben Protokollaufbau. CTRL-S wird für die drahtgebundene Steuerung, z.B. zur Kopplung von Videorecordern benutzt. SIRCS wird für die Infrarot-Datenübertragung verwendet und ist mit 40 kHz moduliert.

In Abbildung 3.4.1 ist der Code der Sony Fernbedienung RM-816 beim Drücken der Taste 6 dargestellt. Man erhält eine binäre Folge von 1010000 10000, welches Befehl fünf und Gerät eins bedeutet.

Taste	Bitfolge
1	0000000 10000
2	1000000 10000
3	0100000 10000
4	1100000 10000
5	0010000 10000
6	1010000 10000
7	0110000 10000

Tabelle 3.3: Bitfolgen der Sony Fernbedienung RM-816 (SIRCS-Code)

In Tabelle 3.3 sind die gesendeten Bitfolgen der Sony Fernbedienung RM-816 aufgelistet. Der Zahlenwert der Adresse-Bits ist als TV definiert. In Tabelle 9.3 auf Seite 58 (Anhang) sind die Codes der Systemadressen von verschiedenen Geräten, in Tabelle 9.4 ist ein Auszug aus den gemeinsamen Befehlen aufgelistet.

Weitere Informationen über den SIRCS- bzw. CNTRL-S-Code sind im Buch [Woh02] zu finden.

3.4.3 RC5-Code

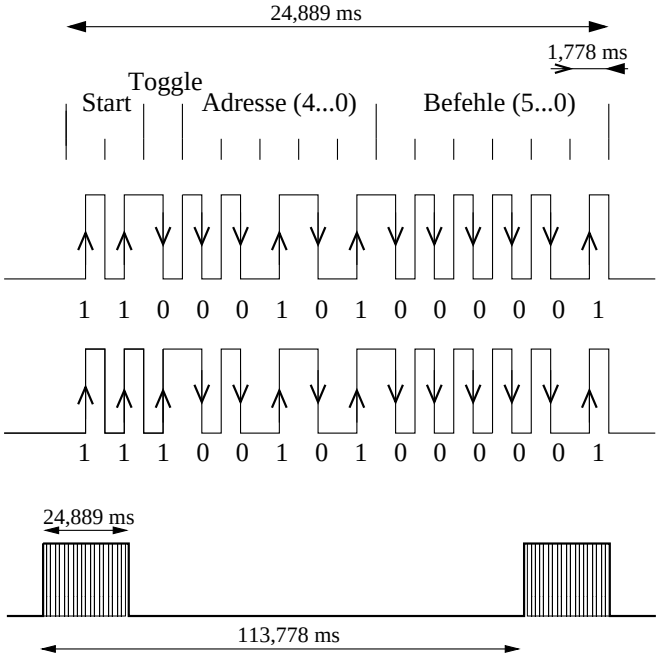


Abbildung 3.4.2: Protokollaufbau RC5-Code

Ein weit verbreiteter Standard in Europa ist der von der Firma Philips entwickelte RC5-Code. Jede Übertragung beginnt mit zwei Start-Bits, gefolgt vom Toggle-Bit. Im Anschluß werden fünf Bits für die Adresse übertragen. Das Datenwort endet mit sechs Bits für den Befehl. Ein Datenwort besteht demzufolge aus 14 Bits.

Die Start-Bits dienen zum Einstellen des AGC-Pegels [1] im Empfänger-IC. Das Toggle-Bit wechselt bei jedem neuen Tastendruck den Wert, um zwischen einem erneuten oder dauerhaften Druck der gleichen Taste unterscheiden zu können. Zwischen 32 verschiedenen Geräten kann mit Hilfe der Systemadressen-Bits unterscheiden werden ($2^5 = 32$). Mit den sechs Befehls-Bits ist jeder Taste ein eindeutiges Signal zugeordnet ($2^6 = 64$). In Tabelle 9.1 und 9.2 auf Seite 56 (Anhang) ist die Zuordnung der einzelnen Geräte und Befehle nachzulesen.

Beim RC5-Code werden die Befehle biphase kodiert, das bedeutet, ein Bit wird aus zwei alternierenden Halbbits zusammengesetzt. Die steigende Flanke ist also eine logische 1, die fallende Flanke eine logische 0. Das Signal ist mit 36 kHz moduliert.

In Abbildung 3.4.2 ist das Datenwort der Fernbedienung des Philips Videorecorder VR 675 dargestellt, wenn die Taste 1 betätigt wird. Nach nochmaligem Drücken der Taste 1 ändert sich das Toggle-Bit und das unten gezeigte Datenwort wird gesendet. Man erhält als binäre Folge 11000101000001 bzw. 11100101000001.

[1] Auto Gain Control

In der Spezifikation wurde festgelegt, dass die Übertragung eines Bits 1,788 ms dauert. Daraus folgt, dass die Übertragungszeit der 14 Bits eines Datenworts 24.889 ms anhält. Solange eine Taste gedrückt gehalten bleibt, wird das Signal alle 113,778 ms wiederholt [Woh02].

Taste	Bitfolge	Taste	Bitfolge
0	11000101000000	-	11000101100001
1	11000101000001	ZURÜCK	11000101100101
2	11000101000010	VOR	11000101100110
3	11000101000011	STILL	11000101101001
4	11000101000100	CLEAR	11000101110001
5	11000101000101	PLAY	11000101110101
6	11000101000110	STOP	11000101110110
7	11000101000111	RECORD	11000101110111
8	11000101001000	SP/LP	11000101111010
9	11000101001001		
SELECT	11000101001011	SYSTEM	10000101001010
STANDBY	11000101001100	OK	10000101010111
TIMER	11000101011101	INDEX	10000101110000
+	11000101100000	MONITOR	10000101111010

Tabelle 3.4: Bitfolgen des Philips VR 675 (RC5-Code)

In Tabelle 3.4 sind alle Tasten der Fernbedienung des Philips Videorecorder VR 675 und deren Bitfolgen aufgeführt. Die Bitfolgen beginnen mit den beiden Start-Bits. Das Toggle-Bit hat in dieser Tabelle immer den Wert 0. Im Anschluß folgen die fünf Adress-Bits, wobei der Zahlenwert VCR1 definiert (vgl. Tabelle 9.1 auf Seite 56).

3.4.4 DENON-Code

Der DENON-Code beginnt mit fünf Adress-Bits, darauf folgen zehn Befehls-Bits und endet mit einem Stop-Bit. Der Code wird mit einer Modulationsfrequenz von 32 MHz gesendet und hat kein Start-Bit.

Eine logische 1 ist durch 275 μs High- 1900 μs Lowpegel und die logische 0 durch 275 μs High- 775 μs Lowpegel definiert. Die Bits unterscheiden sich also nur durch ihr Pausenverhältnis. Dies wird in dieser Diplomarbeit als Pulsabstandsmodulation bezeichnet. Die 16 Bits werden nach 65 ms noch einmal wiederholt, allerdings werden dann die Befehls-Bits invertiert übertragen.

In Abbildung 3.4.3 ist das Datenwort der Taste 4 einer Denon CD-Player Fernbedienung dargestellt. Man erkennt die Bitfolge 00010 1010001000 1, beim Wiederholen 00010 0101110111 1.

In Tabelle 3.5 sind die gesendeten Bitfolgen der Tasten 1 bis 9 aufgelistet. In der dritten Spalte ist zu erkennen, dass die Befehls-Bits beim Wiederholen invertiert sind.

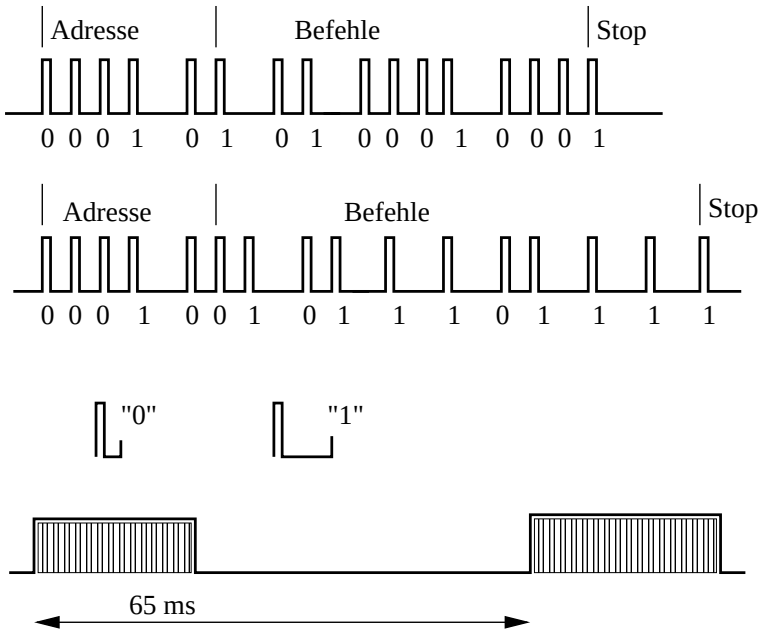


Abbildung 3.4.3: Protokollaufbau DENON-Code

Taste	gesendete Bitfolge	Wiederholung
1	00010 0100001000 1	00010 1011110111 1
2	00010 1100001000 1	00010 0011110111 1
3	00010 0010001000 1	00010 1101110111 1
4	00010 1010001000 1	00010 0101110111 1
5	00010 0110001000 1	00010 1001110111 1
6	00010 1110001000 1	00010 0001110111 1
7	00010 0001001000 1	00010 1110110111 1
8	00010 1001001000 1	00010 1001001000 1
9	00010 0101001000 1	00010 1010110111 1

Tabelle 3.5: Bitfolgen des Denon CD Players (DENON-Code)

3.4.5 NEC-Code

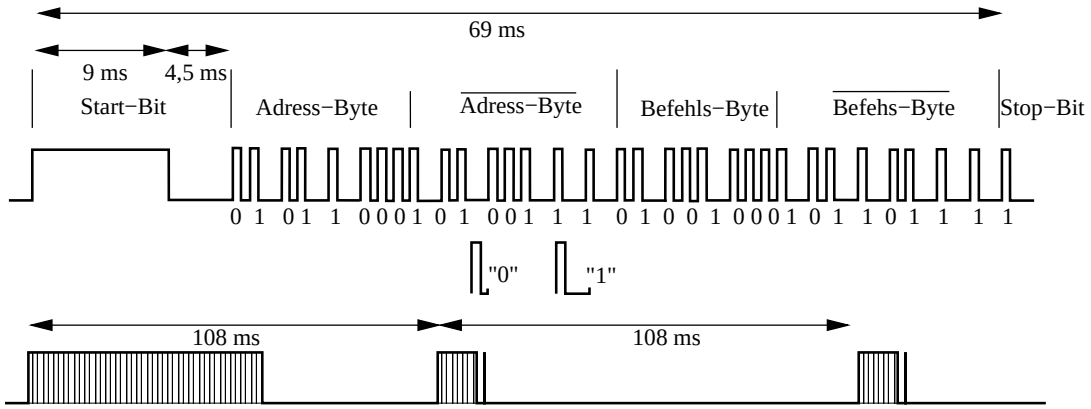


Abbildung 3.4.4: Protokollaufbau NEC-Codes

Taste	zwei Adresse-Bytes	zwei Befehls-Bytes	Stop-Bit
1	01011000 10100111	00001000 11110111	1
2	01011000 10100111	10001000 01110111	1
3	01011000 10100111	01001000 10110111	1
4	01011000 10100111	11001000 00110111	1
5	01011000 10100111	00101000 11010111	1
6	01011000 10100111	10101000 01010111	1
7	01011000 10100111	01101000 10010111	1
8	01011000 10100111	11101000 00010111	1
9	01011000 10100111	00011000 11100111	1
0	01011000 10100111	01011000 10100111	1

Tabelle 3.6: Bitfolgen des NEC CD-Players (NEC-Code)

Der NEC-Code arbeitet mit einer Trägerfrequenz von 36 kHz und wird im Pulsabstandsmodulations-Verfahren ausgestrahlt. Er beginnt mit einem 9 ms langen Start-Bit. Nach einer Pause von 4,5 ms beginnt das Datenwort mit 32 Bits. Dieses teilt sich in zwei Adress-Bytes und zwei Befehls-Bytes auf. Dabei werden jeweils acht Bits normal und die darauffolgenden acht Bits invertiert gesendet. Beendet wird der Code mit einem Stop-Bit. Das komplette Protokoll hat eine Länge von 67,5 ms [Woh02].

Bei längerem Tastendruck wird alle 108 ms das Start-Bit mit der Pause und ein einzelnes Bit wiederholt. Dies hat den Vorteil, dass der Stromverbrauch etwas geringer ist. Aber wenn das erste Protokoll nicht fehlerfrei empfangen wurde, kann nur ein erneutes Drücken der Taste zum erfolgreichen Dekodieren führen.

Eine logische 1 wird durch 0,56 ms High- 1,69 ms Lowpegel und eine logische 0 wird durch 0,56 ms High- 0,565 ms Lowpegel definiert. Weitere Informationen sind im

Buch [Woh02] zu finden. In Abbildung 3.4.4 ist der Code der Fernbedienung (Taste 3) des NEC CD-Players dargestellt.

In der Tabelle 3.6 sind die Tasten 1 bis 0 einer NEC CD-Player Fernbedienung aufgelistet. Es ist zu erkennen, dass das jeweils nachfolgende Byte der Adress- bzw. Befehls-Bytes invertiert gesendet werden.

3.4.6 RECS80-Code

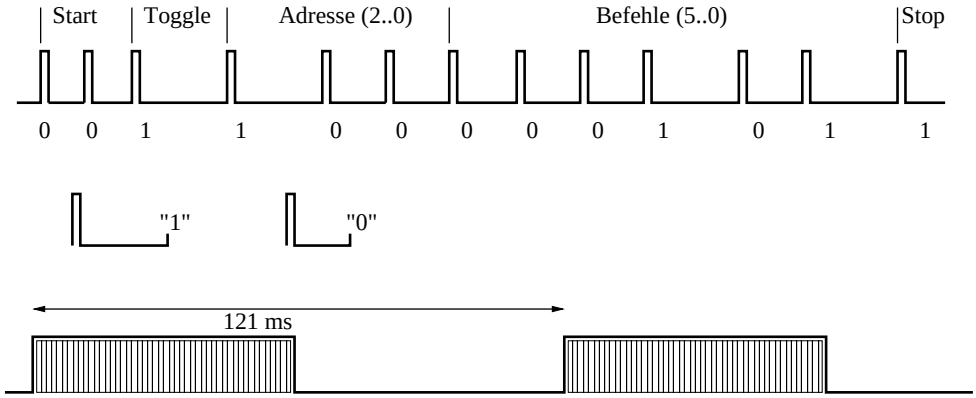


Abbildung 3.4.5: Protokollaufbau RECS80-Code

Taste	Bitfolge
0	00 0 100 000000 1
1	00 0 100 000001 1
2	00 0 100 000010 1
3	00 0 100 000011 1
4	00 0 100 000100 1
5	00 0 100 000101 1
6	00 0 100 000110 1
7	00 0 100 000111 1
8	00 0 100 001000 1
9	00 0 100 001001 1

Tabelle 3.7: Bitfolgen des Nokia SAT-Receivers (RECS80-Code)

Der RECS80-Code von Philips hat eine Modulationsfrequenz von 38 kHz und verwendet die Pulsabstandsmodulation [Woh02]. Die Übertragung beginnt mit zwei Start-Bits, gefolgt von einem Toggle-Bit. Die eigentliche Information steckt in den drei Adress- und sechs Befehls-Bits. Abgeschlossen wird der Code mit einem Stop-Bit.

Wenn die Taste gedrückt gehalten wird, wiederholt sich das Protokoll alle 121 ms. Jedes Bit beginnt mit einem Highpegel mit einer Länge von 140 μs. Bei einer logischen 1 ist die Pause 7,60 ms und bei einer logischen 0 ist die Pause 5,06 ms [Woh02].

In Abbildung 3.4.5 ist die Bitfolge der Taste 2 eines Nokia SAT-Receivers dargestellt. In Tabelle 3.7 sind die gesendeten Bitfolgen der Tasten 0 bis 9 aufgelistet.

3.4.7 MOTOROLA-Code

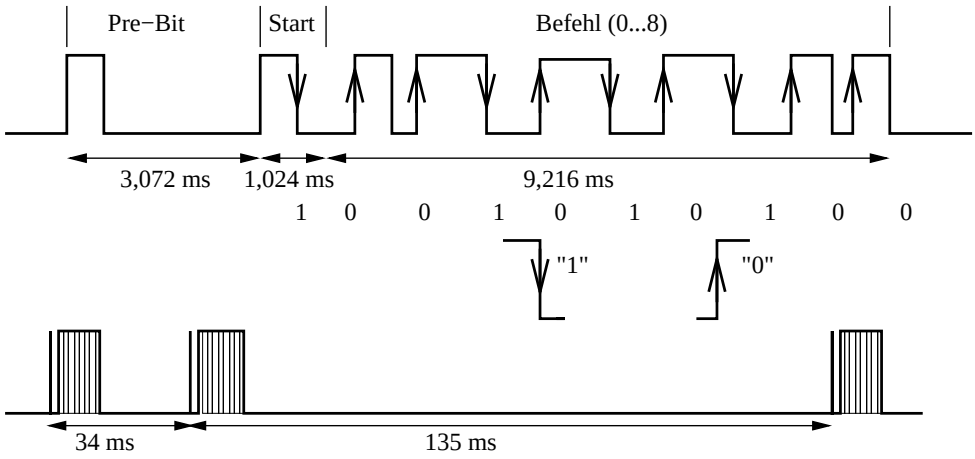


Abbildung 3.4.6: Protokollaufbau MOTOROLA-Code

Der MOTOROLA-Code besteht im wesentlichen aus einem neun Bits langen Datenwort. Die Bits sind biphase kodiert, aber im Gegensatz zum RC5-Code ist eine logische 1 eine fallende Flanke und eine logische 0 eine steigende Flanke. Die Trägerfrequenz ist typischerweise 32 kHz. Der Code beginnt mit einem Pre-Bit, welches mit einem 512 μs langen Highpegel beginnt, gefolgt von einer Pause. Die gesamte Länge des Pre-Bit beträgt 3,072 ms und dient zur Einstellung des AGC-Pegels. Das Start-Bit hat ein 512 μs High- und Lowpegel. Es entspricht einer logischen 1. Im Anschluß folgen die neun Befehls-Bits.

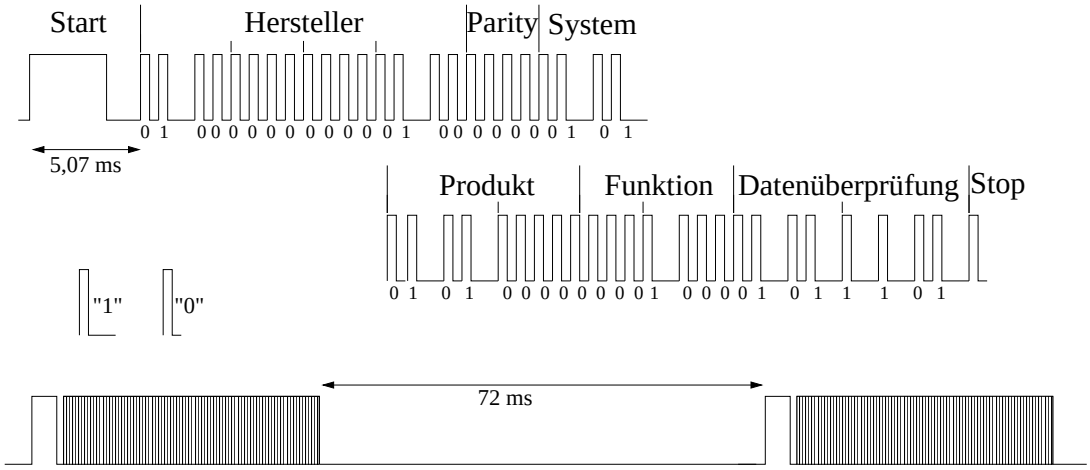
Das vollständige Protokoll erfordert aber vor dieser Übertragung noch eine Startnachricht. Diese besteht aus dem Pre-Bit, gefolgt von dem Start-Bit und neun logischen 1. Falls die Taste nur kurz gedrückt wurde, wird nach dem Datenwort auch noch die Stopnachricht gesendet. Diese hat den selben Aufbau wie die Startnachricht. Die vollständige Übertragung besteht demzufolge aus mindestens drei Telegrammen. Bei lang andauerndem Tastendruck wird das Datenwort alle 135 ms wiederholt. Erst beim Loslassen der Taste wird dann die Stopnachricht gesendet [Woh02].

In Abbildung 3.4.6 ist das gesendete Protokoll (Taste 4) der Fernbedienung eines Kathrein SAT-Receivers dargestellt. In Tabelle 3.8 sind die Bitfolgen des Datenworts, incl. Start-Bit, der Tasten 1 bis 0 aufgelistet.

Taste	Bitfolge
0	1 000010100
1	1 100010100
2	1 010010100
3	1 110010100
4	1 001010100
5	1 101010100
6	1 011010100
7	1 111010100
8	1 000110100
9	1 100110100

Tabelle 3.8: Bitfolgen des Kathrein SAT-Receivers (MOTOROLA-Code)

3.4.8 JAPAN-Code



Hinweis: Die Produkt-Bits werden direkt nach den System-Bits übertragen. Die Darstellung in zwei Zeilen ist nur aus Gründen der besseren Lesbarkeit gewählt worden.

Abbildung 3.4.7: Protokollaufbau JAPAN-Code

Der JAPAN-Code ist, wie der RC5-Code, normiert. Die Datenübertragung für Haushaltsgeräte wurde in einem japanischen Ausschuss festgelegt [2] . Das Protokoll besteht im wesentlichen aus dem 48 Bits langen Datenwort. Es beginnt mit einem 3,38 ms Start-Bit, gefolgt von einer 1,69 ms langen Pause. Die Daten beginnen mit 16 Hersteller-Bits, gefolgt von vier Parity- und vier System-Bits. Im Anschluß werden

[2] Japan's Association for Electric Home Application / Recommended standards for infrared remote controls [Woh02]

acht Produkt- und acht Funktions-Bits gesendet. Abgeschlossen wird das Datenwort mit acht Datenüberprüfungs-Bits. Der vollständige Code endet mit einem Stop-Bit und wird nach einer Pause von 72 ms solange wiederholt, wie die Taste der Fernbedienung gedrückt wird.

Die Hersteller-Bits sind bei der Normierungsbehörde registriert und werden, wie die System-Bits, bei der Herstellung des IC's per Maske in dem Chip implementiert. Die acht Produkt-Bits setzen sich aus zwei maskenprogrammierten und sechs vom Fernbedienungshersteller festgelegten Bits zusammen. Der gesendete Code ist durch externe Verdrahtung am IC festgelegt. Die Parity-Bits (gerade Parität) dienen zur Überprüfung von Übertragungsfehlern. Zusätzlich werden Fehler mit Hilfe der Datenüberprüfungs-Bits festgestellt. Diese werden durch einen speziellen Algorithmus, der die Werte der System-, Produkt- und Funktions-Bits umrechnet, bestimmt. Durch die Funktions-Bits ist jeder Taste ein spezieller Code zugeordnet [Woh02].

Das Signal ist mit 56 kHz moduliert. Eine logische 0 ist durch 0,42 ms High- und 0,42 ms Lowpegel, eine logische 1 durch 0,42 ms High- 1,27 ms Lowpegel definiert (Pulsabstandsmodulation). In Abbildung 3.4.7 ist das Datenwort der Taste 1 von der Fernbedienung des Technics CD-Player SL-PG200A dargestellt. Diese Bitfolge lautet in hexadezimaler Darstellung 0x40040550085D. In Tabelle 3.9 sind die Bitfolgen einiger Tasten aufgelistet. Zu erkennen ist, daß sich nur die Funktions-Bits und die Datenüberprüfungs-Bits unterscheiden.

Taste	Bitfolge
1	0100000000000100 0000 0101 01010000 00001000 01011101 0
2	0100000000000100 0000 0101 01010000 10001000 11011101 0
3	0100000000000100 0000 0101 01010000 01001000 00011101 0
4	0100000000000100 0000 0101 01010000 11001000 10011101 0
5	0100000000000100 0000 0101 01010000 00101000 01111101 0
6	0100000000000100 0000 0101 01010000 10101000 11111101 0
7	0100000000000100 0000 0101 01010000 01101000 00111101 0
8	0100000000000100 0000 0101 01010000 11101000 10111101 0
9	0100000000000100 0000 0101 01010000 00011000 01001101 0
10	0100000000000100 0000 0101 01010000 10100001 11110100 0

Tabelle 3.9: Bitfolgen des Technics CD-Players SL-PG200A (JAPAN-Code)

3.4.9 Sonstige Codes

Für die nun folgenden Protokolle wurden keine passenden Fernbedienungen gefunden. Somit konnten diese Protokolle nicht untersucht und in dieser Diplomarbeit nicht ausführlich beschrieben werden. Allerdings sind Literaturhinweise zu den jeweiligen Codes genannt.

- **SAMSUNG-Code 1**

  Der SAMSUNG-Code 1 besteht aus zwölf Bits, die sich aus drei Hersteller-Bits, drei Bits für bestimmte Tasten und sechs Bits für den Wert der gedrückten Taste zusammensetzen. Nähere Informationen zu diesem Protokoll sind in [Woh02] nachzulesen.

- **SAMSUNG-Code 2**

  Der SAMSUNG-Code 2 beginnt mit einem Start-Bit, gefolgt von zwölf Hersteller-Bits und acht Daten-Bits. Diese 21 Bits werden mindestens zweimal gesendet. Nähere Informationen zu diesem Protokoll sind in [Woh02] zu finden.

- **FERNOST-Code**

  Der FERNOST-Code beginnt mit einem 8,44 ms Start-Bit, gefolgt von einer 4,22 ms Pause. Im Anschluß werden acht Adress-Bits und ein Stop-Bit gesendet. Nach einer weiteren Pause von 4,22 ms werden die acht Befehls-Bits und ein Stop-Bit gesendet. Nähere Informationen zu diesem Protokoll sind in [Woh02] beschrieben.

- **RC6-Code**

  Der RC6-Code ist eine Weiterentwicklung des RC5-Codes. Bei diesem Protokoll halbiert sich die Signallänge. Dadurch können die Datenpakete länger ausfallen [Bar00].

4 LIRC – Linux Infrared Remote Control

4.1 Ziel

In diesem Kapitel soll aus dem LIRC-Projekt die Software vorgestellt werden, die für diese Diplomarbeit benötigt wird. Das Paket ist auf der LIRC-Homepage [LP02] verfügbar. Für die unterschiedliche Hardware sind die dazugehörigen Linux-Treiber aufzulisten, die in den Kernelmodulen implementiert sind. Außerdem sollen Programme vorgestellt werden, die Aufgaben des Fernbedienungs-Testers übernehmen können.

4.2 Allgemeines

LIRC wird von vielen Benutzern zum Steuern von mp3-, DVD-Playern und TV-Karten am Linux-PC benutzt. Dazu kann ein preiswerter Infrarot-Empfänger für den seriellen Port aufgebaut werden. Allerdings kann auch andere Hardware, wie z.B. der IR-Empfänger der TV-Karte genutzt werden. Zusammen mit einer IR-Fernbedienung ist es dann möglich, den nächsten Musiktitel oder einen anderen Kanal zu wählen.

Neben dem Starten von Programmen sind sogar Maus-Events per Fernbedienung simulierbar, also ist auch der Internet-Browser per Tastendruck steuerbar. LIRC erlaubt darüber hinaus bei einer Präsentation mittels einer Fernbedienung zur nächsten Folie zu wechseln, welche der an dem Laptop angeschlossene Projektor an die Wand projiziert [Ric02a].

4.3 Treibermodell

Damit der Aufbau von LIRC verdeutlicht wird, ist in Abbildung 4.3.1 das LIRC-Treibermodell dargestellt [Bar00]. Die Hard- und Software-Schicht ist getrennt. Außerdem ist zu erkennen, dass die Kernelmodule incl. Treiber dem Kernel-Space und die Anwenderprogramme dem User-Space zugeordnet sind.

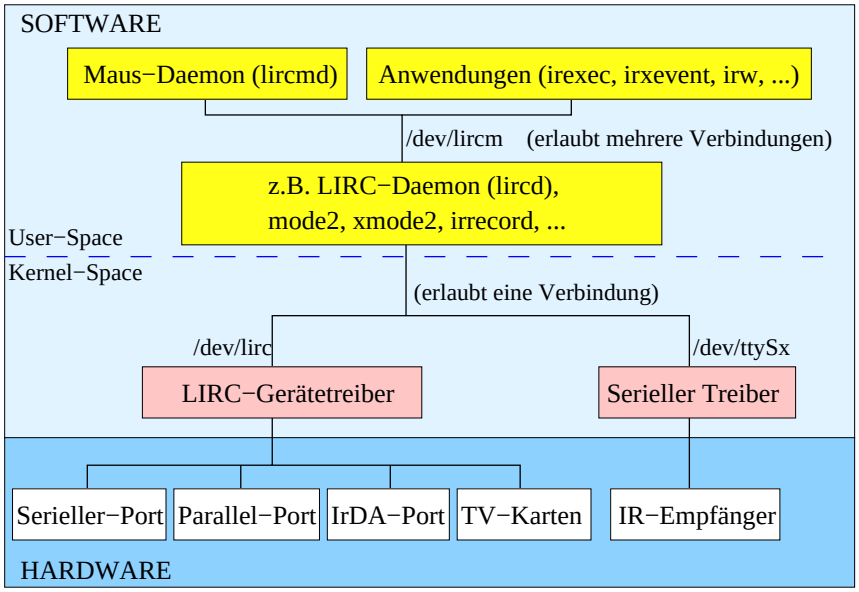


Abbildung 4.3.1: LIRC-Treibermodell

4.4 Kernelmodule

Für die unterschiedliche Hardware die von LIRC unterstützt wird, gibt es verschiedene LIRC-Kernelmodule. In diesen Modulen sind die jeweiligen Linux-Treiber realisiert. So wird u.a. folgende Hardware unterstützt:

- selbstgebaute IR-Sender und IR-Empfänger am seriellen Port
- selbstgebaute IR-Sender und IR-Empfänger am parallelen Port
- interner IR-Port (IrDA) in Laptops und PDA's
- TV-Karten
- IR-Empfänger am seriellen Port

Das Modul für den seriellen Port gibt die empfangenen Signale als Puls- und Pausenlänge über ein 'character device' aus. Diese Gerätedatei */dev/lirc* wird bei der Installation mit dem Befehl `mknod /dev/lirc c 61 0` angelegt. Der Treiber misst die Abstände zwischen Interrupts, die der Zustandswechsel der DCD-Leitung [1] am seriellen Port auslöst. Unterstützt werden alle seriellen Schnittstellen mit einem 16450/16550A-UART-kompatiblen Chipsatz [Bar00]. Wird in */etc/modules.conf* die Zeile `alias char-major-61 lirc_serial` ergänzt, kann das Kernelmodul *lirc_serial* bei Bedarf automatisch eingebunden werden [Sch00].

[1] Data Carrier Detect

Da bei einigen IR-Protokollen Pulse vorkommen, die kürzer als 100 μs sind, reicht die Auflösung der Systemuhr nicht aus. Deshalb wird die Kernelfunktion `do_gettimeofday()` verwendet, deren Auflösung Mikrosekunden-Genauigkeit erreicht [Bar00].

Die interne IR-Schnittstelle des Sharp Zaurus SL-5500G wird vom Kernelmodul *lirc_sir* unterstützt. Für Untersuchungen am PC, bzw. zur Programmierung mit Hilfe der Entwicklungsumgebung (Kapitel 6.3.1 ab Seite 35) werden selbstgebaute IR-Sender (Kapitel 5.5 auf Seite 30) und IR-Empfänger (Kapitel 5.4 ab Seite 27) verwendet. Diese werden vom Kernelmodul *lirc_serial* unterstützt.

4.5 Software des LIRC-Projektes

4.5.1 /dev/lirc

In der Gerätedatei */dev/lirc* werden die empfangenen Daten, die vom Kernelmodul verarbeitet wurden, für andere Programme zur Verfügung gestellt. Die Daten haben folgenden Aufbau:

- Bit 0-23: Länge der Pulse bzw. Pause in Mikrosekunden
- Bit 24: 1 bedeutet Pulse (pulse), 0 bedeutet Pause (space)

Pulse- bzw. Pausenlängen von mehr als 16 Sekunden werden als 0xFFFFFF übergeben. Ob das Signal als Puls- und Pausenlänge vom Kernelmodul übergeben wird, kann durch die Funktion `ioctl` festgestellt werden (`LIRC_MODE_MODE2`).

4.5.2 mode2, xmode2

Das Programm ”mode2” gibt die Puls- und Pausenlänge des IR-Signals, wie in Abbildung 4.5.1 dargestellt, in die Standard-Ausgabe (stdout) aus. Der ausgegebene Zahlenwert entspricht der tatsächlichen Zeit in Mikrosekunden.

```
pulse 3569
space 1666
pulse 514
space 356
pulse 517
space 1227
pulse 517
space 356
...
```

Abbildung 4.5.1: Ausgabe des IR-Signals (mode2)

Bei dem Programm ”xmode2” werden die Puls- und Pausenlängen grafisch in einer X11-Umgebung dargestellt (siehe Abbildung 4.5.2).

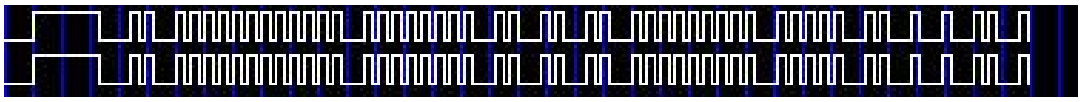


Abbildung 4.5.2: Grafische Ausgabe des IR-Signals (xmode2)

Beide Programme lesen die Werte aus der Gerätedatei */dev/lirc*. Sie dienen zum einen der Überprüfung der selbstgebauten IR-Empfängerschaltung, und zum anderen dem Test von IR-Fernbedienungen. So ist es möglich, den gesendeten Code (Kapitel 3.4) zu erkennen und den Befehl abzulesen.

Dieses Programm arbeitet nicht mit Hardware, die das Signal selber dekodiert, wie z.B. einige TV-Karten oder Irman (*/dev/ttySx*).

4.5.3 irrecord

Das Programm ”irrecord” dient zum Anlernen von noch unbekannten Fernbedienungen. Dabei wird durch eine Analyse der Signalfolgen versucht, das Protokoll mit allen Besonderheiten wie etwa Toggle-Bits automatisch zu erkennen. Das Programm wird mit dem Befehl `irrecord lircd.conf` aufgerufen. Dann folgen einige Informationen des Programms und die Aufforderung, mehrere Tasten der Fernbedienung zu drücken. Auch das Benennen der einzelnen Tasten ist notwendig. Nach erfolgreichem Anlernen wird eine Konfigurationsdatei erzeugt und in dem aktuellen Verzeichnis gespeichert [Ric02a].

4.5.4 lircd.conf

Der Aufbau der Konfigurationsdatei *lircd.conf*, welche die erforderlichen Daten für den LIRC-Daemon liefert, kann z.B. folgendermaßen aussehen:

```
begin remote
  name  TechnicsCD
  bits           16
  flags SPACE_ENC
  eps            30
  aeps          100

  header       3569  1663
  one           514  1228
  zero          514   357
  ptrail        516
  pre_data_bits   32
  pre_data       0x40040550
```

```
    gap          74933
    toggle_bit       0

      begin codes
          play                     0x0000000000005005
          1                        0x000000000000085D
      end codes
  end remote
```

Bei der Analyse der Signalfolge erkannte das Programm ”irrecord”, dass sich die ersten 32 Bits bei jeder Taste, die auf der Fernbedienung beim Anlernen gedrückt wurden, wiederholen (vgl. JAPAN-Code - Kapitel 3.4.8 auf Seite 17). Damit diese redundanten Daten nicht bei jeder Taste aufgeführt werden müssen, wurden diese 32 Bits als `pre_data 0x40040550` in den Kopfzeilen gespeichert. Dadurch müssen beim Tastencode nur die restlichen 16 Bits des Datenworts gespeichert werden.

Außerdem ist das Start-Bit `header` mit 3569 μs High- und 1663 μs Lowpegel festgelegt. Auch eine logische 1, `one`, und eine logische 0, `zero`, ist mikrosekundengenau in den Kopfzeilen gespeichert (vgl. JAPAN-Code).

Die Werte in den Kopfzeilen sind bei vielen Fernbedienungs-Protokollen unterschiedlich. Die aufgeführten Werte beziehen sich speziell auf die IR-Fernbedienung des Technics CD-Players SL-PG-200A. Allerdings sind diese Werte für alle Geräte, die den JAPAN-Code verwenden, sehr ähnlich.

Der Name der Fernbedienung, hier `TechnicsCD`, und die Namen der Tasten, hier `play` bzw. `1`, sind beim Anlernen mit dem Programm ”irrecord” so benannt worden.

4.5.5 LIRC-Daemon (lircd)

Die Zuordnung der übermittelten IR-Signale zu den gedrückten Tasten auf einer Fernbedienung übernimmt lircd – der LIRC-Daemon. Mit Hilfe der Konfigurationsdatei, in der alle wichtigen Parameter beschrieben werden, kann lircd alle gängigen IR-Protokolle dekodieren [Bar00]. Die empfangenen Daten werden vom Kernelmodul über die Gerätedatei */dev/lirc* bzw. */dev/ttySx* an den Daemon übergeben.

Gestartet wird der LIRC-Daemon als root mit dem Befehl `lircd`, vorausgesetzt die Konfigurationsdatei befindet sich im Verzeichnis */etc/lircd.conf*. Es ist auch möglich den Daemon zu starten, wenn die Konfigurationsdatei in einem anderen Verzeichnis gespeichert ist. Das entsprechende Verzeichnis ist dann beim Starten anzugeben und der Befehl lautet dann: `lircd /tmp/user/lircd.conf`

Falls mehr als eine Fernbedienung benutzt werden soll, können mit dem Befehl `cat config1 config2 > config` zwei Konfigurationsdateien zusammengefügt werden.

Für den Fall, dass keine passende Konfigurationsdatei auf der Homepage [LP02] für die Fernbedienung vorhanden ist, kann diese Datei mit dem Programm ”irrecord” erzeugt werden.

4.5.6 irw

Das Programm ”irw” beobachtet die Gerätedatei */dev/lircd* und gibt, falls die empfangene Signalfolge ein Fernbedienungsbefehl aus der Konfigurationsdatei *lircd.conf* ist, eine Zeile in der Standard-Ausgabe (stdout) aus. Diese Zeilen können folgendermaßen aussehen:

```
0000400405505005 00 play TechnicsCD
000040040550085d 00 1 TechnicsCD
```

Die vier Parameter pro Zeile sind:

1. Bitcode
2. Wiederholungszähler (wenn die Taste längere Zeit gedrückt bleibt)
3. Name der Taste
4. Name der Fernbedienung

Der LIRC-Daemon muß gestartet sein, damit das Programm arbeiten kann.

4.5.7 rc

Mit dem Programm ”rc” und einem IR-Sender (Kapitel 5.5 ab Seite 30) ist es möglich, Infrarot-Signale zu senden. Mit dem Befehl `rc SEND_ONCE TechnicsCD play` wird das entsprechende Signal einmal gesendet, damit der Technics CD-Player beginnt eine CD abzuspielen. Die folgenden Anweisungen sind möglich:

- `SEND_ONCE` – sendet den entsprechenden IR-Befehl einmal
- `SEND_START` – beginnt mit dem wiederholten Senden des IR-Befehls
- `SEND_STOP` – beendet das wiederholte Senden des IR-Befehls
- `LIST` – listet die Fernbedienungen auf, die in der Konfigurationsdatei *lircd.conf* aufgelistet sind

Außerdem muß der Name der Fernbedienung und der Name der gewünschten Taste, entsprechend der Bezeichnung in der Konfigurationsdatei *lircd.conf*, angegeben werden. Die Reihenfolge der Argumente ist dabei zu beachten:

```
rc  Anweisung  Fernbedienung  Taste
```

5 Hardware

5.1 Ziel

In diesem Kapitel soll die verwendete Hardware vorgestellt werden. Der Sharp Zaurus SL-5500G übernimmt die Aufgaben des Infrarot-Testgerätes. Außerdem wird ein Linux-Rechner zum Entwickeln der notwendigen Software verwendet. Dieser Linux-Rechner benötigt einen IR-Empfänger und IR-Sender.

5.2 Sharp Zaurus SL-5500G

In Kapitel 2.3.3 auf Seite 6 sind die wesentlichen Merkmale des Sharp Zaurus SL-5500G vorgestellt worden. Beim Entwickeln der Programme hat sich als sehr nützlich herausgestellt, Speicherkarten zu verwenden. Das USB-Gerät *SanDisk cruzer*, welches SD-Karten aufnimmt, wird von SuSE 8.0 als Wechselmedium */media/sda1* erkannt. Daher ist das Speichern und Übertragen von Daten besonders einfach und schnell.

Ein ROM-Update, z.B. Betriebssystem-Update oder das Aufspielen eines selbst kompilierten Kernels, funktioniert mit den CF-Karten problemlos. Die folgenden Speicherkarten sind verwendbar:

- Compact Flash Type II (CF)
- Secure Digital (SD)
- Multi Media Card (MMC)

5.3 Linux-PC

Die Systemanforderungen an den Linux-PC sind nicht hoch. Für die Softwareentwicklung ist ein PC notwendig, allerdings wurden für diese Diplomarbeit zwei PC's mit folgenden, wesentlichen Komponenten verwendet.

Rechner 1:

- 350 MHz Pentium II Prozessor
- 10 GB Partition der Festplatte

- freier serieller Port
- 256 MB Arbeitsspeicher (RAM)

Rechner 2:

- 400 MHz Pentium II Prozessor
- 4 GB Partition der Festplatte
- freier serieller Port
- 256 MB Arbeitsspeicher (RAM)

Verglichen mit aktuell erhältlichen PC's sind die eingesetzten Prozessoren sehr langsam. Allerdings funktioniert das Kompilieren und Linken von den selbst programmierten Entwicklungen in einer akzeptablen Zeit von nur wenigen Sekunden.

Der notwendige Festplattenspeicher, der für die verwendete SuSE 8.0-Installation incl. Entwicklungsumgebung benötigt wird, ist sehr gering. Sogar beim Rechner 2 mit der kleineren Partition von nur 4 GB ist noch mehr als 1 GB Speicherplatz verfügbar.

5.4 Infrarot-Empfänger

5.4.1 Allgemeines

Infrarot-Licht kann mit Hilfe von Fotodioden ausgewertet werden. Germanium-Fotodioden bestehen aus stäbchenförmigen Kristallen mit gezogenem pn-Übergang. Das metallische Gehäuse verfügt am Ende über eine kleine Linse, durch die das Licht auf den Kristall fällt. Dabei ändert sich der pn-Übergang unter Lichteinfluß. Der Übergang vom n- in das p-Gebiet wird in Sperrrichtung vorgespannt. Dabei addieren sich die angelegte Spannung und die Diffusionsspannung.

5.4.2 Empfänger-IC TSOP17xx

Für die Auswertung der Daten mit Hilfe eines Computers werden in dieser Diplomarbeit Infrarot-Empfänger-IC's verwendet. In diesen sind sowohl eine Fotodiode, als auch Elektronik zum Verstärken und Demodulieren integriert. In Abbildung 5.4.1 ist ein TSOP17xx und das dazugehörige Blockschaltbild dargestellt [Vis01].

Es gibt Empfänger-IC's für unterschiedliche Modulationsfrequenzen. Vishay Telefunken bietet für 30 kHz, 33 kHz, 36 kHz, 36,7 kHz, 38 kHz, 40 kHz und 56 kHz angepaßte Empfänger an. Bei den durchgeführten Messungen wurde das TSOP1738 eingesetzt. Dieses IC arbeitet auch mit den anderen Modulationsfrequenzen, allerdings ist der Bandpaß auf die angegebene Frequenz von 38 kHz abgestimmt. Die erzielten Ergebnisse sind bei anderen Modulationsfrequenzen zwar akzeptabel, es ergeben sich allerdings geringere Reichweiten.

Die Baugruppen im Blockschaltbild haben die folgenden Funktionen:

Input: In diesem Block wird für die Vorspannung der Fotodiode und die Auskopplung des Nutzsignals gesorgt. Dabei wird die Wechsel- und Gleichspannung in verschiedenen Schaltungen verarbeitet. Das Nutzsignal wird zum AGC-Block weitergeleitet.

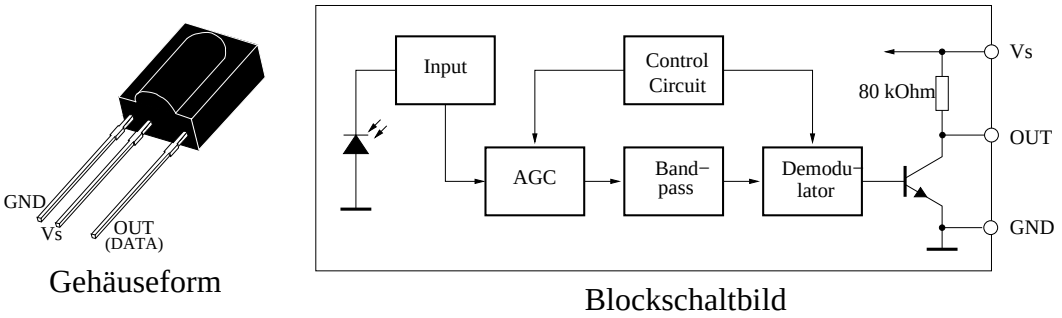
AGC: Automatic Gain Control verstärkt das Nutzsignal mit bis zu 50 dB. Je nach Abstand zwischen IR-Fernbedienung und der Fotodiode des Empfänger-IC's verändert sich die von der Diode erzeugte Spannung.

Bandpaß: Der Bandpaß wird bei der Produktion auf eine feste Modulationsfrequenz abgeglichen und hat eine Güte von 10.

Control Circuit: Der Control Circuit regelt den AGC und den Demodulator.

Demodulator: Diese Stufe sorgt dafür, dass die Trägerfrequenz aus dem Signal demoduliert wird. Nach dieser Stufe hat das Ausgangssignal Highpegel, sobald moduliertes Signal anliegt und Lowpegel, falls nicht moduliertes Signal anliegt.

Transistor: Die verwendete Transistorschaltung ist eine Emitterschaltung. Dadurch wird das Ausgangssignal des Demodulators invertiert. Demzufolge hat der OUT-Pin des Empfänger-IC's Lowpegel (0 Volt), solange IR-Signal empfangen wird. Falls kein IR-Licht vom IC empfangen wird, sperrt der NPN-Transistor und der OUT-Pin hat High-Pegel. —> 'aktiv low'



Gehäuseform Blockschaltbild

Abbildung 5.4.1: Empfänger-IC TSOP17xx

5.4.3 Empfänger-Schaltung

In Abbildung 5.4.2 ist eine Empfängerschaltung aus dem Kommunikationssysteme-Praktikum dargestellt [Ric02a]. Die RS-232-Daten sind bipolar: Eine Spannung von +3 bis +15 Volt markiert einen 'ON'- oder logische 0, während eine Spannung von -3 bis -15 Volt einen 'OFF' oder logische 1 bedeutet [Eit01].

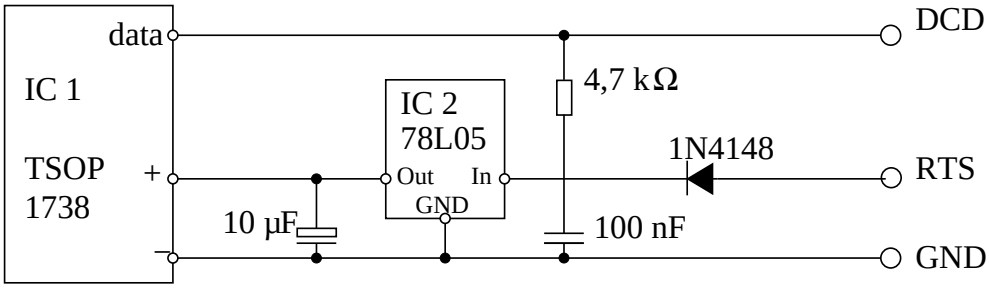


Abbildung 5.4.2: Schaltplan des Empfängers

Der RTS-Leitung [1] dient zusammen mit GND [2] zur Spannungsversorgung. Über die Diode wird nur die positive Spannung geleitet (Einweggleichrichtung). Der Festspannungsregler 78L05 (IC2) stabilisiert diese Spannung auf 5 Volt. Der 10 μF Elektrolytkondensator lädt sich auf und versorgt den IR-Empfänger (IC1 - TSOP1738) mit einer konstanten Spannung. Der 4,7 kΩ Widerstand sorgt dafür, dass am DCD [3] eine Spannung von ca. 4 Volt anliegt. Das Empfänger-IC ist 'aktiv-low'. Dies bedeutet, der Data-Anschluss wird jedesmal wenn Infrarot-Signal empfangen wird, nach Low gezogen. Demzufolge hat beim Empfang von IR-Pulsen der DCD 0 Volt, was alle gängigen I/O Chipsätze als OFF (logische 1) interpretieren. Der 100 nF Kondensator sollte direkt neben dem IC2 plaziert werden, da dieser Signalstörungen und Signalspitzen unterdrückt.

Hinweis: Bei der Überprüfung eines Laptops stellte sich heraus, dass eine maximale Spannung von ca. 5 Volt am DCD anliegt. Der Festspannungsregler benötigt aber eine Eingangsspannung von mindestens 6,7 Volt. (Es ist also möglich, falls die Spannung an anderen Laptops auch bei ca. 5 Volt liegt, den Festspannungsregler durch eine Schutzschaltung mit einer Zenerdiode auszuwechseln. Hierbei muß beachtet werden, dass die maximale Eingangsspannung des TSOP1738 nur 6 Volt beträgt.)

9-polig	25-polig	Bezeichnung
1	8	DCD Data Channel Received Line Signal Detector
5	7	GND Signal Ground
7	4	RTS Request To Send

Tabelle 5.1: RS-232 Schnittstelle [Ric02b]

[1] Request To Send
[2] Ground
[3] Data Carrier Detect

5.5 Infrarot-Sender

5.5.1 Allgemeines

Die Infrarot-Sendedioden (IRED) [4] strahlen mit einer Wellenlänge von ca. 900 nm. Diese Infrarot-Lumineszensdioden basieren auf Gallium-Arsenid (GaAs) und erzielen höhere externe Quantenwirkungsgrade als Lumineszensdioden (LED) [5] im sichtbaren Bereich. Das emittierte Licht wird dabei gebündelt. Dies wird durch die gekrümmte, transparente Kunststoffoberfläche erreicht, da diese wie eine Sammellinse funktioniert. Außerdem setzt der Kunststoff den Grenzwert der Totalreflexion herab [Woh02].

Der Abstrahlwinkel hat bei den IRED's eine große Bedeutung, da ein kleinerer Öffnungswinkel eine deutlich höhere Lichtstärke bewirkt. Allerdings sollte ein Infrarot-Sender auch dann funktionieren, wenn die Sendediode nicht direkt in Richtung des Empfängers strahlt.

5.5.2 Sender-Schaltung

In Abbildung 5.5.1 ist der Schaltplan eines IR-Senders von der LIRC-Homepage [LP02] dargestellt. Die anliegenden Spannungen an der seriellen Schnittstelle sind identisch zu denen der IR-Empfängerschaltung. Jeder Signalpin darf mit maximal 50 mA belastet werden [Eit01].

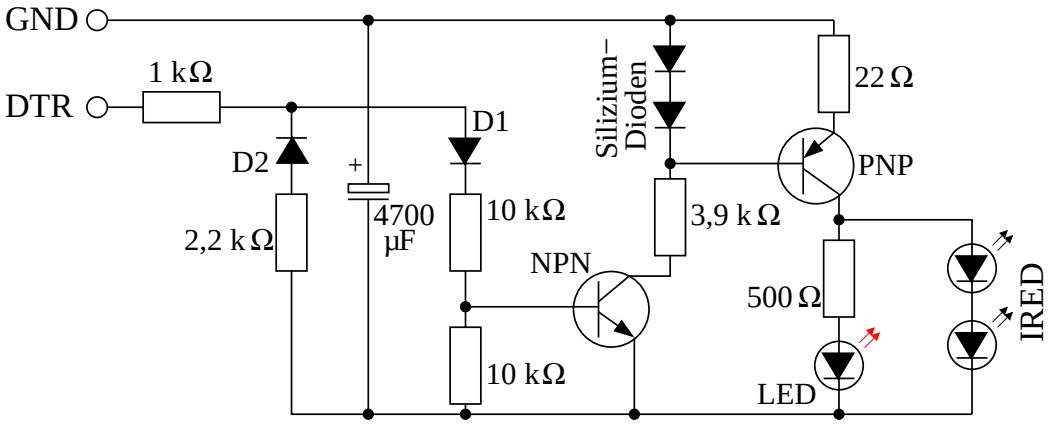


Abbildung 5.5.1: Schaltplan des Senders

Schließt man diese Schaltung an die serielle Schnittstelle des Computers an, wird der 4700 μF Elektrolytkondensator durch die negative Spannung am DTR-Stift [6] über die Diode 2 (D2) in ca. einer Minute aufgeladen. Wird die Zeitkonstante τ berechnet, ergibt sich ein Wert von 15 Sekunden (Gleichung 5.5.1). Nach fünf τ ist der Kondensator vollständig aufgeladen (Gleichung 5.5.2).

[4] Infrared Emitting Diode
[5] Light Emitting Diode
[6] Data Terminal Ready

$$\tau = RC = (1k\Omega + 2,2k\Omega)4700\mu F = 15s \tag{5.5.1}$$

$$5\tau = 75s \tag{5.5.2}$$

Der Strom wird durch den 2,2 kΩ Widerstand und den 1 kΩ Widerstand auf maximal 5 mA begrenzt (Gleichung 5.5.3).

$$I_{max} = \frac{U}{R} = \frac{15V}{1k\Omega + 2,2k\Omega} = 5mA \tag{5.5.3}$$

Der NPN-Transistor ist durch die Diode 1 (D1) gesperrt. Wird ein Sendesignal an die Schnittstelle geschickt, liegen an der DTR-Leitung positive Pulse an. Diese Pulse haben keinen Einfluß auf den Kondensator, da Diode 2 sperrt. Bei jedem positiven Puls wird der NPN-Transistor leitend. Dadurch wird die Spannung an der Basis des PNP-Transistors kleiner. Der PNP-Transistor wird leitend und es fließt Strom vom Kondensator über den 22 Ω Widerstand zu den Sendedioden. Die Infrarot-Dioden senden mit einer Reichweite von ca. 5 Metern.

Die rote LED und der 500 Ω Widerstand sind optional, sie dienen zur Überwachung des IR-Bursts, entladen aber den Kondensator zusätzlich.

Beachtet werden muß, dass die Strompulse den Kondensator schnell entladen. Allerdings reicht die Kapazität von 4700 μF bei den meisten IR-Protokollen aus, wenn zwischen den Befehlen eine kleine Pause eingelegt wird, damit sich der Kondensator erneut aufladen kann.

Hinweis: Alle Typen von Low- / Medium-Power IR-Dioden und Transistoren sind verwendbar. Es ist allerdings wichtig, dass es sich bei den beiden Offset-Dioden um Silizium-Dioden handelt. Diese haben einen Spannungsabfall in Durchlaßrichtung von jeweils 0,7 Volt. Dies ergibt eine Offset-Spannung von 1,4 Volt. Germanium-Dioden haben eine Durchlaßspannung von ca. 0,2 Volt. Es ist möglich, anstatt der zwei Sendedioden auch drei Infrarot-Dioden zu verwenden, allerdings muß dann der Wert des 22 Ω Widerstandes verkleinert werden. Es muß beachtet werden, dass sich dann der Stromverbrauch der Schaltung erhöht und die Kondensatorkapazität vergrößert werden muß.

9-polig	25-polig	Bezeichnung
4	20	DTR Data Terminal Ready
5	7	GND Signal Ground

Tabelle 5.2: RS-232 Schnittstelle [Ric02b]

6 Entwicklungsumgebung

6.1 Ziel

In diesem Kapitel soll gezeigt werden, welche Einstellungen zum Synchronisieren des Sharp Linux-PDA's mit einem SuSE Linux-PC über die USB-Schnittstelle notwendig sind. Außerdem sollen die verschiedenen Entwicklungstools vorgestellt und erklärt werden, damit Software auf dem SuSE 8.0 PC entwickelt und getestet werden kann. Erst dann wird diese Software für den Sharp Zaurus SL-5500G kompiliert und ist auf dem PDA ausführbar.

6.2 Synchronisieren des Sharp Zaurus SL-5500 mit SuSE Linux

Der Linux PDA - Sharp Zaurus SL5500G wird nur mit einer Synchronisationssoftware für Microsoft Windows ausgeliefert. Die mitgelieferte Windows-Software zum Pflegen des Terminplaners und der Kontaktdatenbank funktioniert gut unter Windows 98 und Windows 2000. Diese übernimmt auch die Einstellungen unter Windows.

Damit nicht zum Synchronisieren dieser Daten jedesmal auf das Windows Betriebssystem gewechselt werden muß, sollten Anwenderprogramme wie z.B. Terminplaner, Kontaktdatenbank auch unter Linux gepflegt werden können. Allerdings müssen dafür unter SuSE 8.0 und SuSE 8.1 einige Einstellungen überprüft bzw. verändert werden [jre02].

6.2.1 Installieren des Kernelmoduls usbdnet - SuSE 8.0

Für SuSE 8.0 wird das zusätzliches Kernelmodul *usbdnet* benötigt, um den Zaurus SL-5500G über die USB-Schnittstelle zu verbinden. Damit ein Kernelmodul kompiliert werden kann, sind die SuSE-Linux Kernel-Quellen nötig. Diese können mit *YaST2* installiert werden.

Ein Patch mit den usbdnet-Quellen kann bei [Rua02] heruntergeladen werden. Dieser wird auf die Kernel-Quellen folgendermaßen angewendet:

```
cd /usr/src/linux
zcat usbdnet-SuSE-8.0-2.4.18.patch.gz | patch -p1
```

Mit dem folgenden Befehl wird der Kernel rekonfiguriert:

```
zcat /proc/config.gz >.config
make menuconfig
```

Das Modul `USBD Netzwerk (Encapsulated) Host-to-Host Link` muß aus der Liste der USB-Treiber ausgewählt werden. Die Werte für den Hersteller, das Produkt, die Class und die SubClass brauchen nicht ausgefüllt werden. Nachdem die Konfiguration gespeichert ist, muß der folgende Befehl ausgeführt werden:

```
make dep modules
```

Wenn dies erfolgt ist, muß das erstellte Modul in das entsprechende Modulverzeichnis kopiert und *depmod* ausgeführt werden:

```
cp drivers/usb/usbdnet.o \
                /lib/modules/2.4.18-4GB/kernel/drivers/usb/

depmod -a
```

Hinweis: Es ist ausreichend, dieses Modul zu kopieren. Die Installation eines anderen Moduls oder eines neuen Kernels ist nicht erforderlich [jre02].

6.2.2 Konfigurieren des Hotplug-Subsystems

Das *usbdnet*-Modul funktioniert nicht mit dem *uhci*-Treiber, aber mit dem *usb-uhci*-Treiber. Außerdem nimmt der *CDCEther*-Treiber das Device, welches aber durch *usbdnet* benutzt werden soll. Um dies zu erreichen, muss zunächst */etc/sysconfig/hotplug* modifiziert werden. Dafür ist die folgende Zeile zu ändern:

```
HOTPLUG_USB_HOSTCONTROLLER_LIST= \
                            "uhci usb-uhci usb-ohci ehci-hcd"
```

in

```
HOTPLUG_USB_HOSTCONTROLLER_LIST= \
                            "usb-uhci uhci usb-ohci ehci-hcd"
```

Dann muß *usbdnet* den HOTPLUG_USB_NET_MODULES hinzugefügt werden:

```
HOTPLUG_USB_NET_MODULES= \
                "pegasus usbnet catc kaweth CDCEther usbdnet"
```

Außerdem muß *CDCEther* in */etc/hotplug/blacklist* deaktiviert werden. Dafür wird die folgende Zeile hinzugefügt:

```
# Sharp Zaurus mag CDCEther nicht
CDCEther
```

Mit der folgenden Befehlskette wird das Hotplug-Subsystem neu gestartet:

```
rchotplug stop;sleep 90;rchotplug start
```

Jetzt wird durch den Befehl `lsmod` überprüft, ob das Modul *usb-uhci* geladen wurde. Falls noch das Modul *uhci* geladen ist, muß mit `rchotplug stop` das Hotplug-Subsystem beendet werden und manuell alle USB-Module entfernt werden. Mit `rchotplug start` wird das Subsystem wieder gestartet. Falls nicht alle relevanten Module (z.B. USB-Maus) manuell entfernt werden konnten, ist das System neu zu starten. (Das Neustarten des Systems funktioniert schneller als alle USB-Module manuell zu entfernen.)

Beim Einlegen des Zaurus in die Docking-Station (Cradle) muß nach spätestens zwei Sekunden ein Piepton ertönen. Nach der Eingabe des Befehls `lsmod` sollte jetzt auch das Modul *usbdnet* aufgeführt sein, allerdings das Modul *CDCEther* darf in der Liste nicht erscheinen.

Die Ausgabe des Befehls `ifconfig -a` sollte das Device *usb0* enthalten [jre02].

6.2.3 Einrichten des Netzwerks

Um das USB-Netzwerk zu konfigurieren, muß *YaST2* gestartet werden. Bei *'Konfiguration der Netzwerkkarte'* muß unter *'Manuelle Konfiguration' 'USB'* ausgewählt werden. Als statische IP-Nummer ist eine Adresse im selben Subnetz wie bei dem Sharp Zaurus zu wählen. Die folgenden IP-Adressen können verwendet werden.

- 192.168.129.200 USB-Netzwerk
- 192.168.129.201 Zaurus (Voreinstellung)

Aufgrund eines Problems mit dem Zusammenspiel zwischen dem Hotplug-Subsystem und dem Netzwerkkonfigurationssystem ist es erforderlich, den folgenden Link in */etc/sysconfig/network* zu erstellen:

```
cd /etc/sysconfig/network
ln -s ifcfg-eth-usb-0 ifcfg-usb0
```

Abschließend muß in */etc/sysconfig/hotplug* der Parameter `HOTPLUG_START_NET` auf `yes` geändert werden.

Beim nächsten Einstecken des Sharp Zaurus in die Ladestation (Cradle) wird das Netzwerk unter SuSE 8.0 automatisch geladen [jre02].

6.2.4 Einstellungen bei SuSE 8.1

Bei SuSE 8.1 ist *usbdnet*-Modul bereits installiert und kann genutzt werden. Die Einstellungen, die in den Kapiteln 'Konfigurieren des Hotplug-Subsystems' und 'Einrichten des Netzwerks' beschrieben sind, müssen kontrolliert und gegebenenfalls die Änderungen durchgeführt werden.

Falls das Synchronisieren nicht automatisch funktioniert, kann das *usbdnet*-Modul und das *usb0*-Netzwerk auch manuell als root geladen, bzw. gestartet werden, wenn der Zaurus in der Docking Station steckt.

```
insmod usbdnet
ifconfig usb0 Rechnername
```

Außerdem sollten die statischen IP-Adressen in */etc/hosts* eingetragen werden.

```
192.168.129.201 sharp sl5500 zaurus
192.168.129.200 Rechnername.WORKGROUP    Rechnername (kurz)
```

In den verschiedenen Zaurus-Foren ([Zau03c], [Zau03d], [Zau03e]) gibt es für registrierte Mitglieder noch diverse Ansätze und Beschreibungen, welche Einstellungen bei SuSE 8.1 vorgenommen werden müssen, damit das Synchronisieren automatisch funktioniert.

Da aber auch die Entwicklungstools, die in den folgenden Kapiteln vorgestellt werden, auf dem SuSE 8.1 Rechner nicht funktionierten, wurde auch bei dem Rechner 2 SuSE 8.0 installiert.

6.2.5 Installieren des Qtopia Desktops

Es gibt zwei Möglichkeiten, den Qtopia Desktop zu installieren. Diese Anwendung ist auf der *Zaurus Entwickler-CD für SuSE Linux 8.0* zu finden. Wird der Inhalt der CD laut Anweisung installiert, übernimmt der Installations-Assistent sämtliche Einstellungen.

Die andere Möglichkeit ist, Qtopia Desktop bei [Zau03b] herunterzuladen. Dieses RPM-Paket wird durch den Befehl `rpm -Uvh qtopia-desktop.rpm` installiert.

Nach der erfolgreichen Installation kann der Terminplaner, die Kontaktdatenbank, und die Aufgabenliste synchronisiert werden. Außerdem können Anwendungen als IPK-Paket (z.B. Terminal, File-Manager, Spiele) auf den Sharp Zaurus SL-5500G kopiert werden. Diese Pakete können auf dem PDA entpackt und installiert werden.

6.3 Entwicklungstool

6.3.1 Auswahl der Entwicklungstool

Ein großes Problem in dieser Diplomarbeit war es, lauffähige Entwicklungstools zu finden. Diese müssen so in die Entwicklungsumgebung eingebunden werden, das sowohl Software für den Zaurus kompiliert werden kann, als auch diese Software auf dem Linux PC getestet werden kann.

Zaurus Entwickler-CD

Die *Zaurus Entwickler-CD für SuSE Linux 8.0* ist im Lieferumfang enthalten, wenn der Sharp Zaurus SL-5500G als Entwickler-PDA bestellt wird. Der Inhalt dieser CD kann auch bei [Sha02] heruntergeladen werden (ca. 170 MB). Die folgende Software ist auf der CD vorhanden:

- QtDesigner
- KDevelop
- Qtopia Desktop
- Virtueller Frame-Buffer
- Programme im Ipk-Format für den Zaurus
- Cross-Compiler

Wird der Inhalt dieser CD mit Hilfe des Assistenten installiert, funktioniert *QtDesigner* und *KDevelop*. Es ist aber kein entscheidener Unterschied zu den Versionen, die auf den SuSE-CD's vorhanden sind, zu erkennen.

Allerdings war es nicht möglich, Software zu kompilieren. Die Dokumentation auf der CD ist sehr kurz. Auch die Recherche im Internet löste dieses Problem nicht.

iPaq und Zaurus Anleitung

Mit der Anleitung *IPAQ and Zaurus Development using QPE* von Werner Schulte [Sch02] war es dann möglich, ein Hallo-World Programm für die Text-Konsole (Terminal) zu kompilieren. Dafür mußten Shellscripte verändert werden. Allerdings auch das Herunterladen und Testen weiterer Entwicklungstools aus dem Internet lösten die Probleme nicht. LIRC konnte nicht für den Sharp Zaurus SL-5500G kompiliert werden. Die Entwicklung einer grafischen Ausgabe war nicht möglich.

www.zauruszone.com/howtos

Bei der weiteren Recherche im Internet waren auf dieser Homepage [Zau03b] ausführliche Informationen und lauffähige Entwicklungstools zu finden. In den folgenden Kapiteln sind diese Entwicklungstools erklärt.

6.3.2 Cross-Compilers

Die folgenden RPM-Pakete können bei [Zau03b] heruntergeladen werden:

- gcc-cross-sa1100-2.95.2-0.i386.rpm
  (gcc-Kompiler für ARM-Architektur) 5,0 MB
- binutilis-cross-arm-2.11.2-0.i386.rpm
  (Dienstprogramme für ARM-Architektur) 1,8 MB
- glibc-arm-2.2.2-0.i386.rpm
  (GNU C Bibliotheken für ARM-Architektur) 16,9 MB

- linux-header-arm-sa1100-2.4.6.-3.i386.rpm
  (Linux Header-Dateien für ARM-Architektur) 1,2 MB

Diese Pakete werden folgendermaßen installiert:

```
rpm -Uvh gcc-cross-sa1100-2.95.2-0.i386.rpm
rpm -Uvh binutilis-cross-arm-2.11.2-0.i386.rpm
rpm -Uvh glibc-arm-2.2.2-0.i386.rpm
rpm -Uvh linux-header-arm-sa1100-2.4.6.-3.i386.rpm
```

Den Inhalt dieser Pakete ist dann im Verzeichnis */opt/Embedix/* zu finden.

6.3.3 QPE – Qtopia

Qtopia ist die Bezeichnung für die grafische Oberfläche des Sharp Zaurus. (QPE ist eine alte Bezeichnung.) Damit für diese Oberfläche Anwendungen entwickelt werden können, stellt die Firma Trolltech Entwicklungstools, incl. eines *virtuellen Frame-Buffers*, zu Verfügung. Angeboten werden die Qtopia-Entwicklungstools als kostenlose Version für GPL oder als kostenpflichtige Commercial Edition für kommerzielle Anwendungen. Da die Entwicklungen in dieser Diplomarbeit als freie, kostenlose Programme zur Verfügung gestellt werden, wird die kostenlose Version verwendet. Diese kann bei [Tro03] oder bei [Zau03b] heruntergeladen werden (22 MB) und mit dem folgenden Befehl in das Verzeichnis */opt/Qtopia/* installiert werden:

```
rpm -Uvh qtopia-free-1.5.0-1.i386.rpm
```

Qtopia-Software wird mit der Programmiersprache C++ und der Qt/Embedded-Bibliothek (der Firma Trolltech) entwickelt. Diese Entwicklungen können unter X11 mit Hilfe des *virtuellen Frame-Buffers* betrachtet und getestet werden. Anschließend ist es möglich, diese Entwicklungen für den Sharp Zaurus SL-5500G zu kompilieren. Diese Programme sind, nachdem sie auf den PDA kopiert wurden, sofort ausführbar.

6.3.4 Shellscript dev-x86-qpe.sh

Dieses Shellscript [Zau03b] kann im Homeverzeichnis gespeichert werden. Es dient dazu, die Umgebungsvariablen (environment variables) so zu verändern, dass die Softwareentwicklung für den x86-Rechner kompiliert werden und dann im *virtuellen Frame-Buffer* angeschaut und getestet werden können.

```
if [ -z ${ORG_PATH} ]
then
ORG_PATH=${PATH}
export ORG_PATH
fi

if [ -z ${ORG_LD_LIBRARY_PATH} ]
```

```
then
ORG_LD_LIBRARY_PATH=${LD_LIBRARY_PATH}
export ORG_LD_LIBRARY_PATH
fi

CROSSCOMPILE=/opt/Embedix/tools
QPEDIR=/opt/Qtopia
QTDIR=/opt/Qtopia
PATH=$QTDIR/bin:$QPEDIR/bin:${ORG_PATH}:/opt/Embedix/tools/bin
TMAKEPATH=/opt/Qtopia/tmake/lib/qws/linux-x86-g++/
LD_LIBRARY_PATH=$QTDIR/lib:${ORG_LD_LIBRARY_PATH}
export QPEDIR QTDIR PATH TMAKEPATH LD_LIBRARY_PATH PS1
echo "Altered environment for Sharp Zaurus Development x86"
```

Das Shellscript wird mit dem folgenden Befehl ausgeführt:

```
source /root/dev-x86-qpe.sh
```

6.3.5 Shellscript dev-arm-qpe.sh

Dieses Shellscript [Zau03b] kann auch im Homeverzeichnis gespeichert werden. Es dient dazu, die Umgebungsvariablen so zu verändern, dass die Softwareentwicklung für die ARM-Achitektur des Sharp Zaurus SL-5500G kompiliert wird. Dieses Programm kann mit Hilfe einer SD-Karte auf den Zaurus kopiert werden und ist dann ausführbar.

```
if [ -z ${ORG_PATH} ]
then
ORG_PATH=${PATH}
export ORG_PATH
fi

if [ -z ${ORG_LD_LIBRARY_PATH} ]
then
ORG_LD_LIBRARY_PATH=${LD_LIBRARY_PATH}
export ORG_LD_LIBRARY_PATH
fi

CROSSCOMPILE=/opt/Embedix/tools
QPEDIR=/opt/Qtopia/sharp
QTDIR=/opt/Qtopia/sharp
PATH=$QTDIR/bin:$QPEDIR/bin:$CROSSCOMPILE/bin:${ORG_PATH}
TMAKEPATH=/opt/Qtopia/tmake/lib/qws/linux-sharp-g++/
LD_LIBRARY_PATH=$QTDIR/lib:${ORG_LD_LIBRARY_PATH}
export QPEDIR QTDIR PATH LD_LIBRARY_PATH TMAKEPATH PS1
echo "Altered environment for Sharp Zaurus Development ARM"
```

Das Shellscript wird mit dem folgenden Befehl ausgeführt:

```
source /root/dev-arm-qpe.sh
```

6.4 Kompilieren des Zaurus-Kernels

6.4.1 Installieren der Kernelsources

LIRC benötigt zur Konfiguration und Installation den Kernelsource-Code incl. der vollständigen Header-Dateien. Heruntergeladen werden können die vollständigen Zaurus-Kernelsources (ca. 22 MB) bei [Zau03b]. Ist der Kernel fehlerfrei kompiliert, sind alle benötigten Dateien vorhanden.

Die Entwicklungsumgebung für die ARM-Architektur wird durch diesen Befehl gestartet:

```
source /root/dev-arm-qpe.sh
```

Das Archiv wird durch die folgenden Befehle im Verzeichnis */opt/arm-linux* entpackt:

```
cd /opt/arm-linux
bunzip2 linux-sl5500-20020910.tar.bz2
tar -xf linux-sl5500-20020910.tar
```

6.4.2 Konfigurieren des Kernels

Durch die folgenden Befehle wird in dem automatisch erzeugten *linux*-Verzeichnis die menügeführte Konfiguration des Kernels gestartet:

```
cd linux
make menuconfig
```

Die Konfiguration für den Sharp Zaurus SL-5500G ist in den Kernelsources gespeichert. Diese sollten beim Menüpunkt *Load an Alternate Configuration File* durch Eingabe dieser Zeile geladen werden.

```
arch/arm/def-configs/sl5500
```

Nachdem alle Änderungen in der Kernelkonfiguration (z.B. neue Module aufnehmen) vorgenommen wurden, kann diese mit *Exit* verlassen werden. Auf die Frage *Do you wish to save your kernel configuration?* muß mit *Yes* geantwortet werden [Zau03b].

6.4.3 Kompilieren des Kernels

Nachdem die Konfiguration des Kernels abgeschlossen ist, wird der Kernel mit den folgenden Befehlen kompiliert:

```
make dep && make clean && make zImage
```

Der fertig kompilierte, neue Kernel wird im Verzeichnis *arch/arm/boot* als *zImage* gespeichert. Dieser kann mit Hilfe einer CF-Karte auf den Sharp Zaurus SL-5500G überspielt werden.

Falls Module zu den neuen Kernel hinzugefügt werden sollen, werden diese mit dem folgenden Befehl kompiliert:

```
make modules
```

Diese müssen dann in das Zaurus-Filesystem kopiert werden [Zau03b].

6.4.4 Installieren des Kernels

Das Installieren eines neuen Kernels erfolgt beim Sharp Zaurus SL-5500G mit einer CF-Karte.

1. Das Image des Kernels *zImage* muß ins Hauptverzeichnis (*root-Directory*) der CF-Karte kopiert werden. In diesem Verzeichnis darf sich ausschließlich diese Datei befinden.

2. Nachdem die CF-Karte im Zaurus plaziert ist, muß der Zaurus mit Hilfe des Ladegerätes (ohne Docking Station) mit Spannung versorgt werden. Die orangefarbene LED muß leuchten.

3. Das Batteriefach muß geöffnet werden. Allerdings darf die Batterie nicht entfernt werden.

4. Während die Taste *C* und die Taste *D* gedrückt gehalten werden, muß der *FULL RESET* Knopf am Batteriefach gedrückt werden.

5. Sobald dies erfolgt ist, leuchtet die grüne LED (Mail-Symbol) und die orangefarbene LED (Spannungsversorgung). Dieser Vorgang darf nicht unterbrochen werden, solange beide LED's leuchten.

6. Nach ca. 3 Minuten leuchten nicht mehr beide LED's und das ROM-Update war erfolgreich.

7. Die CF-Karte kann entfernt werden und der *FULL RESET* Knopf betätigt werden.

8. Nachdem das Batteriefach geschlossen wurde, muß die Verriegelung wieder auf *NORMAL OPERATION* gestellt werden.

9. Wenn der *On/Off* Knopf betätigt wird, bootet der Zaurus mit dem neuen Kernel.

Eine ausführliche Anleitung ist bei [Sha02] erhältlich. In dieser Anleitung sind einige Warnhinweise enthalten, die unbedingt zu beachten sind!

6.5 LIRC für den Zaurus

6.5.1 Konfigurieren von LIRC

Nach dem fehlerfreien Kompilieren des Zaurus-Kernels kann jetzt auch LIRC (Kapitel 4 ab Seite 20), incl. dem Kernelmodul *lirc_sir*, für den Zaurus kompiliert werden. Allerdings muß zuerst die Entwicklungsumgebung für die ARM-Architektur durch den folgenden Befehl gestartet werden:

```
source /root/dev-arm-qpe.sh
```

Das LIRC-Paket kann mit diesem Befehl entpackt werden:

```
cd lirc_arm
tar -xzf lirc-0.6.6.tar.gz
```

In dem neuen Unterverzeichnis befindet sich das dialog-basierte Shellscript `configure`. Damit ist es möglich die passende Hardware auszuwählen.

```
cd lirc-0.6.6
./configure
```

Der Sharp Zaurus SL-5500G kann bei dem Menüpunkt *" 1 Driver configuration "* unter Punkt *" 7 PDAs "* ausgewählt werden. Eine Datei, bzw. das Shellscript `configure.sh` wird mit dem Menüpunkt *" 3 Save configuration & exit "* gespeichert. Dieses sollte folgendermaßen aussehen, bzw. entsprechend modifiziert werden:

```
./configure \
--with-moduledir=/tmp/modules/2.4.6-rmk1-np2-embedix/misc \
--without-x \
--with-driver=sa1100 \
--with-major=61 \
--with-port=none \
--with-irq=none \
--with-kerneldir=/opt/arm-linux/linux \
--host=arm-linux \
```

Die Anweisungen bedeuten folgendes:

with-moduledir:
: In dieses Verzeichnis wird das Kernelmodul *lirc_sir* geschrieben.

without-x:
Keine X-Windows Anwendung (z.B. xmode2) übersetzen.

with-driver=sa1100:
Prozessor Intel StrongARM SA1100

with-kerneldir:
In diesem Verzeichnis sind die Zaurus-Kernelsources zu finden.

Dieses Shellscript kann mit diesem Befehl gestartet werden:

```
./configure.sh
```

6.5.2 Kompilieren und installieren von LIRC

Nach einer fehlerfreien Konfiguration können das Kernelmodul und die Programme mit diesen Befehlen kompiliert und installiert werden:

```
make
make install
```

Die folgende Software kann z.B. mit Hilfe einer SD-Karte kopiert und auf dem Sharp Zaurus SL-5500G installiert werden:

- /tmp/modules/2.4.6-rmk1-np2-embedix/misc/lirc_sir.o
- daemons/lircd
- tools/irw
- tools/mode2
- tools/rc

6.5.3 Installieren von LIRC auf dem Zaurus

Das Kernelmodul *lirc_sir.o* muß bei dem Zaurus in das Verzeichnis */lib/modules/2.4.6-rmk1-np2-embedix/misc* kopiert und geladen werden.

```
cp /mnt/card/lirc/lirc_sir.o \
   /lib/modules/2.4.6-rmk1-np2-embedix/misc/lirc_sir.o
insmod lirc_sir
```

Mit `lsmod` läßt sich anzeigen, welche Module zur Zeit im Kernel geladen sind. Durch den Befehl `rmmod lirc_sir` kann das Kernelmodul wieder entfernt werden.

Mit `mknod /dev/lirc c 61 0` wird das 'character device' angelegt. Diese Gerätedatei wird für die Übertragung von Daten vom Kernelmodul zu den Programmen benötigt. Die Programme, bzw. der LIRC-Daemon können ins Verzeichnis */usr/local/bin*, die Konfigurationsdatei *lircd.conf* in das Verzeichnis */etc* kopiert werden.

```
mknod /dev/lirc c 61 0
cp /mnt/card/lirc/mode2 /usr/local/bin/mode2
cp /mnt/card/lirc/zmode2 /usr/local/bin/zmode2
cp /mnt/card/lirc/rc /usr/local/bin/rc
cp /mnt/card/lirc/irw /usr/local/bin/irw
cp /mnt/card/lirc/lircd /usr/local/bin/lircd
cp /mnt/card/lirc/lircd.conf /etc/lircd.conf
```

6.6 Kompilieren von eigenen Entwicklungen für den Zaurus

6.6.1 pro-File

Die Datei *zmode2.pro* dient als Beispiel für den Aufbau eines pro-Files. Die vollständige Beschreibung der Softwareentwicklung ”zmode2” folgt im Kapitel 7.

```
TEMPLATE     = app
CONFIG       = qt warn_on release
HEADERS      = zmode2.h
SOURCES      = zmode2.cpp
INCLUDEPATH += $(QPEDIR)/include
DEPENDPATH  += $(QPEDIR)/include
LIBS        += -lqpe -lqte
TARGET       = zmode2
```

HEADERS: In dieser Zeile werden die *Header*-Dateien der Softwareentwicklung angegeben.

SOURCE: In dieser Zeile werden die *C++* Dateien festgelegt. Es können auch mehrere Dateien, durch Leertaste getrennt, eingetragen werden.

LIB: Durch diesen Eintrag werden die Bibliotheken */opt/Qtopia/sharp/lib/libqpe.so* und */opt/Qtopia/sharp/lib/libqte.so* eingebunden. Sind diese Bibliotheken nicht angegeben, funktioniert zwar das Kompilieren (g++), allerdings ist das Linken (gcc) nicht möglich.

TARGET: Hier wird der Name der ausführbaren Datei angegeben.

6.6.2 tmake

Makefiles für Software-Projekte können mit dem Tool `tmake` von der Firma Trolltech erzeugt werden. Da verschiedene Plattformen oft unterschiedliche Compiler verwenden, sollten die jeweiligen *Makefiles* automatisch erstellt werden. Diese werden durch die folgenden Befehlsketten erzeugt:

```
tmake zmode2.pro > Makefile
```

oder

```
tmake -o Makefile zmode2.pro
```

In diesem *Makefile* sind die Informationen für den Befehl

```
make
```

enthalten, damit der Quellcode kompiliert und gelinkt werden kann und eine ausführbare Datei entsteht.

- Wurde die Entwicklungsumgebung für die ARM-Architektur (Zaurus) mit den Befehl `source /root/dev-arm-qpe.sh` gestartet, kann die so entstandene Datei kann mit Hilfe einer SD-Karte auf den Sharp Zaurus SL-5500G kopiert werden. Diese Anwendung ist sofort ausführbar und auf dem Display erkennbar.
- Wurde die x86-Entwicklungsumgebung des Sharp Zaurus mit den Befehl `source /root/dev-x86-qpe.sh` gestartet, kann das fertig kompilierte Programm im *virtuellen Frame-Buffer* auf dem x86-Rechner angeschaut und getestet werden.

  ```
  qvfb &
  ./zmode2 -qws
  ```

Hinweis: Wird von einer zur anderen Entwicklungsumgebung gewechselt, muß der *Makefile* und die automatisch durch `make` erzeugten Dateien gelöscht werden. Dies wird durch diese Befehle erreicht:

```
make clean
rm Makefile
```

7 Softwareentwicklung: ”zmode2”

7.1 Ziel

Das Programm ”xmode2” verwendet die X11-Bibliotheken. Diese Bibliotheken sind für den Sharp Zaurus SL-5500G nicht verfügbar. Deshalb wurde ein Programm für diesen PDA entwickelt, mit dem die Puls- und Pausenlänge von Infrarot-Signal grafisch dargestellt werden kann, das an der internen IR-Schnittstelle empfangen wird.

So soll es mit dem Programm ”zmode2” möglich sein, sowohl das verwendete Protokoll (Kapitel 3.4 ab Seite 8), als auch die Befehle zu erkennen, die von einer Infrarot-Fernbedienung gesendet werden. Ebenfalls ist es dann möglich, fehlende oder fehlerhaft übertragene Bits zu lokalisieren.

7.2 Qt/Embedded

Als Grundlage dienen das Kernelmodul ”lirc_sir” und die Programme ”mode2” und ”xmode2” aus dem LIRC-Projekt [LP02]. Da ”xmode2” die X11-Bibliotheken verwendet, die beim Zaurus nicht zur Verfügung stehen, mußte ein neues Programm entwickelt werden. Die grafische Benutzeroberfläche (GUI) [1] von ”zmode2” ist mit Qt programmiert.

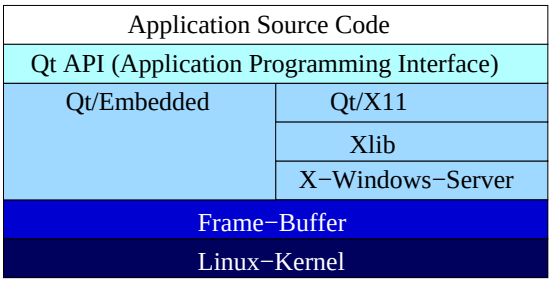


Abbildung 7.2.1: Vergleich Qt/Embedded und QT/X11

Qt ist eine C++ Bibliothek, die von norwegische Firma Trolltech [Tro03] für nicht kommerzielle Anwendungen frei zur Verfügung gestellt wurde. Speziell für kleine,

[1] Graphical User Interface

tragbare Geräte, wie z.B. PDA’s, ist die speichersparende und schnelle QT/Embedded-Bibliothek entwickelt worden [Tro02].

In Abbildung 7.2.1 ist zu erkennen, dass bei Qt/Embedded direkt auf den Frame-Buffer zugegriffen wird. Bei Qt/X11 sind zwei weitere Schichten erforderlich, die Xlib-Bibliothek und der X-Windows-Server.

7.2.1 Event-Verarbeitung

Zu beachten ist bei der Qt-Programmierung, dass Ereignisse, die Auswirkungen auf die Anwendung haben, als Events bezeichnet werden. Dazu zählen unter anderem Timer-, Maus-, Tastatur- und Paint-Events. Diese werden in einer Warteschlange, der sogenannten Event-Queue, gespeichert. Diese Event-Queue wird von der Haupt-Event-Schleife des Objekts der Klasse QPEApplication abgefragt [Leh99]. In Abbildung 7.2.2 ist die Event-Verarbeitung dargestellt.

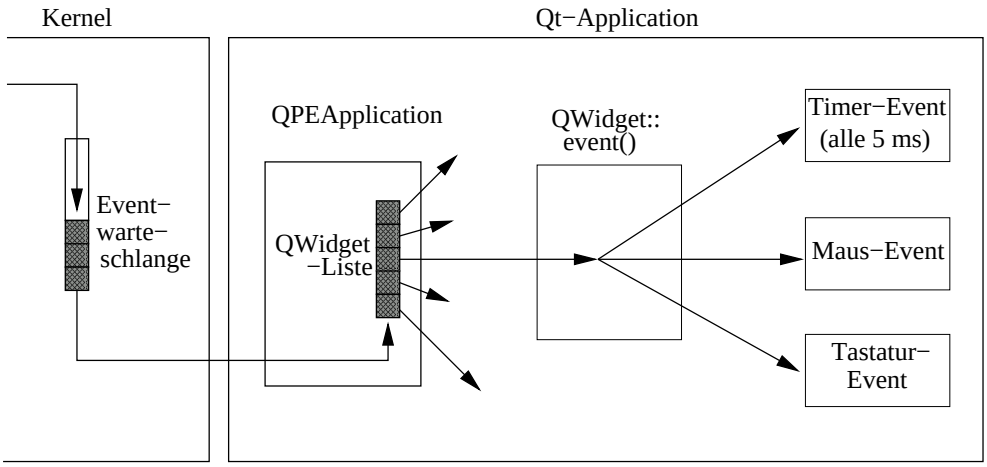


Abbildung 7.2.2: Event-Verarbeitung

In dem Programm ”zmode2” wird das Auslesen jedes Wertes von der Infrarot-Schnittstelle mit Hilfe eines Timer-Events gestartet. Dies ist erforderlich, da ansonsten keine Benutzerbefehle verarbeitet werden können.

Es wird nach jeweils 5 ms überprüft, ob ein neuer Wert an der Gerätedatei (*/dev/lirc*) anliegt. Wenn das der Fall ist, wird dieser Wert ausgewertet und grafisch ausgegeben. Wenn Ereignisse in der Event-Queue warten, wie z.B. Tastendruck zur Zeitänderung oder Maus-Event zum Beenden des Programms, werden diese verarbeitet.

7.2.2 Signal/Slot-Konzept

Die Firma Trolltech hat das Signal/Slot-Konzept entwickelt. In jeder von *QObject* abgeleiteten Klasse kann man neue Methoden als Signal oder Slot definieren. Die Signale

senden dabei Nachrichten aus einem Objekt und die Slots empfangen diese Nachrichten. Über die Methode connect kann dann ein Signal mit einem Slot zur Laufzeit verbunden werden [Leh99]. Im Konstruktor der Klasse Zmode2 sind die folgenden Zeilen vorhanden:

```
timer = new QTimer(this);
connect( timer, SIGNAL(timeout()), SLOT(ReadPaint()) );
timer->start(5);
```

In der ersten Zeile wird ein Objekt `timer` von der Klasse *QTimer* erzeugt. Dann wird `timer` mit dem Slot `ReadPaint()` verbunden. `timer` wird mit dem Wert 5 gestartet, dies bedeutet, das nach jeweils 5 ms ein Timeout erreicht wird. Dieses Timer-Event ruft damit den Slot `ReadPaint()` auf.

Der Wert von 5 ms ist in empirischen Untersuchungen festgelegt worden. Wird dieser Wert reduziert, werden nicht mehr alle Benutzerbefehle verarbeitet.

7.2.3 Steuervariable zmode

zmode (hex)	Bedeutung
xxxxxxxF	device (/dev/lirc) geöffnet
xxxxxxx0	device geschlossen
xxxxxx1x	Zeit geändert
xxxxxx0x	Zeitanzeige geändert
xxxxFxxx	Help durch Tastatur-Event angefordert
xxxxExxx	Help (noch) aktiv
xxxx0xxx	Help beendet

Tabelle 7.1: Steuervariable zmode

Um auf die unterschiedlichen Events reagieren zu können, wird zusätzlich die Steuervariable `zmode` benötigt. In Tabelle 7.1 sind die unterschiedlichen Betriebszustände definiert. So ’erkennt’ das Programm, wenn ein IR-Signal empfangen wird, ob z.B. die Hilfe noch aktiv ist. Falls das der Fall ist, muß der Bildschirm gelöscht werden und in der linken oberen Ecke der empfangende Wert gezeichnet werden. Die Variable zmode ist als Integer deklariert und ist im Konstruktor mit den Wert 0x00FFF010 initialisiert.

7.3 Methoden des Programms zmode2

Das Programm ist in verschiedenen Methoden (Unterprogramme des Objektes) der Klasse *Zmode2* aufgeteilt. Die Klasse *Zmode2* ist von *QWidget* abgeleitet. In den folgenden Kapiteln sind die wichtigsten Methoden erklärt. Der vollständige Quellcode ist im Anhang ab Seite 64 abgedruckt.

7.3.1 void help(void)

Die Methode `void Zmode2::help(void)` dient als online-Hilfe. Diese Hilfe wird bei jedem Programmstart angezeigt und endet, sobald ein Infrarot-Signal empfangen wird. Diese Hilfe ist jederzeit mit dem Key_Up aufrufbar. Durch dieses Tastatur-Event wird auch die Steuervariable gemäß Tabelle 7.1 auf Seite 47 verändert.

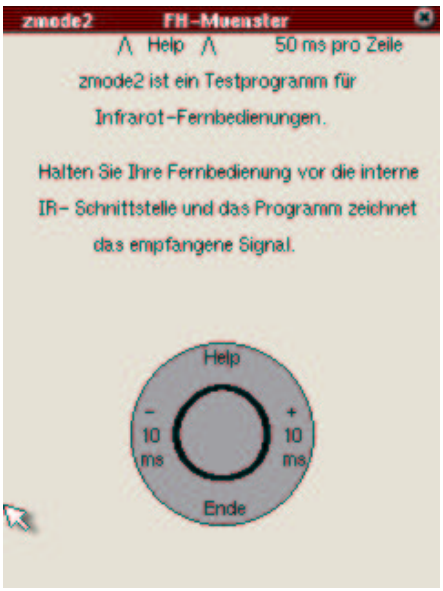


Abbildung 7.3.1: Methode void help(void)

7.3.2 void openDevice(void)

Die Methode `void Zmode2::openDevice()` öffnet die Gerätedatei (*/dev/lirc*), falls die letzten vier Bits der Steuervariable nicht gesetzt sind. Das Öffnen wird mit dem Befehl `fd=open(device,O_NONBLOCK)` erreicht. Dabei bekommt der Filedeskriptor (`fd`) nur dann einen negativen Wert, wenn sich die Gerätedatei nicht öffnen läßt. Die symbolische Konstante `O_NONBLOCK` wird verwendet, um das Blockieren beim Lesen oder Schreiben in einer Gerätedatei zu verhindern [Her99]. Dies war erforderlich, da beim Warten auf einen neuen Wert aus der Gerätedatei sich das Programm ”zmode2” weder bedienen noch beenden ließ, wenn keine Werte vorhanden waren.

Falls beim vorherigen Benutzen das Programm beim Lesen und Zeichnen beendet wurde, sind manchmal noch alte Werte in der Gerätedatei gespeichert. Damit diese nicht mehr relevanten Werte nicht gezeichnet werden, werden nach erfolgreichem Öffnen noch solange die Werte ausgelesen, bis kein alter Wert mehr vorhanden ist. Dies wird durch die folgenden Quellcode-Zeilen erreicht:

```
int result = 0;
while (result!=-1){
```

```
    result=read(fd,&data,sizeof(data));
  }
```

Diese Schleife wird solange durchlaufen, bis `result` den Wert -1 hat. Dies ist aber nur der Fall, wenn keine Werte in der Gerätedatei vorhanden sind.

7.3.3 void ReadPaint()

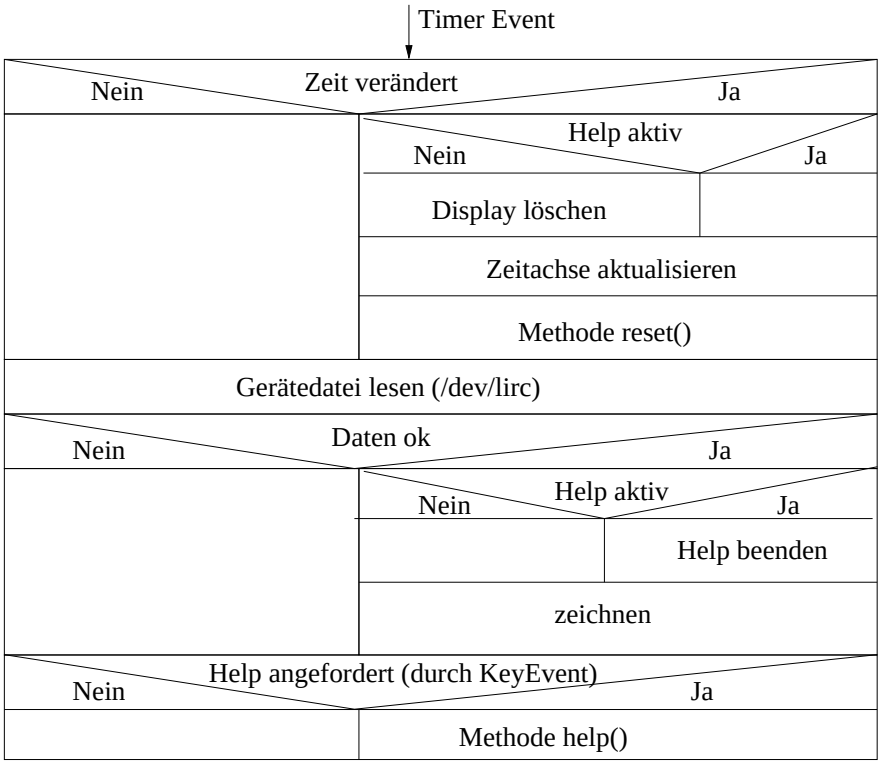


Abbildung 7.3.2: Methode void ReadPaint()

Der Aufbau des Slots `void Zmode2::ReadPaint()` ist im Flußdiagramm (Abbildung 7.3.2) zu erkennen. `ReadPaint()` wird automatisch vom Timer im Abstand von jeweils 5 ms aufgerufen. Zuerst wird der Wert der Steuervariable überprüft um festzustellen, ob durch einen Tastatur-Event der Zeitwert verändert wurde. Falls dies der Fall ist, wird der obere Displayteil gelöscht und der aktuelle Zeitraum pro Zeile angezeigt. Das entsprechende Bit der Steuervariable wird zurückgesetzt und die Methode `reset()` aufgerufen.

Nachdem mit dem Befehl `result=read(fd,&data,sizeof(data))` der Wert gelesen wurde, wird durch die Abfrage `if(result==sizeof(data))` erkannt, dass die Daten in Ordnung sind und dieser Wert gezeichnet werden kann. Falls `result` allerdings den Wert -1 hat, wurde kein IR-Signal empfangen. Bevor gezeichnet werden kann, muß die evtl. noch aktive Help-Einblendung beendet werden.

Mit `y=(data&PULSE_BIT)?yp:ys` wird festgelegt, ob der aktuelle Wert Highpegel, also Pulse, oder Lowpegel (Space) hat.

`x0=(unsigned long)(data&PULSE_MASK)` übergibt die Länge des Pulses, bzw. der Pause.

Dieser Zeitraum wird durch die Gleichung `x2=(int)(x2+x0/(zeit*4.08))` skaliert, wobei `zeit` der Wert ist, der in der oberen Zeile angezeigt wird und einer Zeilenlänge entspricht. Die Konstante 4.08 dient als Korrekturfaktor, damit der angezeigte Zeitraum pro Zeile der tatsächlichen Zeit entspricht.

7.3.4 void keyPressEvent(QKeyEvent *event)

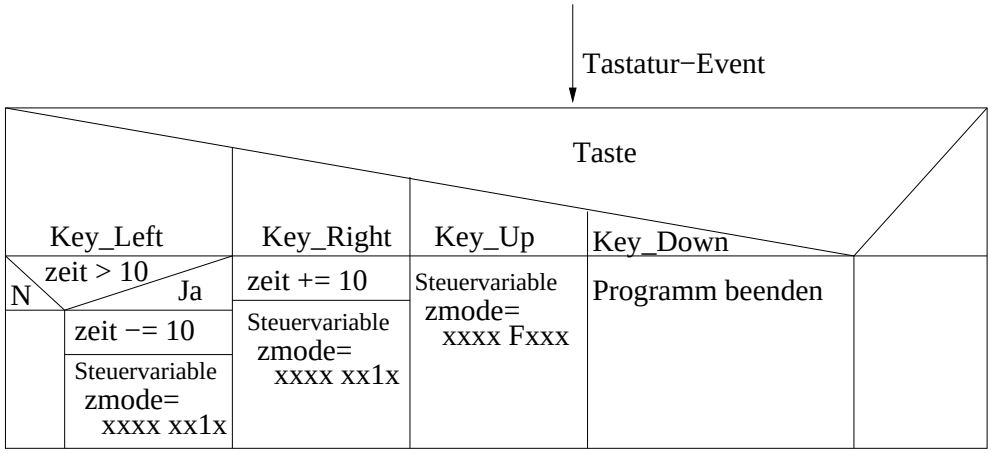


Abbildung 7.3.3: Tastatur-Event

Bei jedem Tastatur-Event wird diese Methode aufgerufen und dient zur Steuerung von ”zmode2”. Dies ist mit einer switch-Anweisung realisiert. Wenn Key_Left gedrückt wird, reduziert sich der Wert von `zeit` um 10 ms. Um `zeit` um 10 ms zu erhöhen, muß Key_Right gedrückt werden. In beiden Fällen verändert sich die Steuervariable gemäß Tabelle 7.1 auf Seite 47. Das Programm ”zmode2” läßt sich mit Key_Down oder der Cancel-Taste beenden.

7.3.5 Sonstige Methoden

void reset(void) In der Methode `void Zmode2::reset(void)` der Klasse Zmode2 werden die x- und y-Variablen initialisiert. Diese Methode wird im Konstruktor und bei jeder Zeitänderung aufgerufen. Die x-Variablen repräsentieren die Signalbreite (Zeit), die y-Variablen die Signalhöhe (Amplitude).

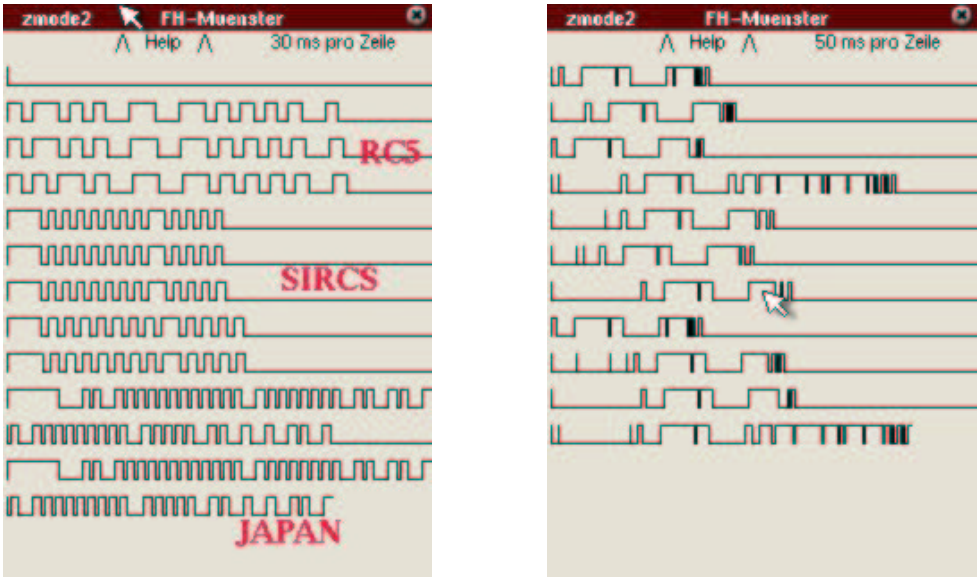
void closeDevice(void) Geschlossen wird die Gerätedatei (*/dev/lirc*) wird durch die Methode `void Zmode2::closeDecive(void)`.

void mouseReleaseEvent(QMouseEvent *e) Bei jedem Maus-Event wird die Methode `void Zmode2::mouseReleaseEvent(QMouseEvent *e)` gestartet.

7.4 Test der Software

7.4.1 Fernbedienungs-Protokolle

Die verschiedenen Fernbedienungs-Protokolle lassen sich mit dem Programm ”zmode2” überprüfen. In Abbildung 7.4.1 sind Screenshots von verschiedenen Infrarot-Fernbedienungen dargestellt.



verschiedene Fernbedienungs-Protokolle IrDA-Übertragung

Abbildung 7.4.1: Screenshots Datenübertragung

7.4.2 IrDA

Mit dem Programm ”zmode2” können nicht nur IR-Fernbedienungssignale, sondern auch andere Infrarot-Datenübertragungsprotokolle grafisch dargestellt werden. So kann auch eine IrDA-Übertragung [2], die bei Taschenrechnern, Mobiltelefonen, PDA’s und Laptops angewendet wird, überprüft werden. In Abbildung 7.4.1 ist ein Auszug aus einer IrDA-Übertragung dargestellt.

[2] Infrared Data Association

7.4.3 Positive Ergänzungen

Das Programm ”zmode2” wurde von einigen Personen getestet. Dabei wurden folgende Verbessungsvorschläge genannt:

- Signalhöhe (Amplitude) vergrößern
- Änderungsmöglichkeit der Startzeit
- Änderungsoption der Hintergrundfarbe
- Speichern und Aufrufen einer Konfigurationsdatei
- Zoomen eines Signalausschnitts

7.4.4 Realisierungsmöglichkeiten der Ergänzungen

Diese Vorschläge können mit geringen Aufwand programmiert werden, wurden aber aus Zeitmangel nicht realisiert. In diesem Abschnitt werden Realisierungsmöglichkeiten aufgelistet, wie sich die Ergänzungen im Programm ”zmode2” programmieren lassen.

Signalhöhe: Die Signalhöhe (Amplitude) beträgt 10 Pixel (siehe Methode ReadPaint()). Diese könnte variabel gestalten werden, so dass der Benutzer diesen Wert verändern kann.

Startzeit: Der Startzeitwert ist im Konstruktor initialisiert und liegt bei 50 ms pro Zeile. Für die meisten IR-Fernbedienungsprotokolle hat sich der Startwert von 50 ms als optimal herausgestellt.

Hintergrundfarbe: Die Hintergrundfarbe ist weiß und die Schriftfarbe schwarz. Aus ergonomischen Gründen ist es vielfach angenehmer, wenn ein schwarzer Hintergrund und die Schriftfarbe weiß gewählt wird. Diese Stellen sind im Quelltext auskommentiert und können bei Bedarf ins Programm aufgenommen werden. Allerdings ist die Berechnung der einzelnen Pixel aufwendiger und zeitintensiver [Leh99].

Konfigurationsdatei Diese Features können in der nächsten Version entweder beim Starten mit Parametern übergeben oder mit Hilfe einer Konfigurationsdatei gespeichert werden. Die Konfigurationsdatei würde beim Programmstart automatisch geladen.

Zoomen eines Signalausschnitts Aus Untersuchungszwecken sollte die Möglichkeit integriert sein, ein empfangenes IR-Signal zoomen zu können, z.B. durch Speichern der Einzelwerte. Die Werte könnten mit einer anderen Zeitachse und Amplitude noch einmal gezeichnet werden.

Das Zoomen läßt sich auch mit der Klasse *QScrollView* realisieren. Die Klasse ist speziell für eine größere Darstellung eines verschiebbaren Ausschnitts des Fensters vorgesehen [Leh99].

8 Zusammenfassung und Ausblick

Zusammenfassung

Der Sharp Zaurus SL-5500G dient als Testgerät für Infrarot-Fernbedienungen. Nach einer erfolgreichen Installation der Entwicklungstools ist das Kompilieren von Programmen aus dem LIRC-Projekt möglich. Die interne IrDA-Schnittstelle wird vom Kernelmodul *lirc_sir* unterstützt. Dadurch wird keine zusätzliche Hardware für den PDA verwendet.

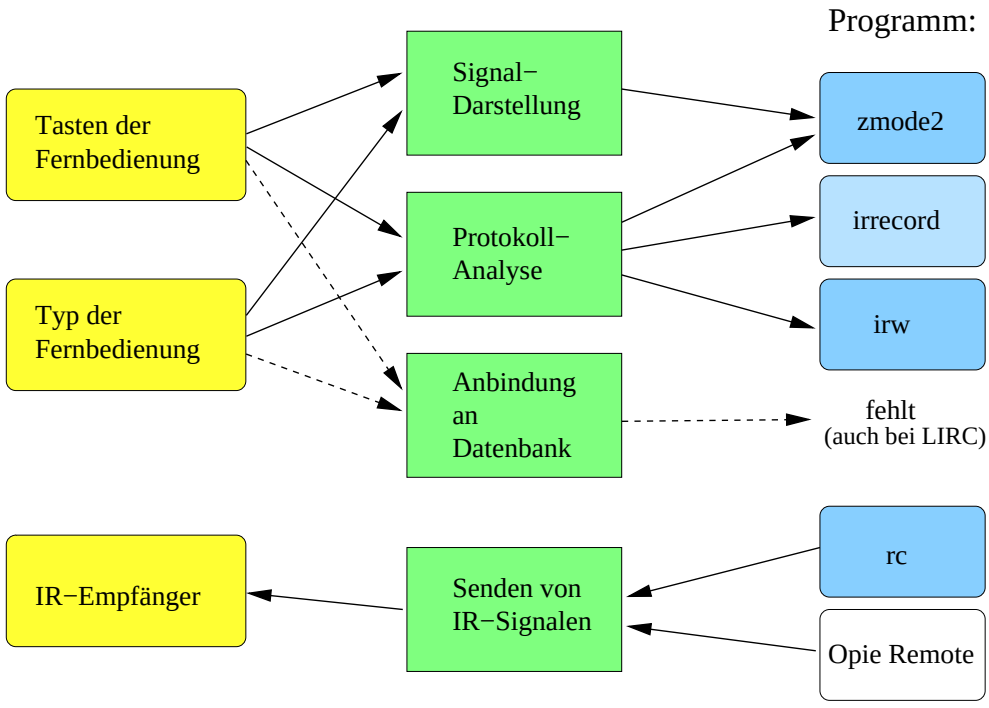


Abbildung 8.0.1: Aufgaben der Programme

Das empfangene Infrarot-Signal wird mit Hilfe der Softwareentwicklung "**zmode2**" als Puls- und Pausenlänge grafisch auf dem Display dargestellt. Somit wird die Funktion der Fernbedienung überprüft. Außerdem ist die Erkennung und Analyse der gesendeten Protokolle und die Feststellung fehlerhafter Übertragungen möglich.

Das Programm "**irrecord**", welches nicht auf dem Sharp Zaurus funktioniert, ermöglicht das Anlernen von unbekannten Fernbedienungen. Dabei werden die Befehle der Fernbedienungen, die der Linux-PC empfängt, analysiert und die Software erkennt das verwendete Protokoll. Nachdem alle Tasten angelernt sind, wird eine Konfigurationsdatei *lircd.conf* erzeugt.

Mit Hilfe des Programms "**irw**" wird der Name der Taste und der Name der Fernbedienung, wie diese in der Konfigurationsdatei gespeichert sind, angezeigt.

Das Programm "**rc**" ermöglicht das Senden von IR-Signalen. Dadurch ist die Überprüfung von IR-Empfängern mit einer maximalen Entfernung von 4 Metern möglich.

Ausblick

In einer nachfolgenden Diplomarbeit sollten die Befehle der unterschiedlichen Fernbedienungen nicht in einer Konfigurationsdatei, sondern direkt in einer **Datenbank** gespeichert werden. Eine Datenbank bietet folgende Vorteile:

- Aufnahme der Protokolle und Befehle einer großen Anzahl von Fernbedienungen
- Zuordung gleicher IR-Signalfolgen zu unterschiedlichen Fernbedienungen
- automatischer Nachweis, welcher Hersteller gleiche Protokolle und Befehle verwendet

Das Programm "**irrecord**" sollte modifiziert werden, damit es auf dem Sharp Zaurus SL-5500G ausführbar ist. Hierdurch entfällt das Anlernen unbekannter Fernbedienungen am Linux-PC. Diese neuen Fernbedienungen könnten automatisch in der Datenbank aufgenommen bzw. mit vorhandenen verglichen werden.

Mit einer grafischen Oberfläche ausgestattet, sollte die gesamte Software mit einem **Hauptprogramm** gestartet werden können. Dadurch müssen die einzelnen Programme nicht durch Eingeben der Befehle in der Textkonsole (Terminal) gestartet werden.

Ebenfalls könnte das Programm "**Opie Remote**" [Spi02] zum Senden von IR-Signalen verwendet werden. Dieses Programm besitzt eine grafische Oberfläche. Das Senden von IR-Befehlen funktioniert durch Berühren eines Feldes (Taste der Fernbedienung) auf dem Display.

Zusätzlich sollten die positiven Ergänzungen bei der Softwareentwicklung "**zmode2**" (Kapitel 7.4.4 auf Seite 52) realisiert werden.

9 Anhang

9.1 Fernbedienungs-Protokolle

9.1.1 RC5-Code (Tabellen)

Adresse	Gerät	Adresse	Gerät
00	TV1	0E	CD-Photo
01	TV2	0F	Reserviert
02	Videotext	10	Audio-Vorverstärker
03	Erweiterung (TV1, TV2)	11	Tuner
04	Laser Vision Player	12	Anal. Kassettenrekorder
05	Videorecorder (VCR1)	13	Audio-Vorverstärker
06	Videorecorder (VCR2)	14	CD
07	Reserviert	15	Audio-Rack o. Aufn.gerät
08	SAT1	16	Audio Satellitenempfänger
09	Erweiterung (VCR1, VCR2)	17	DCC-Rekorder
0A	SAT2	18	Reserviert
0B	Reserviert	19	Reserviert
0C	CD-Video	1A	Beschreibbare CD
0D	Reserviert	1B...1F	Reserviert

Tabelle 9.1: Systemadressen (hex) und Geräte des RC5-Codes [Woh02]

Befehl	Bedeutung	Befehl	Bedeutung
00	0	12	Brightness +
01	1	13	Brightness -
02	2	14	Color +
03	3	15	Color -
04	4	16	Bass +
05	5	17	Bass -
06	6	18	Treble +
07	7	19	Treble -
08	8	1A	Balance right
09	9	1B	Balance left
10	Volume +		
11	Volume -		

Tabelle 9.2: Gemeinsame Befehle (hex) aller Adressen des RC5-Codes [Woh02]

9.1.2 SIRCS- bzw. CNTRL-S-Code (Tabellen)

Systemadresse	Gerät
01	TV
02	VTR1
04	VTR2
06	Laserdisk
07	VTR2
0B	VTR3
0C	Surround Sound Processor
10	Cassettendeck, Tuner
11	CD Player
12	Equalizer
A4	TV digital effects (8 Bit Systemadresse)

Tabelle 9.3: Systemadressen (hex) und Geräte des SIRCS-Codes [Woh02]

Befehl	Bedeutung	Befehl	Bedeutung
00	1	12	Volume +
01	2	13	Volume -
02	3	1A	Color +
03	4	1B	Color -
04	5	1E	Brightness +
05	6	1F	Brightness -
06	7	25	Balance right
07	8	26	Balance left
08	9	70	Treble +
09	0	71	Treble -
10	Channel +	72	Bass +
11	Channel -	73	Bass -

Tabelle 9.4: Gemeinsame Befehle (hex) aller Adressen des SIRCS-Codes [Woh02]

9.2 lircd.conf

Hier ist eine Konfigurationsdatei *lircd.conf* dargestellt. In dieser Datei sind vier unterschiedliche Fernbedienungen aufgenommen. Der Philips VCR und TV verwenden den RC5-Code. Der Technics CD-Player benutzt den JAPAN-Code und der Grundig TV den MOTOROLA-Code. Das Erkennen des MOTOROLA-Codes hat mit dem Programm ”irrecord” nicht funktioniert, deshalb sind die Puls- und Pausenlängen mikrosekundengenau gespeichert (vgl. Kapitel 3.4.7 ab Seite 16 - Startnachricht).

```
################################################################
# brand:                       /tmp/krautstrunk/philipsVCR.conf
# model no. of remote control:
# devices being controlled by this remote:
#  Philips Videorecorder FH Muenster

begin remote

  name  PhilipsVCR
  bits           13
  flags RC5|CONST_LENGTH
  eps            30
  aeps          100

  one           950   819
  zero          950   819
  plead         956
  gap          113378
  min_repeat      3
  toggle_bit      2

      begin codes
          record                   0x0000000000001177
          standby                  0x000000000000114C
          select                   0x000000000000114B
          timer                    0x000000000000115D
          clear                    0x0000000000001171
          1                        0x0000000000001141
          2                        0x0000000000001142
          3                        0x0000000000001143
          4                        0x0000000000001144
          5                        0x0000000000001145
          6                        0x0000000000001146
          7                        0x0000000000001147
          8                        0x0000000000001148
          9                        0x0000000000001149
          0                        0x0000000000001140
          monitor                  0x000000000000017A
          ok                       0x0000000000000157
          minus                    0x0000000000001161
          plus                     0x0000000000001160
          play                     0x0000000000001175
          zurueck                  0x0000000000001165
          vor                      0x0000000000001166
          stop                     0x0000000000001176
          index                    0x0000000000000170
          sp_lp                    0x000000000000117A
          still                    0x0000000000001169
          system                   0x000000000000014A
      end codes

end remote
```

```
############################################################
# brand:                       TechnicsCD
# model no. of remote control:
# devices being controlled by this remote:
# Technics CD Player

begin remote

  name  TechnicsCD
  bits           16
  flags SPACE_ENC
  eps            30
  aeps          100

  header       3569  1663
  one           514  1228
  zero          514   357
  ptrail        516
  pre_data_bits   32
  pre_data       0x40040550
  gap          74933
  toggle_bit      0


      begin codes
          open                     0x00000000000080D5
          stop                     0x0000000000000055
          pause                    0x0000000000006035
          play                     0x0000000000005005
          zuruck                   0x0000000000004015
          vor                      0x000000000000C095
          davor                    0x00000000000092C7
          naechste                 0x0000000000005207
          1                        0x000000000000085D
          2                        0x00000000000088DD
          3                        0x000000000000481D
          4                        0x000000000000C89D
          5                        0x000000000000287D
          6                        0x000000000000A8FD
          7                        0x000000000000683D
          8                        0x000000000000E8BD
          9                        0x000000000000184D
          10                       0x000000000000A1F4
          groesser10               0x0000000000002174
          0                        0x00000000000098CD
          program                  0x0000000000005104
          clear                    0x0000000000000154
          recall                   0x00000000000081D4
          time_mode                0x000000000000AAFF
          ab_repeat                0x0000000000001247
          repeat                   0x000000000000E2B7
          random                   0x000000000000B2E7
          auto_cue                 0x000000000000D184
          leiser                   0x00000000000084D1
          lauter                   0x0000000000000451
          tape_length              0x000000000000D580
          side_ab                  0x0000000000003560
          peak_search              0x000000000000F1A4
      end codes

end remote


############################################################
```

```
#
# brand:                       /tmp/krautstrunk/philipsTV.conf
# model no. of remote control:
# devices being controlled by this remote:
# Philips Fernseher FH Muenster
# irrecord funktionierte mit der Austausch FB nicht richtig.
# Habe zum Erkennen des Codes (RC-5) die VCR-FB genommen,
# die Tasten erlernen, natürlich mit der  TV-FB.

begin remote

  name  PhilipsTV
  bits           13
  flags RC5|CONST_LENGTH
  eps            30
  aeps          100

  one           956   813
  zero          956   813
  plead         975
  gap          113390
  min_repeat      2
  toggle_bit      0


      begin codes
          1                        0x0000000000001801
          2                        0x0000000000001002
          3                        0x0000000000001803
          4                        0x0000000000001004
          5                        0x0000000000001805
          6                        0x0000000000001006
          7                        0x0000000000001807
          8                        0x0000000000001008
          9                        0x0000000000001809
          0                        0x0000000000001000
          10                       0x000000000000180A
          mute                     0x000000000000100D
          vt                       0x000000000000183C
          vtaus                    0x000000000000103F
          lauter                   0x0000000000001810
          leiser                   0x0000000000001011
          plus                     0x0000000000001820
          minus                    0x0000000000001021
          aus                      0x000000000000180C
      end codes

end remote


#################################################################
#
# brand:                       /tmp/krautstrunk/grundigTV_4.conf
# model no. of remote control:
# devices being controlled by this remote:
# Grundig Fernbedienung FH Muenster

begin remote

  name   GrundigTV
  flags RAW_CODES
  eps            30
  aeps          100

  ptrail          0
```

```
repeat     0     0
gap    122520

    begin raw_codes

        name 1
            606    2597     582     484     609     460
            582     485     609     458     584     484
            608     459     610     459     608     471
            597     458     584   20757     608    2593
            583     484     607     995     580     484
            608     459    1142     996     578     484
            610     457     608     459     608

        name 2
            581    2622     582     483     609     460
            582     484     584     484     583     484
            585     483     583     484     609     461
            607     457     584   20754     584    2619
            582    1019    1115    1022     578     483
           1143     995     579     484     608     459
            608     458     583

        name 3
            558    2645     556     510     558     510
            557     510     557     510     558     510
            557     511     558     508     558     509
            558     511     556   20781     557    2642
            558     509     558     509     557    1044
            557     509    1090    1044     556     510
            557     509     557     509     558

        name 4
            556    2647     556     510     558     510
            557     511     556     510     558     510
            557     511     556     511     557     510
            557     511     556   20783     555    2645
            556    1044     556     509    1091    1044
           1089    1044     555     510     557     510
            557     509     557

        name 5
            608    2594     608     459     609     458
            609     458     609     459     609     458
            609     458     610     458     608     459
            608     459     609   20728     609    2593
            607     459     608     994    1139     995
           1138     994     606     458     609     457
            609     458     608

        name 6
            584    2618     559     509     584     484
            583     484     583     484     583     484
            584     484     582     485     583     484
            583     484     584   20752     557    2645
            555    1045    1113     484     583    1020
           1113    1018     582     485     581     484
            583     483     584

        name 7
            609    2594     634     434     609     458
            635     433     634     434     634     433
            609     458     634     434     609     459
            633     434     634   20710     635    2568
            607     459     608     458     610     457
```

```
            633     970    1164     970     630     434
            609     458     633     434     608

        name 8
            609    2594     610     457     610     458
            609     459     609     458     609     458
            610     458     609     459     634     432
            609     459     613   20726     609    2593
            607     995     605     458     609     458
           1142     458     608     996     630     432
            610     457     609     458     608

        name 9
            608    2597     606     460     631     437
            606     461     607     461     606     461
            607     460     606     462     606     462
            605     462     606   20733     607    2596
            604     461     606     996     604     461
           1139     462     604     997     604     461
            606     461     605     461     605

        name 0
            607    2598     605     462     606     462
            605     463     605     462     606     461
            606     462     606     462     605     462
            606     461     606   20734     606    2595
            605     996     604     461     606     461
            607     460    1139     996     603     461
            606     461     605     462     605

    end raw_codes

end remote
```

9.3 Quellcode ”zmode2”

9.3.1 zmode2.h

```
#ifndef ZMODE2_HEADER
#define ZMODE2_HEADER
#include <qslider.h>
#include <qwidget.h>
#include <qpainter.h>
#include <sys/time.h>
#include "lirc.h"

class Zmode2 : public QWidget {
  Q_OBJECT

  public:
    Zmode2( QWidget *parent = 0, const char *name = 0, WFlags f = 0 );//Konst.
    \~Zmode2 ();
    char *device;         //  /dev/lirc
    int fd;               // Filedeskriptor
    int zmode;            // Steuerung des Programms
    lirc_t data;          // Daten aus /dev/lirc
    int zeit;             // ms pro Zeile
    int max_y;            // Displayhöhe
    int max_x;            // Displaybreite
    int y;                // ausgelesener y Wert
    int yp;               // y Pulse
    int ys;               // y Space
    int x0;               // ausgelesener x Wert
    int x1;               // alter x Wert
    int x2;               // neuer x Wert
    QTimer *timer;        // alle 5 ms Zeichen einlesen
    void reset(void);     // alle Variaben .........
    void stop(void);
    void openDevice(void);  // oeffnet /dev/lirc
    void closeDevice(void); // schliesst /dev/lirc
    void help(void);      // Hilfe

  public slots:
    void ReadPaint();      // Zeichen lesen und zeichnen

  signals:
    void clicked();       // beendet die Anwendung

  protected:
    void SetDeviceName(char *s);
    void mouseReleaseEvent(QMouseEvent *);  // Mausbedienung
    void keyPressEvent(QKeyEvent *event);   // Tastaturbedienung
} ;

#endif
```

9.3.2 zmode2.cpp

```
#ifdef HAVE_CONFIG_H
# include <config.h>
#endif

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <getopt.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <qpe/qpeapplication.h>
#include <qwidget.h>
#include <qpainter.h>
#include <qmessagebox.h>
#include <qtimer.h>
#include "zmode2.h"


//***********************************************************************
Zmode2::Zmode2(QWidget *parent, const char *name, WFlags f)
: QWidget(parent, name, f) {
  zeit = 50;              // Startzeitwert 50ms
  zmode = 0x00FFF010;     // 0  0  ... ... help ... zeit open
  max_y = 280;            // 320-30 Pixel sichtbare Displayhoehe
  max_x = 238;            // 240-2 Pixel nutzbare Displaybreite
  resize (max_x, max_y);
  timer = new QTimer(this);
  connect( timer, SIGNAL(timeout()), SLOT(animate()) );
  timer->start(5);  // alle 5ms
  reset();
  openDevice();
}
//***************************************************************************
Zmode2::~Zmode2 () {
  //close(fd);
  exit(EXIT_SUCCESS);      //Terminalfenster wieder frei, OHNE close(fd)
}
//***************************************************************************
void Zmode2::help(void) {
  if ((zmode&0x0000F000)==0x0000F000)    {   // HELP    durch KeyEvent
    zmode = zmode - 0x00001000  ;            // zmode = xxxxExxx (help noch aktiv)
    QPainter p (this);
    p.eraseRect(0,20,max_x,max_y);           // löscht unteren Bildschirm
    p.setBrush(QBrush(gray));
    p.drawEllipse (70,170,100,100);
    //     p.setBrush(QBrush(black));
    p.setPen(QPen(Qt::black, 3, Qt::SolidLine));
    p.drawEllipse (95,195,50,50);
    p.setPen(QPen(Qt::black, 1, Qt::SolidLine));
    p.drawText(111,182,"Help");
    p.drawText(111,265,"Ende");
    p.drawText( 72,200,20,40,AlignCenter|WordBreak,"- 10 ms");
    p.drawText(153,200,15,40,AlignCenter|WordBreak,"+ 10 ms");
    //     p.setPen(QPen(Qt::white, 1, Qt::SolidLine));
    p.drawText(30,30, "    zmode2 ist ein Testprogramm für");
    p.drawText(30,50, "       Infrarot-Fernbedienungen.");
    p.drawText(20,80, "Halten Sie Ihre Fernbedienung vor die interne");
    p.drawText(20,100,"IR- Schnittstelle und das Programm zeichnet ");
    p.drawText(20,120,"          das empfangene Signal.");
```

```
    p.flush();
    repaint(FALSE);
    sleep(1);
    start();
  }
}
//*****************************************************************************
void Zmode2::ReadPaint(void)  {
  QPainter p (this);
  //p.setPen(QPen(Qt::white, 1, Qt::SolidLine));
  if (zmode&0x000000F0) {             // Zeit veraendert        !=0
    if (!(zmode&0x0000F000))  {       // nur, wenn nicht Help   ==0
     p.eraseRect(0,0,max_x,max_y);    // loescht Display
    }
    p.eraseRect(0,0,max_x,20);        // loescht oberes Display
    p.drawText(60,10, " /\\   Help   /\\ ");
    p.drawText(150,10,QString().setNum(zeit) +" ms pro Zeile");
    zmode = zmode - (zmode&0x000000F0); // xxxxxx0x  Zeit geändert
    reset();                          // Variablen initialisieren
  }
  int result;
  result=read(fd,&data,sizeof(data));
    printf("\n result = %i \n",result);
  if(result==sizeof(data)){               // Daten o.k.
    if ((zmode&0x0000F000)==0x0000E000) { // Help noch aktiv
      zmode = zmode - 0x0000E000;         // xxxx0xxx Hilfe beendet
      p.eraseRect(0,20,max_x,max_y);      // loescht unteren Bildschirm
      reset();                            //  ??????????ß
    }
    y = (data&PULSE_BIT) ? yp:ys ;          // aktueller y Wert
    x0 =(unsigned long) (data&PULSE_MASK);  // aktueller x Wert (Laenge des Zeichens)
          // p.setPen(QPen(Qt::white, 1, Qt::SolidLine));
    x1=x2;
    x2= (int)(x2+x0/(zeit*4.08));       // Länge des Zeichens (skaliert)
    p.drawLine ( x1, y, x2, y);         // Pulse bzw. Space zeichnen
    p.drawLine ( x1, yp, x1,ys);        // senkrechte Linien Pulse-Space
    if (x2 >= (max_x)) {                // Zeilenende
      x2 = 2;
      yp = yp + 20;                     // Amplitude Pulse neue Zeile
      ys = yp + 10;                     // Amplitude Space neue Zeile
    }
    if (ys >= (max_y)) {                // Bildschirmende
       yp = 20;                         // Amplitude Pulse
       ys = 30;                         // Amplitude Space
       y = 0;
       p.eraseRect(0,20,max_x,max_y);   // loescht unteren Bildschirm
    }
  }
  p.flush();
  repaint (FALSE);
  if ((zmode&0x0000F000)!=0)  help();     // Help durch KeyEvent
}
//*****************************************************************************
void Zmode2::SetDeviceName(char *s){
   device= s;
}
//*****************************************************************************
void Zmode2::openDevice( )  {
  if (!(zmode&0x0000000F)) {     // /dev/lirc noch nicht geoeffnet
    unsigned long mode;
    struct stat s;
    SetDeviceName("/dev/lirc");
    //    fd=open(device,O_RDONLY);
    fd=open(device,O_NONBLOCK);
    if(fd==-1)  {
```

```
      QMessageBox::critical(this,"zmode2",QString("Error opening /dev/lirc"));
     exit(EXIT_FAILURE);
   };
    if ( (fstat(fd,&s)!=-1) && (S_ISFIFO(s.st_mode)) ) { // ist device eine Pipe?
   // can't do ioctls on a pipe
   }
    else if(ioctl(fd,LIRC_GET_REC_MODE,&mode)==-1 || mode!=LIRC_MODE_MODE2) {
      QMessageBox::critical(this,"zmode2",
                                QString("zmode2 only pulse/space layer"));
      close(fd);
      exit(EXIT_FAILURE);
    }
    sleep(1);         //warten, bis device wirklich fertig geöffnet ist
    zmode = zmode + 0x0000000F;   //  device ist geoeffnet!

     // Der Zaurus hatte oft noch alte Werte im device, deshalb werden hier
     // die alten Werte ausgelesen, aber nicht ausgewertet.
    int result = 0;
    while (result!=-1){
      result=read(fd,&data,sizeof(data));
    }
  }
}
//*************************************************************************

void Zmode2::mouseReleaseEvent(QMouseEvent *e) {

  if ( rect().contains( e->pos() ) ){
   //  emit clicked();
  }
}

//*********************************************************************
void Zmode2::closeDevice( )  {
  zmode = zmode - 0x0000000F;
  close(fd);
}
//*********************Tastatur Behandlungsroutinen*********************

void Zmode2::keyPressEvent(QKeyEvent *event)
{
  QWidget::keyPressEvent(event);
  switch (event->key()) {
  case Key_Left:
        if (zeit>10){
          zeit -= 10;
          zmode = zmode + 0x00000010;
        }
        break;
  case Key_Right:
        zeit += 10;
        zmode = zmode + 0x00000010;
        break;
  case Key_Up:
        zmode = (zmode&0x0000F000)? zmode : (zmode+0x0000F000);
        break;
  case Key_Down:
  case Key_Escape:
        exit(EXIT_SUCCESS);
        break;
  }

}

//***************** Ende Tastatur Behandlungsroutinen ******************
```

```
void Zmode2::start(void){
 x1 = 0;          // alte x Position
 x2 = 2;          // neue x Position
 x0 = 0;          // ausgelesener Wert (Laenge)
 yp = 20;         // Amplitude Pulse
 ys = 30;         // Amplitude Space
 y = 0;
}
//************************************************************************
```

9.3.3 Mainprogramm

Datei: zmode2.cpp

```
//************************************************************************
int main(int argc,char **argv)
{
  QPEApplication app(argc, argv);
  Zmode2 w;     //ein Objekt von Klasse Zmode2 erzeugen
  //   w.setBackgroundColor(Qt::black);       // ohne Farben
  w.setCaption("  zmode2             FH-Muenster");
  QObject::connect(&w,SIGNAL(clicked()),&app,SLOT(quit()) );
  app.setMainWidget (&w);
  w.showMaximized();
  //    w.show();
  return app.exec();//return(EXIT_SUCCESS);
}
//************************************************************************
```

Literaturverzeichnis

[Bar00] BARTELMUS, CHRISTOPH: *Ein Pinguin sieht (infra)rot*. c't magazin für computer technik, 18, 2000.

[Eit01] EITLE COMPUTER: *RS-232 — Die serielle Schnittstelle*. Internet: http://www.eitle-computer.at/computer/pchw/rs232/rs232.htm, 2001.

[Fae02] FAERBER, NILS: *Sharp Zaurus SL-500D im Test. Außen hui!* Linux-Magazin, 02, 2002.

[FD02] FAERBER, NILS und MIRKO DÖLLE: *Gmate Yopy YP-3000 im Test. Was lange währt ...* Linux-Magazin, 07, 2002.

[Fis02] FISCHER, DIRK: *Impulstechnik - Vorlesung*. Fachhochschule Münster, Fachbereich Elektrotechnik, 2002.

[Fre91] FREE SOFTWARE FOUNDATION: *GNU General Public License*. Internet: http://www.gnu.org/copyleft/gpl.html, Juni 1991.

[Gma00] GMATE: *Homepage*. Internet: http://www.gmate.com, Januar 2000.

[Her99] HEROLD, HELMUT: *Linux - Unix Systemprogrammierung*. Eddision-Wesley-Longman, Bonn, 1999.

[Jan02] JANSSEN, MICHAEL: *Infrarottechnik – Das Spektrum*. Internet: http://www.mijan.de, 2002.

[JK99] JANKE, THORSTEN und MARKUS KOPPERS: *Optimierung und Implementierung eines Systems zur Unterdrückung von Werbeblöcken bei Aufzeichnungen mit dem Videorecorder.* Diplomarbeit, Fachhochschule Münster, Fachbereich Elektrotechnik, September 1999.

[jre02] JREUTER: *Verbindung von Zaurus SL5500 mit SuSE Linux*. Internet: http://sdb.suse.de/de/sdb/html/jreuter_zaurus_usb.html, Juni 2002.

[Leh99] LEHNER, BURKHARD: *KDE- und Qt-Programmierung. GUI-Entwicklung für Linux*. Eddision-Wesley-Longman, München, 1999.

[LP02] LIRC-PROJECT: *Homepage*. Internet: http://www.lirc.org, 2002.

[Ric02a] RICHERT, PETER: *Kommunikationssysteme - Praktikum*. Internet: http://www.ktet.fh-muenster.de, 2002.

[Ric02b] RICHERT, PETER: *Kommunikationssysteme - Vorlesung*. Internet: http://www.ktet.fh-muenster.de, 2002.

[Rua02] RUAULT, CHARLES-EDOUARD: *Patch für SuSE 8.0-2.4.18 – usbd-net*. Internet: http://www.ruault.com/Zaurus/patches/usbdnet-SuSE-8.0-2.4.18.patch.gz, Juni 2002.

[Sch00] SCHREIBER, KARSTEN: *Pinguin ferngesteuert*. Linux-Magazin, 09, 2000.

[Sch02] SCHULTE, WERNER: *IPAQ and Zaurus Development using OPE*. Internet: http://www.uv-ac.de/ipaqhelp/pdf.tar.gz, September 2002.

[Sha02] SHARP: *ROM Doownload - Entwickler-CD für Zaurus SL-5500G*. Internet: http://www.zaurus.de, August 2002.

[Spi02] SPIRALMAN, THOMAS: *Opie Remote*. Internet: http://www.rit.edu/ tfs1812/, Juni 2002.

[Tro02] TROLLTECH: *Qt/Embedded Whitepaper*. Internet: http://www.trolltech.com, Januar 2002.

[Tro03] TROLLTECH: *Homepage*. Internet: http://www.trolltech.com, Januar 2003.

[Vis01] VISHAY TELEFUNKEN: *TSOP17.. Data Manual*. Internet:http://www.vishay.com, April 2001.

[Woh02] WOHLRABE, FRANK: *INFRAROT-Datenübertragung*. Elektor-Verlag GmbH, Aachen, 2002.

[wP01] PROJECT WINLIRC: *Homepage*. Internet: http://winlircsourceforce.net, Juli 2001.

[Xac02] XACT DISTRIBUTOR: *Homepage*. Internet: http://www.yopy.at, November 2002.

[Yop03] YOPY-ENTWICKLERPLATTFORM: *Englischsprachige Homepage*. Internet: http://www.yopydeveloper.org, Januar 2003.

[Zau03a] ZAURUS-ENTWICKLERPLATTFORM: *Englischsprachige Homepage*. Internet: http://developer.sharpsec.com, Januar 2003.

[Zau03b] ZAURUS-ENTWICKLERPLATTFORM: *Englischsprachige Homepage*. Internet: http://www.zauruszone.com/howtos, Januar 2003.

[Zau03c] ZAURUS-FORMUM: *Homepage*. Internet: http://zaurus.help4free.de, Januar 2003.

[Zau03d] ZAURUS-FORMUM: *Homepage*. Internet: http://merlinto.boerde.de, Januar 2003.

[Zau03e] ZAURUS-FORMUM: *Homepage*. Internet: http://zaurus.loveslinux.com, Januar 2003.