Diplomarbeit

Dario Borchers

Untersuchung der Störsicherheit von Entfernungsmessungen mit Ultraschall auf einem Risc Controller Board mit Embedded Linux

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Dario Borchers

Untersuchung der Störsicherheit von Entfernungsmessungen mit Ultraschall auf einem Risc Controller Board mit Embedded Linux

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Informations- und Elektrotechnik
Studienrichtung Informationstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.rer.nat. Jochen Schneider
Zweitgutachter : Prof. Dr. Thomas Lehmann

Abgegeben am 12. April 2012

Dario Borchers

Thema der Diplomarbeit

Untersuchung der Störsicherheit von Entfernungsmessungen mit Ultraschall auf einem Risc Controller Board mit Embedded Linux

Stichworte

Ultraschall, BeagleBoard-xM, Netbeans, Eclipse, Software, Embedded Linux, Abstandsmessungen, Touchdisplay, Netzwerk.

Kurzzusammenfassung

In Dieser Arbeit wird ein BeagleBoard-xM als vollwertiger Linux-Rechner im Netzwerk der HAW Hamburg in Betrieb genommen. Anschließend werden verschiedene Möglichkeiten der Softwareentwicklung evaluiert. Anschließend werden drei Sensoren an diesem Entwicklungsboard angeschlossen und Software für deren Auswertung erstellt. Diese Software wird systematisch gestört, und beobachtet, wie stark die Messwerte in diesem Falle abweichen.

Dario Borchers

Title of the paper

Analysis of the interference immunity of distance measurements using ultrasonic sound on a risc controller board with embedded Linux.

Keywords

Ultrasonic sound, BeagleBoard-xM, Netbeans, Eclipse, Software, Embedded Linux, Distance measurements, Touchdisplay, Network.

Abstract

In this paper, a BeagleBoard-xM is being putted into operation as a fully operational Linux Pc in the Network of the University of applied Sciences Hamburg. Then, different methods of developing software for this Board are evaluated. Then, three different sensors have been connected to the board. Software for interpreting the data has been written. This software is has been systematically disturbed, to analyse the interference immunity of the measurements.

Inhalt

Abkürzungsverzeichnis

BIOS	Basic Input and Output System
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DFT	Discrete Fourier Transformation
DNS	Domain Name System
DRM	Digital Rights Management
DSP	Digital Signal Processor
ECC	Error Checking and Correction
FAT	File Allocation Table
FFT	Fast Fourier Transformation
FIR	Finite Impulse Response
FTP	File Transfer Protocol
GCC	Gnu Compiler Collection
GDB	Gnu Debugger
GID	Group ID Number
GNU	Gnu's not Unix
GPIO	General Purpose Input and Output
GPU	Graphics Processing Unit
HAW	Hochschule für Angewandte Wissenschaften
HDMI	High Definition Multimedia Interface
HID	Human Interface Device
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
I²C, IIC	Inter-Integrated Circuit Bus
IIR	Infinite Impulse Response
IP	Internet Protocol
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LSB	Least Significant Bit
NFS	Network File System
NTSC	National Television Systems Comitee
MAC	Media Access Control
MaC	Multiply and Accumulate
MBR	Master Boot Record
MCU	Micro Controller Unit
MSAD	Microsoft Active Directory

MSB	Most Significant Bit
OSI	Open Systems Interconnection
PAL	Phase Alternation Line
PC	Personal Computer
PE	Portable Executable
RAM	Random Access Memory
ROM	Read Only Memory
RSE	Remote Systems Explorer
SCM	system Control Module
SCP	Secure Copy
SD-Card	Secure Digital Memory Card
SFR	Special Function Register
SMB	Server Block Message Protocol
SoC	System on a Chip
SRAM	Static Random Access Memory
SSH	Secure Shell
SSHFS	Secure Shell File System
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TI	Firma Texas Instruments
TPM	Trusted Platform Computing
TTL	Transistor-Transistor-Logic
UDP	Universal Datagram Protocol
UID	User ID Number
USB	Universal Serial Bus
VGA	Video Graphics Array
VM	Virtual Machine
WebDAV	Web based Distributed Authoring and Versioning

1. Hintergrund

In diesem Kapitel wird zum einen der Hintergrund, vor dem diese Arbeit entstanden ist, erläutert, außerdem wird die Problemstellung und das verwendete RISC-Controller-Board erläutert. Danach wird herausgestellt, dass es sich bei dem verwendeten Entwicklungs-Board nicht um ein „kleines“ Entwicklungs-Board handelt, sondern dass man aufgrund des auf dem Board installierten Linux-Betriebssystems die Systemsicherheit nicht vernachlässigen darf.

1.1 Hintergrund und Problemstellung

Die Hochschule für angewandte Wissenschaften Hamburg verfügt über eine Reihe von BeagleBoard-xM’s. Diese sollen im Labor für Digitale Informationstechnik für Laborversuche in dem Fach Betriebssysteme eingesetzt werden. Im Rahmen dieser Arbeit soll ein solches BeagleBoard-xM im Netzwerk der HAW Hamburg in Betrieb genommen werden. Weiter soll geprüft werden, auf welche Art Software für dieses Board entwickelt werden kann. Darüber hinaus soll untersucht werden, wie mit der Kombination von Board und dem darauf ausgeführtem Linux-Betriebssystem einfache Messungen durchgeführt werden können. Dies soll anhand von Entfernungsmessungen mit Ultraschallsensoren erfolgen. Es stehen zwei verschiedene Typen von Ultraschallsensoren zur Verfügung, die an das Board angeschlossen und ausgewertet werden sollen. Schliesslich sollen Rückschlüsse auf das Scheduling des Linux-Kernels gezogen werden, indem versucht wird, die Software, welche diese Messungen durchführt, gezielt zu beeinflussen bzw. zu stören.

1.2 BeagleBoard-xM

Das BeagleBoard-xM ist ein lüfterloser, kostengünstiger Einplatinencomputer für den Ausbildungsbetrieb an Hochschulen. Es wurde von der BeagleBoard.org- und der TI-Gemeinschaft entwickelt und ist außerdem eine Plattform für komplexere Designs. Das Design und das Layout des Boards ist unter einer Open Source Lizenz erhältlich, sodass es für die Allgemeinheit möglich ist, das Design an eigene Bedürfnisse anzupassen, und im Anschluss eigene Boards fertigen zu lassen [1].

Als Hauptbestandteil wird auf dem BeagleBoard-xM ist ein DM3530 System-on-a-Chip (SoC) von der Firma Texas Instruments (TI) verwendet. Dieser SoC ist im Wesentlichen eine Kombination aus Prozessor (CPU) und Graphikprozessor (GPU). Außerdem ist diesem SoC ein TI TMS320C64x+ Signalprozessor (Digital Signal Processor, DSP) integriert. Ein Signalprozessor ist kein „Generalist“ wie andere CPU’s, sondern ist auf die in der digitalen Signalverarbeitung gebräuchlichsten Algorithmen ausgerichtet.

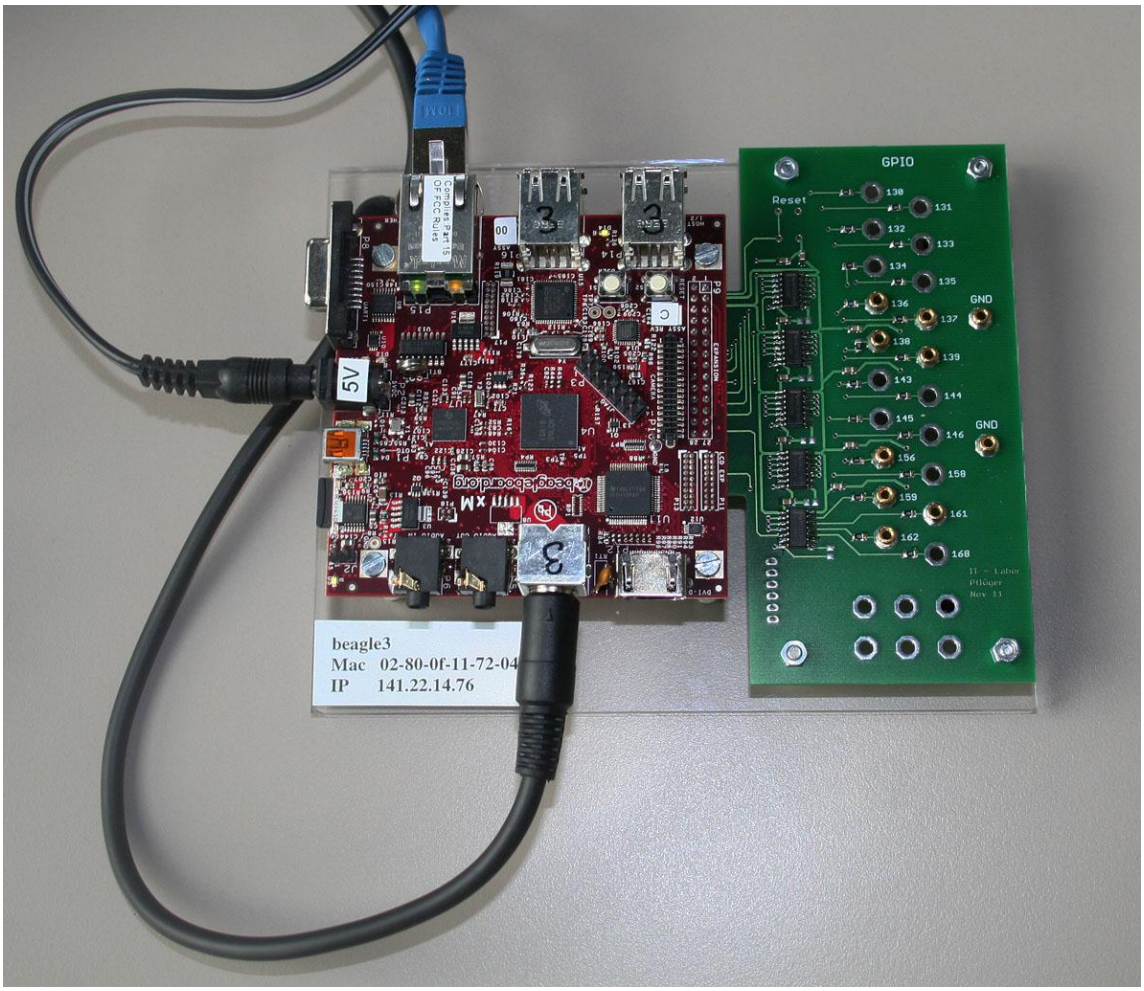

Bild 1: BeagleBoard-xM Rev.C mit Pegelumsetzer im Labor der HAW Hamburg

In der digitalen Signalverarbeitung müssen sehr häufig zwei Gleitkommazahlen miteinander multipliziert und im Anschluss aufsummiert werden (Multiply and Accumulate, MaC-Operation). Typische Algorithmen sind die Diskrete Fourier Transformation (DFT), Finite Impulse Response Filter (FIR-Filter) oder Infinite Impulse Response Filter (IIR-Filter). Genau diese MaC-Operationen können von DSP's in sehr wenigen Machinenzyklen abgearbeitet werden, wogegen die „Generalisten" ein Vielfaches an Zyklen brauchen würden. Weitere Features der DSP's sind u.a. Sättigung von Variablen (eine Variable Läuft nicht über und kippt das Vorzeichenbit, sondern bleibt auf dem höchsten möglichen positiven Wert stehen, bleibt also in der Sättigung) und die Adressierung im Bitreverse-Modus (dies ist bei der schnellen Diskreten Fourier Transformation, der FFT, nützlich). Aufgrund der genannten Optimierungen für die digitale Signalverarbeitung wird der DSP in dieser Arbeit nicht verwendet oder weiter berücksichtigt.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{j2\pi\frac{kn}{N}}$$

Gl. 1: DFT

$$x[n] = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X[k] \cdot e^{j2\pi \frac{kn}{N}}$$ **Gl. 2: inverse DFT**

Als CPU wird ein Coretex-A8 (Codename: Modena) der Firma ARM Ltd. verwendet. Sie implementiert den kompletten ARM-v7 Befehlssatz und verfügt über 32KB Level1-Cache und kann außerdem zwischen 600MHz und 1,0GHz getaktet werden. Für das DRM wurde zusätzlich ein TPM-Modul integriert. Als Level2-Cache stehen 256KB Speicher mit ECC-Option zur Verfügung. Allerdings verfügt diese CPU nicht, wie beispielsweise moderne Prozessoren von AMD oder Intel, über einen in die CPU integrierten Speichercontroller.

Als Schnittstellen stehen auf dem Board unter anderem RS232, USB 2.0 und Fast Ethernet zur Verfügung. Die Grafikausgabe erfolgt wahlweise digital über HDMI oder analog über einen S-Video Ausgang, der zwischen PAL und NTSC Modus umgeschaltet werden kann. Die Audio-Ein- und Ausgabe erfolgt über 3,5mm Klinkenstecker. Außerdem steht ein General Purpose IO-Feld (GPIO) mit 28 Pins zur Verfügung. Hier kann also eigene Peripherie angeschlossen werden, ggf. ist ein Pegelwandler nötig.

Das hier verwendete BeagleBoard-xM taktet die CPU mit 1,0 GHz, und verfügt über 512 MB RAM. Es ist außerdem eine Fassung für Mikro-SD Speicherkarten verbaut, sodass Massenspeicher nicht nur an den USB-Port angeschlossen werden können. Ab Werk ist ein Bootloader in den SoC installiert worden, der in der Lage ist, ein FAT-32 Dateisystem von einer Speicherkarte zu lesen und von dieser dann ein Betriebssystem zu starten.

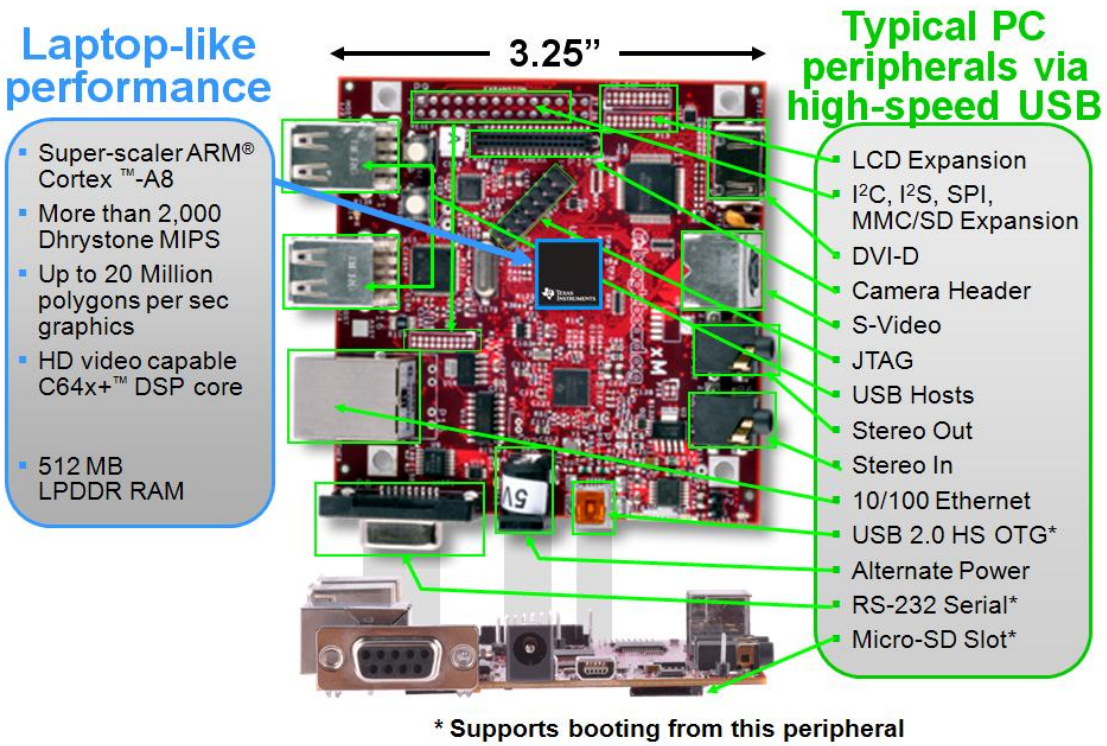

Bild 2: BeagleBoard-xM Werbeflyer [1]

Da der SoC eine ARM Cortex A8 CPU enthält, ist es in Kombination mit den Bootloadern möglich, ein Embedded Linux von einer Speicherkarte zu booten und es dann auf diesem Board laufen zu lassen. Auf diese Weise erhält man einen vollwertigen Linux Rechner, der wahlweise als Client oder eben auch als Server im Netzwerk fungieren kann. Aufgrund der geringen Abmessungen der Leiterplatte und der geringen Stromaufnahme eignet sich die Board-Linux Kombination sehr gut als Thin-Client oder für den mobilen Einsatz.

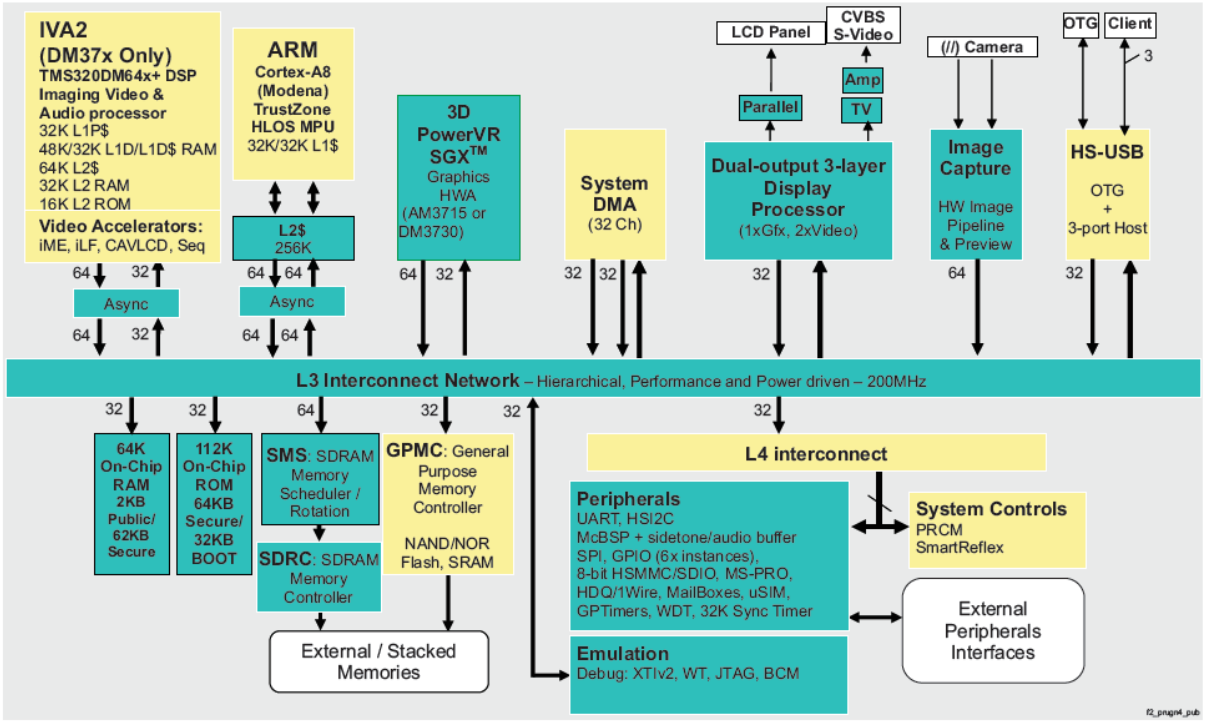

Bild 3: Blockdiagramm des SoC [16]

Es existiert eine Vielzahl von unterschiedlichen Betriebssystemen für ARM Cortex A8 Prozessoren und damit für dieses Board, darunter Microsoft Windows Embedded Compact 7.0, Microsoft Windows Embedded CE 6.0 sowie diverse Linux Distributionen. Da im Ausbildungsbetrieb der HAW Hamburg Debian Linux eingesetzt wird und aufgrund der großen Verbreitung von Debian Linux soll diese Distribution auch für diese Arbeit eingesetzt werden.

Von der Hochschule für angewandte Wissenschaften Hamburg wurde ein BeagleBoard-xM mit einem fertigen Debian-Image auf einer Speicherkarte bereitgestellt. Das Board bootet das auf der SD-Karte vorinstallierte Linux und ist somit betriebsbereit.

1.3 Systemsicherheit

Da es sich bei dem BeagleBoard-xM in Kombination mit Debian Linux um einen vollwertigen Linux-Rechner handelt, welcher mit einer „Laptop-like Performance“ beworben wird (siehe Abbildung 2), darf die Systemsicherheit nicht außer Acht gelassen werden. Erschwerend kommt hinzu, dass das Board wie jedes andere Gerät in das Netzwerk der HAW Hamburg integriert wird und daher eine im Internet routbare IP-Adresse bekommt. Das Board ist also von jedem an das Internet angeschlossenem Rechner direkt erreichbar, was eine Vielzahl von vollautomatischen Angriffsversuchen nach sich zieht.

```
Feb  4 14:05:35 beagle3 sshd[10136]: Invalid user oracle from 216.240.157.176
Feb  4 14:05:39 beagle3 sshd[10138]: Invalid user oracle from 216.240.157.176
Feb  4 14:05:43 beagle3 sshd[10140]: Invalid user oracle from 216.240.157.176
Feb  4 14:05:46 beagle3 sshd[10142]: Invalid user db2inst1 from 216.240.157.176
Feb  4 14:05:53 beagle3 sshd[10146]: Invalid user db2inst1 from 216.240.157.176
Feb  4 14:05:56 beagle3 sshd[10148]: Invalid user max from 216.240.157.176
Feb  4 14:06:00 beagle3 sshd[10150]: Invalid user max from 216.240.157.176
Feb  4 14:06:11 beagle3 sshd[10156]: Invalid user rednet from 216.240.157.176
Feb  4 14:06:14 beagle3 sshd[10158]: Invalid user rednet from 216.240.157.176
```

Listing 1: Auszug aus /var/log/auth.log

Aufgrund dieses Umstandes ist zwingend auf die Vergabe von starken Passwörtern (keine ganzen Wörter, mindestens acht Zeichen, große und kleine Buchstaben sowie Zahlen) zu achten. Des Weiteren müssen Sicherheitsupdates nach deren Veröffentlichung zeitnah eingespielt werden, um die bekannt gewordenen Sicherheitslücken zu schließen. Auch muss darauf geachtet werden, welches Programm und welcher Deamon einen Netzwerk Port öffnet. Sollte eine Netzwerknutzung dieses Programms nicht erforderlich sein, so sollte diese Software entweder umkonfiguriert werden oder aber es muss eine Netzwerkkommunikation unter Zuhilfenahme eines Paketfilters (Firewall) verhindert werden, um die den Angreifern zur Verfügung gestellte Angriffsfläche zu minimieren.

2. Inbetriebnahme des Boards im Netzwerk der HAW Hamburg

In diesem Kapitel werden die Probleme und deren Lösungen bei der Inbetriebnahme des Boards im IT-Netzwerk der HAW-Hamburg erklärt. Es mussten eine vorgegebene IP-Adresse und eine ebenfalls vorgegebene MAC-Adresse eingestellt werden. Außerdem sollte ein Display an den S-Video Ausgang angeschlossen werden. Um diesen aktivieren zu können, musste entscheidend in den Bootprozess des Betriebssystems eingegriffen werden, daher wird auch dieser hier erläutert. Anschließend wird die Konfiguration der graphischen Oberfläche, dem X.Org-Server für diese Konfiguration dargestellt.

2.1 IP- und MAC-Adresse

Um das BeagleBoard-xM im Netzwerk der HAW Hamburg verwenden zu können, wurde von den Administratoren des Netzwerkes eine IP- und eine Macadresse bereitgestellt. Die Netzwerkschnittstelle besaß schon eine andere IP-Adresse zum Zeitpunkt der Übernahme des Boards, daher musste die Netzwerkschnittstelle neu konfiguriert werden.

Das erste BeagleBoard hatte keine Netzwerkschnittstelle. Um dieses mit einem Netzwerk verbinden zu können, war noch ein externer USB-Netzwerkadapter notwendig. Das BeagleBoard-xM hat einen SMSC Lan9512 Chip verbaut. Dies ist ein USB 2.0 Hub und ein Fast Ethernet Controller in einem Gehäuse. Der Fast Ethernet Controller ist per USB mit dem Hub verbunden, sodass er vom Linux-Kernel als USB-Gerät identifiziert wird:

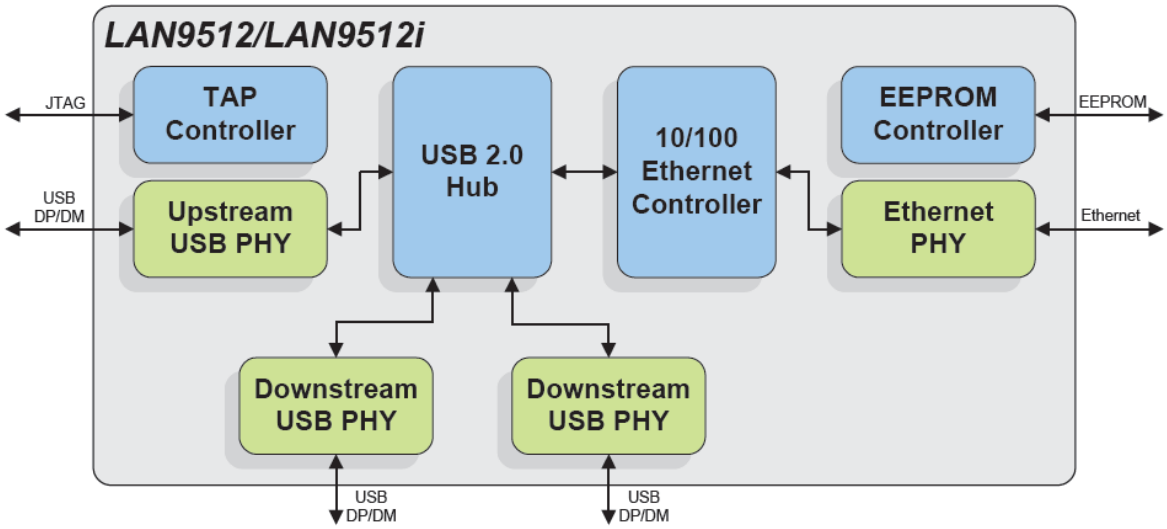

Bild 4: Internes Blockdiagramm des SMSC LAN9512 [4]

```
[   11.988128] smsc95xx v1.0.4
[   12.158050] smsc95xx 1-2.1:1.0: eth0: register 'smsc95xx' at usb-ehci-omap.0-2.1, smsc95xx USB 2.0 Ethernet, 6a:c7:cc:93:6f:0e
```

Listing 2: Auszug aus den Startmeldungen des Linux-Kernels

Der SMSC-Chip kann die Konfiguration, wozu unter anderem auch die MAC-Adresse des Ethernet-Controllers zählt, nach einem Start oder Neustart aus einem angeschlossenen Speicherbaustein lesen. Dieser Baustein ist auf dem BeagleBoard-xM allerdings nicht verbaut. Der Fast Ethernet Controller muss daher durch den Gerätetreiber unter Linux konfiguriert werden. Der Treiber versucht zunächst den nicht angeschlossenen Baustein auszulesen. Da keiner vorhanden ist, wird in Folge dessen eine zufällige Mac-Adresse generiert:

```
604 {
605         /* try reading mac address from EEPROM */
606         if (smsc95xx_read_eeprom(dev, EEPROM_MAC_OFFSET, ETH_ALEN,
607                         dev->net->dev_addr) == 0) {
608                 if (is_valid_ether_addr(dev->net->dev_addr)) {
609                         /* eeprom values are valid so use them */
610                         netif_dbg(dev, ifup, dev->net, "MAC address read from EEPROM\n");
611                         return;
612                 }
613         }
614
615         /* no eeprom, or eeprom values are invalid. generate random MAC */
616         random_ether_addr(dev->net->dev_addr);
617         netif_dbg(dev, ifup, dev->net, "MAC address set to random_ether_addr\n");
618 }
```

Listing 3: Auszug aus Linux/drivers/net/usb/smsc95xx.c

In einem privatem Netzwerk sollte dieses Verhalten nicht stören, im Netzwerk der HAW Hamburg werden Geräte allerdings anhand ihrer IP-Adresse und anhand ihrer MAC-Adresse identifiziert, sodass trotz fehlendem Speicherbaustein für die Konfiguration zwingend eine feste und durch die Administratoren des Netzwerks vorgegebene Mac-Adresse automatisch beim Start des Boards eingestellt werden muss.

Debian Linux verwendet für die Hinterlegung der Konfiguration der Netzwerkschnittstelle die Textdatei /etc/network/interfaces. Hier lassen sich Parameter wie IP-Adresse, Standard-Router und DNS-Server eintragen. Zusätzlich zu diesen Informationen kann hier auch die MAC-Adresse der Schnittstelle festgelegt werden. Dies geschieht über die Zeile „hwaddress ether <MAC Adresse>“. Ein gültiger Eintrag lautet also „`hwaddress ether 02:80:0F:11:72:04`“. Dies wird vom Gerätetreiber allerdings nicht wahrgenommen. Stattdessen muss vor der Aktivierung der Schnittstelle mit dem „ifconfig“ Kommando die MAC-Adresse gesetzt werden. Die Konfigurationsdatei bietet dafür den Eintrag „pre-up“. Das nach „pre-up“ folgende Kommando wird nach der Konfiguration und vor der Aktivierung der

Schnittstelle ausgeführt. Für die für diese Arbeit vorgegeben Parameter lautet die Konfiguration demnach:

```
# The primary network interface
allow-hotplug eth0
iface eth0 inet static
       address 141.22.14.76
       netmask 255.255.252.0
       network 141.22.12.0
       broadcast 141.22.15.255
       gateway 141.22.12.1
       dns-nameservers 141.22.192.100
       dns-search etech.haw-hamburg.de
       pre-up /sbin/ifconfig eth0 hw ether 02:80:0F:11:72:04
```

Listing 4: Auszug aus der Datei /etc/network/interfaces

Durch diese Konfiguration ist sichergestellt, dass die Netzwerkschnittstelle beim Start des Boards mit der richtigen MAC-Adresse konfiguriert wird.

2.2 Display

Am Board soll ein acht Zoll großes Touchdisplay der Firma Lilliput Electronics Co., Ltd. aus China [11] als Monitor angeschlossen werden. Dieses Display verfügt über einen Eingang für ein herkömmliches Video Graphics Array (VGA) Signal und einen Eingang für ein Composite Video Signal.

2.2.1 S-Video

Da das Board neben einem digitalem High Definition Multimedia Interface (HDMI) auch einen analogen S-Video Ausgang mit einer Mini-DIN Buchse besitzt, soll das Display an letztere angeschlossen werden. Um das Board mit dem Display verbinden zu können, müssen das Luminanz- und das Chrominanzsignal des S-Video Ausgangs zu einem Signal zusammengemischt werden. Das Luminazsignal („Helligkeitsinformation“, Y in Bild 5) würde für sich alleine schon genügen, man würde allerdings nur ein schwarz-weiß Bild auf dem Display erhalten. Daher muss das Luminanzsignal („Farbinformation“, C in Bild 5), wie in Bild 5 dargestellt, beigemischt werden.

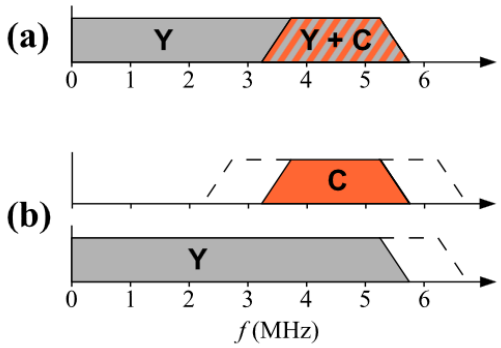

Bild 5: Spektren von S-Video (a) und Composite-Video (b) [5]

Um die beiden Signale zusammenzumischen, gibt es bereits fertige S-Video auf Composite Video Adapter preiswert im Fachhandel. Daher wird auf einen Eigenbau verzichtet.

Um nun den S-Video Ausgang einzuschalten, muss die serielle Konsole deaktiviert werden und es muss dem Kernel über einen Bootparameter angegeben werden, dass das Display am S-Video Ausgang als primäres Display verwendet werden soll. Daher muss in den Bootprozess eingegriffen werden. Findet dies nicht statt, wird der HDMI-Ausgang angesteuert. Wird der Linux-Kernel mit einem speziellen Parameter gestartet, werden alle Textmeldungen, die normalerweise auf dem Bildschirm erscheinen, über eine RS232-Schnittstelle gesandt. Dies ist nützlich, wenn der mit Linux betriebene Rechner, so wie in diesem Fall, über keinen Bildschirm verfügt.

2.2.1.1 Bootprozess

Der verwendete Prozessor verfügt über 64kB internen SRAM (Static Random Access Memory) und 32 Kilobyte ROM-Speicher (Read Only Memory). In diesen ist ab Werk ein Bootcode, der sog. ROM-Code, installiert. Dieser initialisiert den Hauptspeicher des Boards und die Schnittstelle zur SD-Card. Danach lädt er einen größeren Bootloader, den MLO, eine Kombination aus dem Texas Instruments X-Bootloader und einer Konfigurationsdatei, die angibt, an welche Speicheradresse dieser geladen werden soll, von der Speicherkarte in den Hauptspeicher und startet ihn. Der U-Boot-Loader befindet sich mit dem MLO in einer primären Partition mit einem FAT-16 Dateisystem und MBR den ersten Sektoren der SD-Card. Für mehr Funktionalität reicht der vorhandene Programmspeicher nicht aus. Nachdem U-Boot durch den X-Bootloader gestartet wurde, stehen diesem neben den 64kB on-chip SRAM nun auch die 512MB Hauptspeicher zur Verfügung, daher ist er umfangreicher als der TI X-Bootloader ([16], Kapitel 26.4.7.6).

Der U-Boot-Loader ist in der Lage, komplexere Dateisysteme als das FAT-16 auf der SD-Karte zu lesen. Er lädt eine Konfigurationsdatei, in der die Startparameter des Linux-Kernels stehen, nebst dem Kernelimage aus dem EXT3-Dateisystem aus einer logischen Partition auf

der Speicherkarte in den Hauptspeicher. Anschließend wird der Kernel mit den angegebenen Parametern gestartet. Der Startprozess des Boards ist also vierphasig.

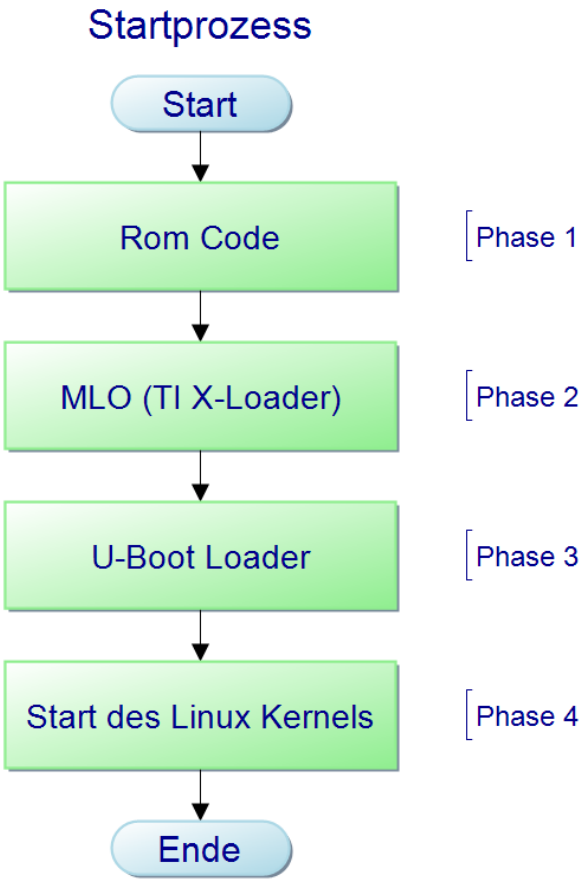

Bild 6: Startprozess des Boards

Dieser Kette aus Bootloadern kommt letzten Endes die gleiche Funktionalität zu wie dem Basic Input and Output System (BIOS) auf einem PC: Initialisierung der Hardware und Start des Betriebssystems.

Um nun die Serielle Konsole des Linux Kernels deaktivieren zu können, muss also die Konfigurationsdatei des U-Boot Loaders geändert werden. Diese befindet sich unter /boot/uboot/ und heisst „boot.scr“ und wird mit einem Programm aus der datei „boot.cmd“ im gleichen Verzeichnis erzeugt:

```
mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "Debian on BeagleBoard-xM" -
d boot.cmd boot.scr
```

Listing 5: Aufruf des mkimage-Kommandos

Es darf nur die boot.cmd geändert werden. Die boot.scr enthält zusätzlich zu den Einstellungen auch noch eine Prüfsumme, die durch den U-Boot Loader beim Starten geprüft

wird. Wenn die Prüfsumme nicht stimmt, wird der Startvorgang abgebrochen und eine Fehlermeldung auf die Serielle Schnittstelle ausgegeben.

Um die Serielle Konsole zu deaktivieren, genügt es, die Zeichenkette „`console=tty0 console=ttyS0,115200n8`“ mit einem Texteditor aus den Bootparametern zu entfernen, die Datei „boot.scr“ mit dem o.g. Kommando neu zu erzeugen und das Board neu zu booten. Um nun zusätzlich den S-Video Ausgang in Betrieb zu nehmen, muss den Bootparametern die Zeichenkette „`omapfb.mode=tv:pal omapdss.def_disp=tv omapfb.video_mode=720x576MR-24@25`“ hinzugefügt werden. Dies bewirkt, dass als Standard der TV-Ausgang (S-Video) im in Europa verbreiteten PAL-Modus mit einer Auflösung von 720x576 Bildpunkten mit 24 Bit Farbtiefe bei 25 Bildern pro Sekunde verwendet wird. Die gewählte Auflösung entspricht der in der Pal-Norm für das terrestrische Fernsehen in Europa verwendeten Auflösung.

Die Startparameter in der boot.cmd Datei lauten daher:

```
setenv bootargs root=/dev/mmcblk0p5 rootwait ro fixrtc buddy=${buddy}
mpurate=1000 omapfb.mode=tv:pal omapdss.def_disp=tv
omapfb.video_mode=720x576MR-24@25
```

Listing 6: Auszug aus der Datei /boot/uboot/boot.cmd

Wenn man den Linux Kernel mit den o.g. Parametern startet, erscheinen dessen Bootmeldungen auf dem Display. Man sieht allerdings die Meldungen der vorherigen Bootloader (X- und U-Boot) nicht. Wenn das System ohne den X.Org-Server gestartet wird, ist das Display ein Ersatz für einen Monitor.

Die Änderung des Parameters mpurate von „${mpurate}“ auf „1000“ bewirkt, dass die CPU nicht mit 600MHz, sondern mit 1000MHz, also 1,0GHz, getaktet wird. Dies Funktioniert allerdings erst ab Linux 3.1 stabil. Bevor dieser Parameter verwendet werden kann, ist also ggf. das Einspielen eines neuen Kernels notwendig. Wenn der Aufruf erfolgreich ist, liefert der Kernel beim Start eine entsprechende Meldung:

```
[    0.103088] Switched to new clocking rate (Crystal/Core/MPU): 26.0/332/1000 MHz
```

Listing 7: Startmeldung des Kernels über das Umschalten des Systemtaktes

2.2.1.2 X.Org Server

„Der X.Org-Server ist der Hauptteil der offiziellen Referenzimplementierung X.Org des X-Window-Systems.“ [7]

„Das X Window System (auch: X Version 11, X11, X) ist ein Netzwerkprotokoll und eine Software, die Fenster auf Bitmap-Displays auf den meisten unixoiden Betriebssystemen und OpenVMS ermöglicht.“ [6]

Der X.Org Server ist nicht nur unter Linux dafür zuständig, dem Benutzer eine graphische Benutzeroberfläche zur Verfügung zu stellen. Er kann, mit entsprechenden Treibermodulen ausgestattet, direkt mit der Hardware kommunizieren und so auch komplexere Berechnungen der Oberflächenelemente auf dedizierte Recheneinheiten (z.B.: Graphic Processing Units, GPU’s, oder auch DSP‘s) auslagern, was zu einer Entlastung des Hauptprozessors und außerdem zu einem signifikanten Geschwindigkeitsgewinn der graphischen Anwendungen führt.

Um den X.Org Server auf dem BeagleBoard-xM verwenden zu können, muss dieser installiert sein und es muss eine Konfigurationsdatei erstellt werden. Unter Debian Linux heißt diese „xorg.conf“ und liegt unter „/etc/X11/“. Diese Datei beinhaltet, welche Treibermodule der Server beim Start laden soll und wie die Angeschlossenen Monitore und Eingabegeräte konfiguriert und verwendet werden sollen.

```
Section "Monitor"
        Identifier "Configured Monitor"
EndSection

Section "Screen"
        Identifier "Default Screen"
        Device "Configured Video Device"
        #Limited by SGX?
        DefaultDepth 16
        SubSection "Display"
        Depth  16
        Modes  "320x200"
        EndSubSection
EndSection

Section "Device"
         Identifier "Configured Video Device"
#        Driver "omapfb"
#        Option "fb" "/dev/fb0"
EndSection
```

Listing 8: Inhalt der Datei /etc/X11/xorg.conf [21]

Mit dieser Konfiguration ist der X.Org Server in der Lage, zu starten. Das Treibermodul, welches der Server lädt, das sogenannte OmapDSS, verwendet auch den S-Video Ausgang des Boards. Dieser muss allerdings noch einmal im Treiber selbst eingeschaltet werden, bevor ein Bild auf dem Display erscheint. Dieses Einschalten geschieht durch den Befehl

„`echo 1 > /sys/devices/omapdss/display1/enabled`“.

Dieser Befehl muss zwingend vor jedem Start des Servers ausgeführt werden. Um diesen Befehl automatisch beim Start des Boards und vor dem automatischen Start des X.Org Servers auszuführen, wird ein Init-Skript benötigt. Dieses Skript ist ein Shellscript, welches beim Booten des Boards mit dem Parameter „enable“ vom init-Prozess aufgerufen wird.

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          tv-out
# Required-Start:    $local_fs $syslog kdm
# Required-Stop:     $local_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# X-Interactive:     true
# Short-Description: enable/disbale s-video output
### END INIT INFO

set -e

enable() {
    sudo sh -c "echo 1 > /sys/devices/omapdss/display1/enabled"
}

disable() {
    sudo sh -c "echo 0 > /sys/devices/omapdss/display1/enabled"
}

case $1 in
       start)
              echo "Enabling Tv-Output"
              enable
       ;;
       stop)
              echo "Disablig Tv-Output"
              disable
       ;;
       restart)
              echo "Restarting Tv-Output"
              disable
              enable
       ;;
       status)
           cat /sys/devices/omapdss/display1/enabled
       ;;
       *)
              exit 1
       ;;
esac
```

Listing 9: Inhalt der Datei /etc/init.d/enable-tv

Durch dieses Skript wird sichergestellt, dass der S-Video Ausgang beim Systemstart eingeschaltet wird und dass damit die graphische Anmeldemaske des Sitzungsmanagers angezeigt werden kann.

Es zeigten sich allerdings drei Probleme beim Betrieb des Displays mit einer Signalspeisung per S-Video. Das erste Problem war, dass das Display nicht das komplette Bild darstellt und keine Einstellmöglichkeiten für eine Korrektur bietet. Es fehlten an den an den seitlichen Rändern sowie oben und unten einige Bildpunkte: d.h. es wurde lediglich ein Bereich um den

Mittelpunkt des an das Display gesendeten Bildes dargestellt. Dies führt dazu, dass auf der Linux Konsole einige Zeilen nicht dargestellt werden können.

Das zweite Problem lag in der Unschärfe des dargestellten Bildes. Analoge Signale sind sehr empfindlich gegen Rauschen, und die vorgestellte Lösung mit dem gekauften S-Video Adapter verstärkt das Problem allerdings zusätzlich.

Bild 7: Unscharfe Darstellung der Linux Konsole

Das dritte Problem ist die Touch-Funktionalität: Der Controller, der im Display prüfen soll, ob das Display berührt wurde, wird per USB mit dem Board verbunden. Er emuliert dann eine Maus mit zwei Tasten, sodass zunächst kein zusätzlicher Treiber nötig ist. Der Controller wird aber nur aktiviert, wenn das Display über den VGA-Eingang gespeist wird. Führt man das Signal per S-Video zu, bleibt der Controller deaktiviert. Die Touch-Funktionalität ist nicht vorhanden.

Aus diesen drei Problemen folgt, dass eine Speisung des Displays per S-Video keine geeignete Lösung für den dauerhaften Betrieb im Labor der HAW Hamburg darstellt. Es muss nach einer Lösung gesucht werden, die das Display mit einem VGA-Signal speisen kann.

2.2.2 HDMI

Um ein besseres Ergebnis zu erzielen, wurde ein im Vergleich zum S-Video-Adapter sehr viel aufwendigerer und auch teurerer HDMI zu VGA Umsetzer angeschafft. Der Umsetzer bietet keinerlei Einstellmöglichkeiten, sodass daher keine Konfiguration notwendig ist. Der HDMI-Ausgang des Boards wird mit dem HDMI-Eingang des Umsetzers verbunden, der VGA-Ausgang des Umsetzers mit dem VGA-Eingang des Displays. Der USB-Steckverbinder des Displays wird direkt mit dem Board verbunden. Die in Kapitel 2.2.1.1 hinzugefügten Parameter für den Start des Linux-Kernels müssen wieder entfernt werden. Nach einem Neustart des Boards muss das Display ggf. noch auf „PC“ als Signalquelle umgestellt werden. Danach zeigt das Display ein scharfes Bild an und aktiviert den Touch-Controller, was der Kernel mit einer Meldung bestätigt.

```
[    3.612823] usb 1-2.2: new full speed USB device number 4 using ehci-omap
[    3.731567] usb 1-2.2: New USB device found, idVendor=0eef, idProduct=0001
[    3.734649] usb 1-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[    3.740600] usb 1-2.2: Product: USB TouchController
[    3.743499] usb 1-2.2: Manufacturer: eGalax Inc.
[    4.234741] input: eGalax Inc. USB TouchController as /devices/platform/usbhs-
omap.0/ehci-omap.0/usb1/1-2/1-2.2/1-2.2:1.0/input/input1
[    4.242248] input: eGalax Inc. USB TouchController as /devices/platform/usbhs-
omap.0/ehci-omap.0/usb1/1-2/1-2.2/1-2.2:1.0/input/input2
[    4.250000] generic-usb 0003:0EEF:0001.0001: input,hidraw0: USB HID v2.10
Pointer [eGalax Inc. USB TouchController] on usb-ehci-omap.0-2.2/input0
[    4.256439] usbcore: registered new interface driver usbhid
[    4.259613] usbhid: USB HID core driver
```

Listing 10: Startmeldungen des Kernels über eGalax Touch Controller

Nun kann das mit dem Display mitgelieferte Installationsskript als Benutzer „root“ gestartet werden. Dieses prüft zunächst, ob die Berechtigungen des aufrufenden Benutzers ausreichen. Sind sie ausreichend, wird das mitgelieferte Modul für den X.Org Server installiert und automatisch in die Konfigurationsdatei des X.Org Servers eingebunden. Da der Linux-Kernel beim Start automatisch ein Standardmodul für den Touch-Controller lädt (USB-HID, siehe Listing 10), wird dieses nach Nachfrage für diesen Controller auf die schwarze Liste gesetzt, sodass es nicht mehr geladen wird, der Controller durch das Modul für den X.Org-Server verwaltet werden kann. Somit zeigt das Display nun ein scharfes Bild an und die Touch-Funktionalität ist auch gegeben.

Es bleibt das Problem offen, dass die die Touch-Funktionalität nicht präzise genug funktioniert. Dieses Problem lässt sich durch eine Kalibration des Displays lösen. Für die Kalibration wird vom Hersteller des Displays ([11]) eine Software mitgeliefert. Mit dieser lässt sich das Display unter Windows und Linux kalibrieren - sie ist allerdings nicht auf ARM-Prozessoren ausführbar (siehe Listing 11).

```
e6bs@beagle3:~$ eGalaxTouch
-bash: /usr/bin/eGalaxTouch: cannot execute binary file
e6bs@beagle3:~$ file /usr/local/eGalaxTouch32/eGalaxTouch
/usr/local/eGalaxTouch32/eGalaxTouch: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.2.5, not stripped
```

Listing 11: Eigenschaften der Datei eGalaxTouch

Das Display wurde an einem Labor-PC unter Debian Linux kalibriert. Die Datei mit den erzeugten Kalibrationsdaten (/var/lib/eeti.param) wurde im Anschluss auf das Board übertragen und der X-Server neu gestartet. Damit funktionierte die Touch-Funktionalität gut, allerdings bewegte sich der Cursor auf der Abszissenachse genau entgegengesetzt zur Bewegungsrichtung des Fingers auf dem Display. Die Eingabe muss auf dieser Achse also um 180° gedreht werden. Dies geschieht nach der Anmeldung an der graphischen Oberfläche mit dem Befehl „xinput set-prop 8 „Evdev Axis Inversion“ 1, 0“. Die „8“ ist die ID des TouchControllers unter X. Da es passieren kann, dass sie sich bei einem Neustart des Boards ändert, sollte sie zuvor mit dem Befehl „xinput list“ nachgeschlagen werden.

Bild 8: per VGA gespeistes Display

Nach dem Kalibrieren des Displays auf einem Linux-PC und dem Spiegeln der Abszissenachse für die Eingabe funktionierten das Display und die Touch-Funktionalität hinreichend präzise. Damit steht ein berührungssensitiver Bildschirm für das BeagleBoard-xM zur Verfügung.

3. Entwicklungsumgebung

In diesem Kapitel wird die Fragestellung untersucht, auf welche Weise am effektivsten Software für das BeagleBoard-xM entwickelt werden kann. Da die Ressourcen des Boards knapp bemessen sind, ist es sehr unergonomisch, die Software direkt auf dem Board zu entwickeln. Daher wird im ersten Schritt untersucht, auf welche Art ein Datentransfer auf das Board stattfinden kann. Im zweiten Schritt werden dann die Entwicklungsumgebungen Netbeans und Eclipse auf ihre Verwendbarkeit für das sogenannte Remote Development hin untersucht.

3.1 Datenaustausch

Nachdem das Board in Betrieb genommen wurde, stellt sich nun die Frage, mit welcher Umgebung man am effektivsten Software für dieses entwickelt. Auf dem Board ist die GNU Compiler Collection (GCC) installiert. Man könnte also einen Texteditor auf dem Board starten, den Quelltext schreiben, die Textdatei abspeichern und kompilieren. Dieses Vorgehen ist allerdings sehr unergonomisch. Ergonomischer ist es, eine komplette Entwicklungsumgebung, wie beispielsweise Eclipse, zu verwenden. Aufgrund der Ressourcenknappheit des Boards und der Größe des verwendeten Displays ist es allerdings trotzdem unergonomisch und von langen Warte- und Ladezeiten geprägt, die Software mit einer IDE direkt auf dem Board zu entwickeln. Die Idee ist daher, die IDE auf einem Entwicklungs-PC laufen zu lassen und lediglich den Prozess des Kompilierens, des Linkens und des Ausführens auf dem Board ausführen zu lassen.

Um den Quelltext auf einem Windows-PC schreiben und auf dem Board kompilieren, linken und ausführen zu können, muss der Quelltext beiden Maschinen zur Verfügung stehen. Man benötigt also eine gemeinsame Ressource, auf der der Quelltext liegt oder eine Möglichkeit, den Quelltext auf das Board zu übertragen. Idealerweise erstellt man auf dem Board also eine Freigabe, weist dieser auf dem Windows-PC einen Laufwerksbuchstaben zu und legt das Projektverzeichniss der IDE auf dieses Netzlaufwerk.

Da sowohl das BeagleBoard-xM als auch der PC über eine konfigurierte und betriebsbereite Netzwerkschnittstelle verfügen, bietet sich die Verwendung dieser an. Es existieren diverse Möglichkeiten, Speicherplatz für die Verwendung in einem Netzwerk freizugeben. Dazu zählen Web-Based Distributed Authoring and Versioning (WebDAV), das Network File System (NFS), Secure Copy (SCP), sowie das Server Block Message Protocol (SMB) und das Secure Shell File System.

3.1.1 Network File System

Microsoft stellt seit Windows 2000 Server die Microsoft Services for UNIX, seit Windows Server 2003R2 das Subsystem for UNIX-based Applications als Erweiterungspaket für das Betriebssystem Windows Server zur Verfügung. Dieses Paket enthält unter anderem einen NFS-Client und einen NFS-Server. In den Desktop Versionen von Windows ist der NFS-Server nicht enthalten, es ist lediglich der NFS Client enthalten.

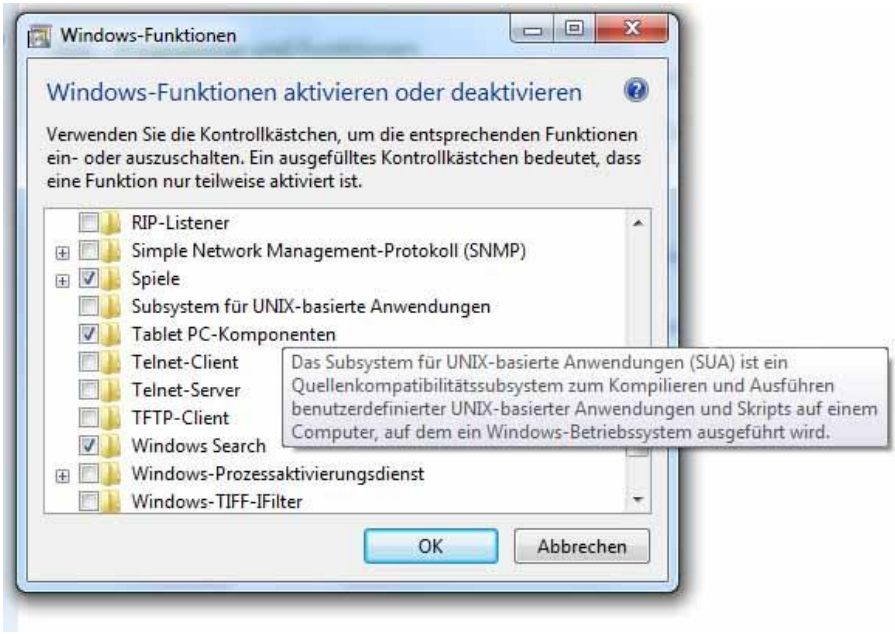

Bild 9: Subsystem Für Unix in der Windows-Systemsteuerung unter Windows 7

Es gibt also die Möglichkeit, auf dem BeagleBoard-xM ein Verzeichnis zu erstellen, dieses per NFS im Netzwerk freizugeben und dieser Freigabe auf dem Windows-PC einen Laufwerksbuchstaben zuzuweisen. Um allerdings auf diese Freigabe zugreifen zu können, muss eine Abbildung zwischen den Windows Benutzern und den UNIX Benutzern geschaffen werden. Es reicht also nicht einfach, einen Benutzernamen und ein Passwort in einer Anmeldemaske einzugeben. Dazu muss eine Kopie der /etc/passwd und der /etc/shadow auf den Windows-Pc überspielt werden. Dabei besteht das Problem, dass Windows nicht die korrekte User- und Gruppen-id zum Anmelden am NFS-Server verwendet, wie sie in der /etc/passwd definiert ist, sondern stattdessen einen Default Wert, der in der Registry steht und für alle Windowsbenutzer gleich ist. D.h. Windows verwendet für jeden Windows-Benutzer die gleiche UID/GID-Kombination für das Login am NFS-Server [2]. Daher ist das NFS an von mehreren Personen genutzten Windows-PCs keine geeignete Möglichkeit für eine sichere Datenübertragung in einem Netzwerk. Würde die Entwicklung hingegen auf Linux PCs stattfinden, so wäre das NFS ein geeignetes Mittel, um Dateien zwischen dem Entwicklungs-PC und dem Board austauschen zu können.

Ein weiterer gravierender Nachteil des NFS ist, das normalerweise das UDP Protokoll, welches keine Transportsicherung kennt, verwendet wird. Außerdem sind die Namen der Freigaben ohne Anmeldung am Server sichtbar, was Angreifern zusätzliche Angriffsfläche verschafft, weswegen das NFS oftmals in Firewalls gesperrt ist.

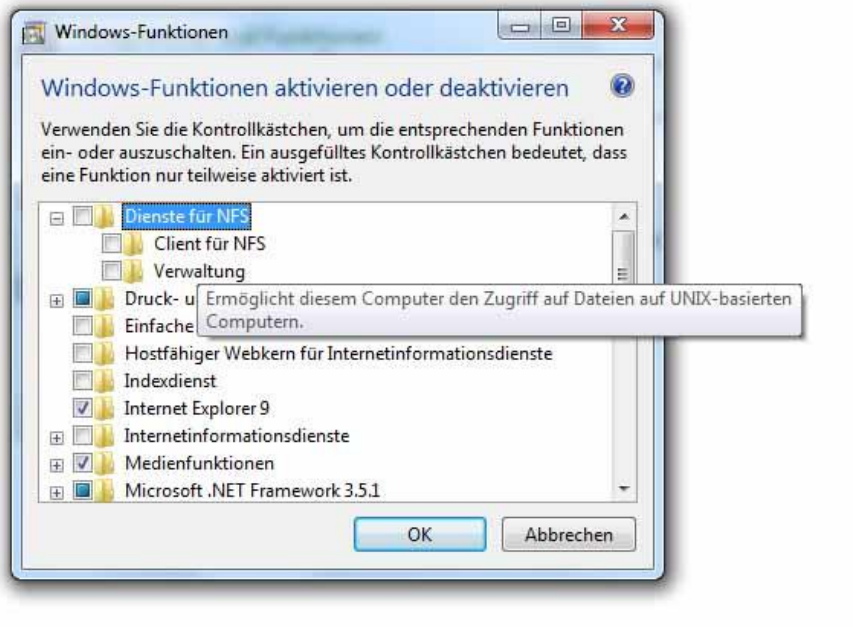

Bild 10: Dienste für NFS in der Systemsteuerung von Windows 7

3.1.2 Web-Based Distributed Authoring and Versioning (WebDAV)

Das Web-Based Distributed Authoring and Versioning ist kein eigenes Protokoll, sondern eine Erweiterung des Hypertext Transmission Protocols (http) und erbt damit alle Vor- und auch Nachteile. Der Größte Vorteil des Protokolls ist die Einfachheit und die daraus resultierende Robustheit. Diese Einfachheit ist allerdings auch der größte Nachteil des Protokolls: es ist zustandslos und kennt keinerlei Verschlüsselung der übertragenen Daten, es sendet also einen Benutzernamen und das dazugehörige Passwort im Klartext durch das Netzwerk. Für dieses Problem existieren allerdings Lösungen, die die Verschlüsselung nicht auf Anwendungs- sondern auf Präsentationsebene ansetzten (Bsp.: Secure Socket Layer, SSL). Durch Erzeugung oder durch den Kauf eines Zertifikats von einer Zertifizierungsstelle lässt sich SSL im Apache Webserver verwenden.

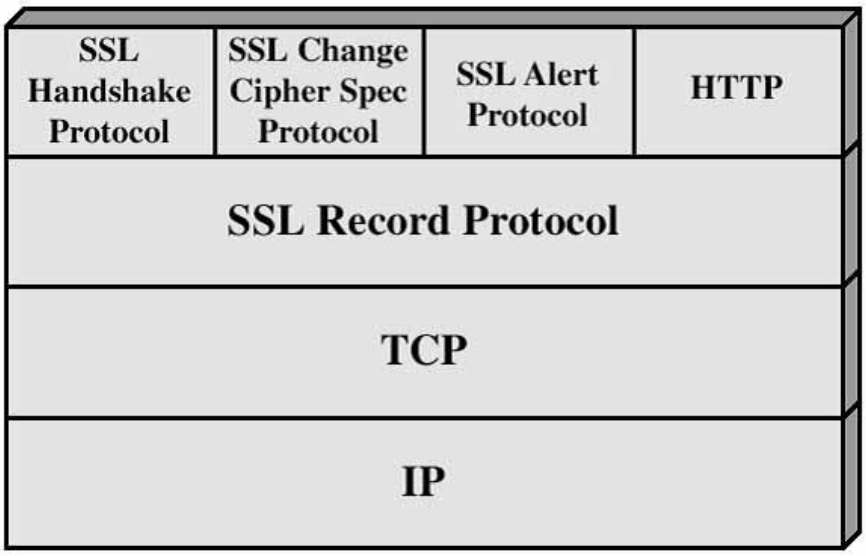

Bild 11: SSL im Osi-Schichtenmodell [3]

Aufgrund der hohen Verbreitung des http- und des https-Protokolls sind sie in den seltensten Fällen in einer Firewall gesperrt. Daher ist es auch für den Einsatz im Internet geeignet. Falls also das BeagleBoard-xM und der PC, auf dem die Software entwickelt wird, sich nicht im gleichen Netzwerk befinden, ist das https-Protokoll eine Möglichkeit für den Datenaustausch. Für Debian Linux ist eine Vielzahl von Webservern verfügbar. Am gebräuchlichsten ist jedoch der Apache Webserver.

Ein weiterer Nachteil ist, dass das http-Protokoll zwar eine grundlegende Authentifikation kennt, die dazugehörige Datenbank mit den Benutzernamen und Passwörtern wird allerdings nur Webserverintern verwendet und ist vollkommen unabhängig von den Benutzernamen und den Passwörtern des zugrundeliegenden Betriebssystems. Dies lässt sich zwar umkonfigurieren, sodass Webserver und Betriebssystem auf die gleiche Datenbank zugreifen (z.B.: Lightweight Directory Access Protocol [LDAP], Microsoft Active Directory [MSAD]). Dies stellt für den hier verfolgten Zweck, den einfachen und automatischen Dateiaustausch zwischen einem Entwicklungs-PC und dem BeagleBoard-xM allerdings einen nicht zu unterschätzenden Mehraufwand da.

Der Webserver läuft aus Gründen der Systemsicherheit unter einem eigenen Systembenutzer, der keine erweiterten Benutzerrechte im System hat. Unter Debian Linux heißt dieser „www-data“. Wenn man also eine Datei auf einer WebDAV Freigabe speichert, übergibt man diese Datei dem Webserver und dieser schreibt sie dann in das lokale Dateisystem. Die Datei wird also mit dem Systembenutzer des Webservers und nicht mit dem am Webserver angemeldeten Benutzer geschrieben. Dies führt in dem Augenblick zu Problemen mit den Berechtigungen, in dem man eine zweite Verbindung unter einem eigenen Systembenutzer zu dem Board aufbaut, um den Compiler zu starten. Bevor nun der Compiler gestartet werden kann, müssen diese Probleme gelöst werden. Wenn nun der Quelltext kompiliert wurde, so hat das erzeugte

Kompilat der Berechtigungen des Systembenutzers. Bevor nun über eine WebDAV-Freigabe darauf zugegriffen werden kann, müssen diese Berechtigungen ebenfalls korrigiert werden.

Bild 12: WebDAV-Freigabe als Laufwerk unter Windows 7

Der Vorteil dieses Verfahrens ist allerdings, dass die erzeugte WebDAV-Freigabe unter M.S. Windows 7 als normales Laufwerk in den Explorer integriert und genutzt werden kann. Daher bietet sich die WebDAV-Erweiterungen in Kombination mit SSL für eine Remote-Tätigkeit, beispielsweise von zu Hause aus an.

3.1.3 Server Block Message Protocol

Das Server Block Message Protocol (SMB-Protokoll) wird für den Datenaustausch zwischen Windows- Servern und Clients verwendet. Es wurde von der Firma Microsoft entwickelt und es sind in den Desktopversionen von Microsoft Windows sowohl ein Server (in eingeschränkter Form) und ein Client enthalten. Es ist allerdings in der Firewall der HAW Hamburg gesperrt. Daher ist ein Datenaustausch über das Internet nicht möglich.

Als Server steht für das Linux Betriebssystem der Samba Server zur Verfügung. Dieser ist ein vollwertiger und kostenloser Server unter einer Open Source Lizenz. Den Freigaben, die dieser Server bereitstellt, kann unter Windows ein Laufwerksbuchstabe zugewiesen werden. Der Vorteil ist, dass dieser Server auf die Benutzerdatenbank des zugrundeliegenden Betriebssystems zurückgreift. Daher gibt es die o.g. Probleme mit den Berechtigungen hier nicht. Auf diese Weise ist ein automatischer Datenaustausch mit dem Board möglich. Daher soll eine SMB-Freigabe auf dem Board für diese Arbeit genutzt werden.

Die Konfigurationsdatei des Servers heißt smb.conf und liegt unter /etc/samba/. Sie enthält neben der globalen Konfiguration (Freigabe von Druckern aktivieren, Server ist Domaincontroller oder Masterbrowser in der Arbeitsgruppe etc.) auch die Definitionen der einzelnen Freigaben. Bei der unter Debian Linux dem Installationspaket beiliegenden Konfigurationsdatei werden die Benutzerverzeichnisse unter /home/ standardmäßig für den Lesezugriff freigegeben. Um auch den Schreibzugriff zu ermöglichen, muss der Parameter „read only = yes“ in „read only = no“ im Abschnitt [homes] geändert und anschließend der Server neu gestartet werden.

Nun kann man sich mit seinem Systembenutzernamen und seinem Systempasswort am Samba Server anmelden und danach auf sein Benutzerverzeichnis zugreifen. Dieser Freigabe weist man auf dem Windows PC für die Entwicklung einen freien Laufwerksbuchstaben zu. Auf diesem neu geschaffenem Laufwerk legt man nun in der Entwicklungsumgebung das Projekt an. Man arbeitet also direkt auf dem Dateisystem und damit auf der Speicherkarte des BeagleBoard-xM. Aufgrund dessen kann es bei sehr großen Projekten zu Performanceengpässen kommen.

3.1.4 Secure Shell File System

Findet die Entwicklung unter Linux statt, existiert die Möglichkeit, auf ein entferntes Dateisystem mit dem sog. SSHFS per SSH zuzugreifen. Man kann sich damit ein entferntes Verzeichnis in den lokalen Verzeichnisbaum einhängen, die Benutzerauthentifizierung sowie die Verschlüsselung der Übertragung werden dabei durch SSH gewährleistet. Da SSH auf die Datenbank mit den Systembenutzern zugreift, gibt es auch keine Probleme mit den Berechtigungen neu angelegter Dateien. Unter Debian Linux muss dazu das Paket „sshfs“ mit dem Kommando „`apt-get install sshfs`“ installiert werden. Auf dem Board muss dazu lediglich der Openssh Server installiert sein. Weitere Komponenten werden serverseitig nicht benötigt. Der zum Einhängen nötige Aufruf lautet „`sshfs user@host:/path/to/directory /local/directory`“. Es wird in SSH-typischer Weise nach einem Passwort für den Benutzer am entfernten System gefragt, alternativ können auch SSH-Keys für die Authentifikation verwendet werden.

```
server1:~# ls -l /opt/e6bs/
insgesamt 0
server1:~# sshfs e6bs@beagle3.etech.haw-hamburg.de:/home/e6bs /opt/e6bs/
e6bs@beagle3.etech.haw-hamburg.de's password:
server1:~# ls -l /opt/e6bs/
insgesamt 32844
drwxr-xr-x 1 root       root           4096  8. Feb 19:38 BeagleBoard-xM GPIO
drwxr-xr-x 1 root       root           4096 10. Feb 14:43 BeagleBoard-xM Sonar
.
.
server1:~# mount
.
.
fusectl on /sys/fs/fuse/connections type fusectl (rw)
e6bs@beagle3.etech.haw-hamburg.de:/home/e6bs on /opt/e6bs type fuse.sshfs
(rw,nosuid,nodev,max_read=65536)
server1:~#
```

Listing 12: Einhängen eines Verzeichnisses per SSHFS

Da die Übertragung verschlüsselt stattfindet und da SSH in den seltensten Fällen in Firewalls gesperrt wird, eignet sich dieses Vorgehen für eine Tätigkeit von entfernten Orten, also auch von zuhause aus.

Aufgrund dessen, dass der Samba-Server die Systembenutzerdatenbank verwendet, es nicht möglich ist, diesen über das Internet anzugreifen, und es keine Probleme mit den Berechtigungen im Dateisystem gibt, soll dieser für die Entwicklung verwendet werden.

3.2 Netbeans sowie die Erstellung eines Remote Build Project

Netbeans ist eine freie Entwicklungsumgebung (Integrated Development Environment, IDE) für die Sprachen C/C++, Java, PHP und andere. Die IDE wurde ursprünglich von einer von der Firma Sun Microsystems Inc. aufgekauften Firma entwickelt. Sun Microsystems Inc. gab diese Software kurz nach der Übernahme als Open Source frei, so dass sich eine größere Gemeinschaft um das Projekt bilden konnte, die es seitdem kontinuierlich weiterentwickelt. Zum Zeitpunkt der Entstehung dieser Arbeit ist die Version 7.0.1 aktuell, die unter http://www.netbeans.org kostenlos heruntergeladen werden kann. Um Netbeans verwenden zu können, muss eine Java Virtual Maschine auf dem PC installiert sein, da die IDE in Java geschrieben ist. Möchte man die IDE für C oder C++ Entwicklung nutzen, sollte man das C/C++ Paket von der Internetpräsenz des Projektes herunterladen.

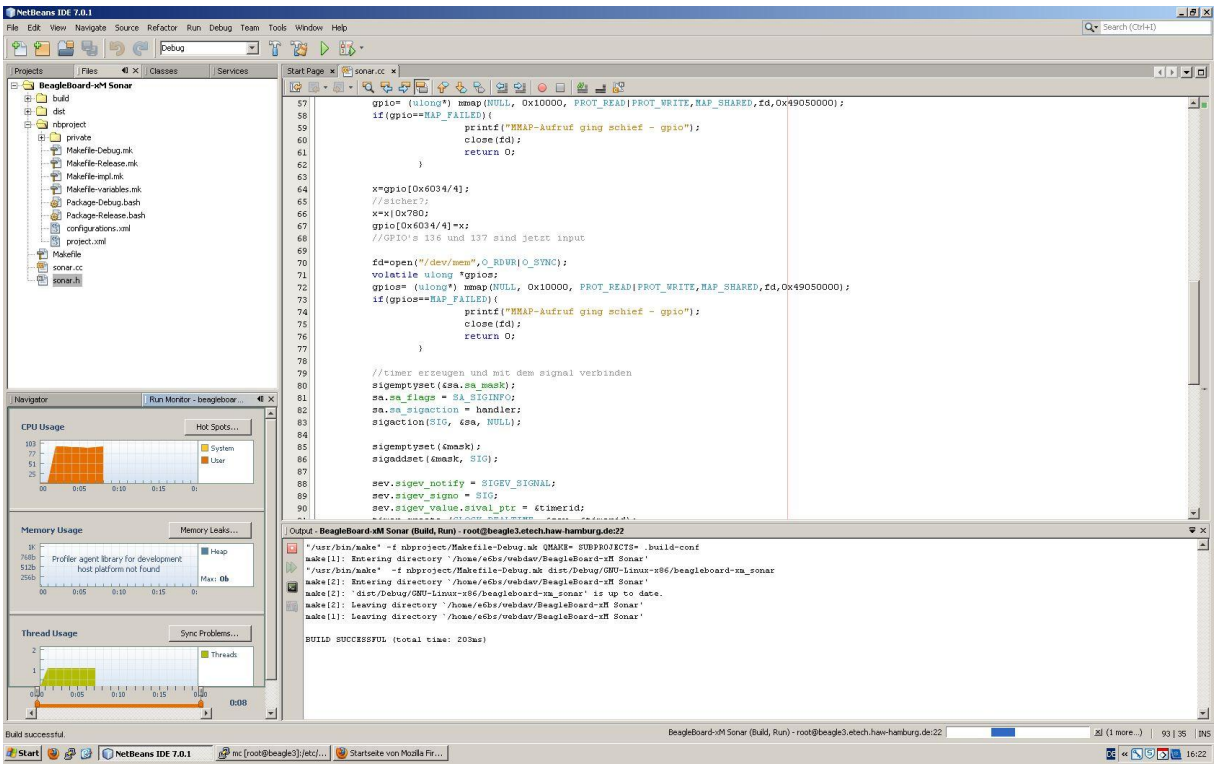

Bild 13: Netbeans 7.0.1 auf einem Windows XP PC für die C/C++ Entwicklung

Da Netbeans unter einer Open-Source Lizenz erhältlich ist, ist es stark mit den gängigen Open Source Entwicklungswerkzeugen wie der Gnu Compiler Collection (gcc) oder dem Gnu Debugger (gdb) verzahnt.

Die IDE bietet von sich aus das Feature des „Remote Developement“. Man trennt also das Schreiben des Quelltextes, und die Erzeugung, das Ausführen und das Ausführen im Einzelschrittverfahren von Maschinencode. Dazu kann man sog. „Build Hosts“ definieren. Auf diesem Rechner im Netzwerk wird dann der geschriebene Quelltext kompiliert, gelinkt und ausgeführt. Ggf. findet dort auch die Fehlersuche im Einzelschrittverfahren statt. Damit dies funktioniert, muss zum einen ein für die IDE verwendbarer Zugang zum Zielsystem existieren und es müssen auch ein Compiler und ein Debugger installiert sein.

Sind die benötigten Programme auf dem Board installiert und konfiguriert, kann das Projekt in der IDE angelegt werden. Dazu muss zuerst der Remote Build Host konfiguriert werden. Dazu wird unter „Window“ der Menüpunkt „Services“ ausgewählt. Es öffnet sich links das Fenster „Services“. Unter C/C++ Build Host macht man einen Rechtsklick und wählt „add new Host“. Es öffnet sich ein Dialogfenster, in welchem im ersten Schritt nach der IP-Addresse oder dem Hostnamen gefragt wird. Im zweiten Schritt wird nach der Zugangsmethode gefragt. Hier bietet sich die SSH-Verbindung an. Nachdem Benutzername und Passwort eingegeben wurden, fängt die IDE an, den neuen Host zu überprüfen und die benötigten Programme auf dem Host zu suchen. Wenn alle Programme gefunden wurden, wird eine Zusammenfassung angezeigt, auf der die Einstellungen noch einmal geprüft werden können. Ist alles korrekt eingestellt und erkannt worden, kann der Dialog mit „Finish“ beendet werden. Der Remote Build Host ist nun konfiguriert und kann in einem Projekt verwendet werden.

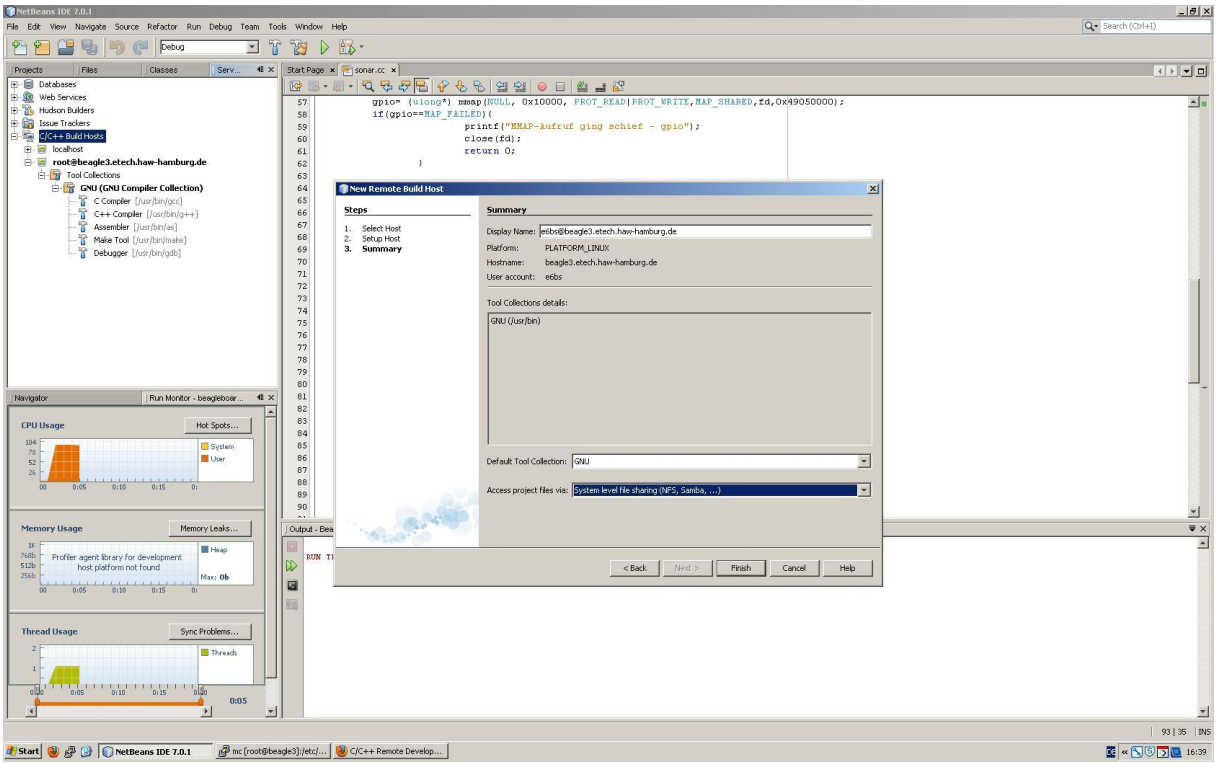

Bild 14: Zusammenfassung der Einstellungen

Nun kann ein neues Projekt angelegt werden. Dazu wählt man unter „File“ den Punkt „New Project“. Im ersten Schritt wählt man den Projekttyp aus, es existieren zahlreiche Beispielprojekte. Im zweiten Schritt wählt man den Speicherort, den Build Host und die zu verwendende Toolchain aus. Als Speicherort sollte hier die zuvor eingerichtete Sambafreigabe verwendet werden, die man entweder direkt in der Form \\servername\freigabe angibt oder der man zuvor einen Laufwerksbuchstaben zugewiesen hat. Als Build Host wählt man den zuvor definierten, also nicht „Localhost“, aus. Als Toolchain wählt man die zuvor von der IDE erkannte Toolchain aus (GNU Compiler Collection). Mit einem Klick auf „Finish“ wird der Dialog geschlossen und das Projekt angelegt.

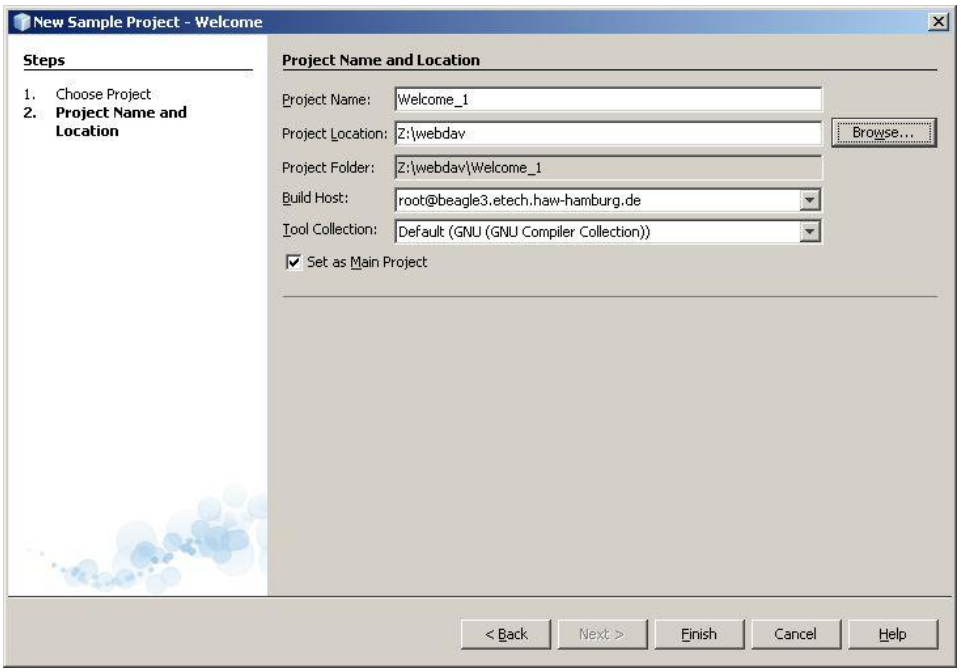

Bild 15: Zusammenfassung der Projekteinstellungen

Nachdem das Dialogfenster geschlossen wurde, ist das Projekt bereit für den ersten Kompilationsvorgang. Diesen startet man mit „Bild Main Project“ im Menüpunkt „Run“. Bei dem ersten Vorgang fragt Netbeans nach einem „Mapping“. Hier muss angegeben werden, unter welchem Pfad man auf dem zuvor definierten und ausgewähltem Remote Build Host auf das Projektverzeichniss zugreifen kann. Ist diese Abbildung der Pfade zum Projektverzeichniss korrekt gesetzt und das Dialogfeld geschlossen, startet der Kompilationsvorgang, der keine Fehlermeldungen mehr bringen sollte. Für eine genauere Schritt für Schritt Anleitung für die Konfiguration der IDE sei auf das C/C++ Remote Development Tutorial unter [9] verwiesen.

3.3 Eclipse

„Eclipse (von engl.: eclipse = Sonnenfinsternis, Finsternis, Verdunkelung) ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung für die Programmiersprache Java genutzt, aber mittlerweile wird es wegen seiner Erweiterbarkeit auch für viele andere Entwicklungsaufgaben eingesetzt. Für Eclipse gibt es eine Vielzahl sowohl quelloffener als auch kommerzieller Erweiterungen.“ [10]

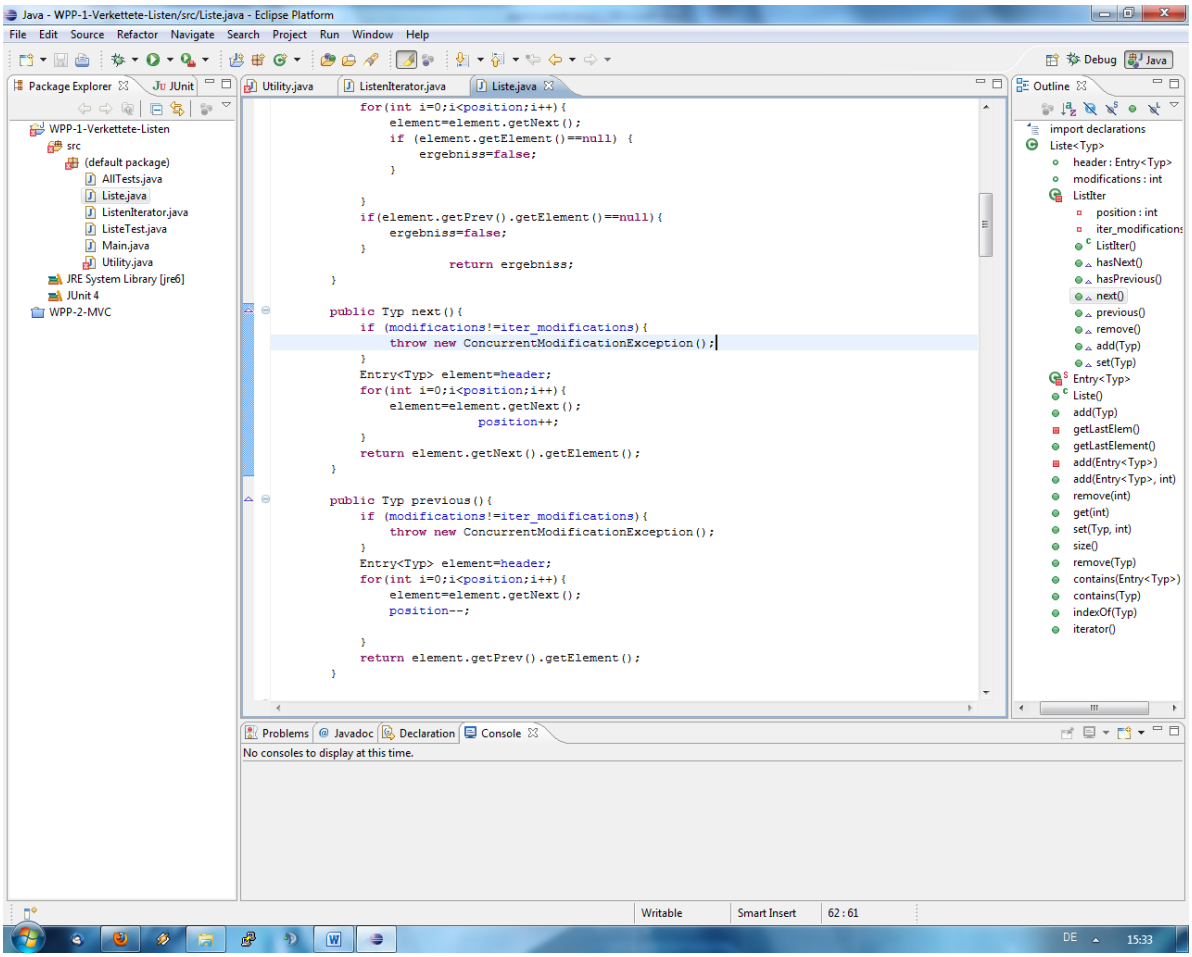

Bild 16: Eclipse Europa für die Entwicklung von Java Programmen

Im Ausbildungsbetrieb im Department Informations- und Elektrotechnik an der HAW Hamburg wird sehr viel mit Eclipse gearbeitet. Eclipse bietet von sich aus allerdings keine Unterstützung für das Kompilieren, Linken und Ausführen von Programmtext auf entfernten Rechnern. Diese Funktionalität lässt sich nur über Umwege realisieren.

Für Eclipse existiert der Remote Systems Explorer (RSE) Plugin [14]. Dies fügt der IDE eine neue Perspektive hinzu. In dieser Perspektive lässt sich das entfernte System verwalten. Man erhält eine Übersicht über die laufenden Prozesse, Zugriff auf das Dateisystem des entfernten Rechners sowie im Falle von Linux ebenfalls SSH-Zugriff. Außerdem ist es nach der Installation möglich, bei der Erstellung eines neuen Projekts die Projektverzeichnisse auf dem entfernten Host abzulegen und über den Package Explorer darauf zuzugreifen, als würden sie auf dem lokalen System liegen.

Das RSE-Plugin besteht aus zwei Komponenten: Zum einen aus dem eigentlichen Plugin für die IDE, zum anderen einer Server-Komponente, dem DStore-Server, welcher auf dem zu verwaltendem System laufen muss. Da der DStore-Server in Java geschrieben ist, muss auf dem Board zwingend eine JVM installiert sein. Hier bietet sich eine Embedded Java Version von der Firma Oracle Corporation an, die man nach einer Registrierung auf deren Internetauftritt herunterladen kann. Das Startskript ist in Perl geschrieben, daher muss

zusätzlich der Perl-Interpreter installiert sein. Als Parameter benötigt das Skript noch den Port, auf dem der Server lauschen soll. Der Server läuft anschließend mit den Berechtigungen des Benutzers, der ihn gestartet hat. Nun kann man in der IDE den Server konfigurieren und verwenden. Am Server muss man sich mit einem Benutzernamen und einem Passwort authentifizieren. Diese Kombination wird gegen die Systemdatenbank geprüft. Da dadurch ein Brute-Force-Angriff auf das Board ermöglicht wird und der Server keine Protokolle über gescheiterte Anmeldeversuche schreibt, sollte man beim Start des Servers einen Port wählen, der in der Firewall des Departments Informations- und Elektrotechnik an der HAW Hamburg gesperrt ist (siehe Kap. 1.3).

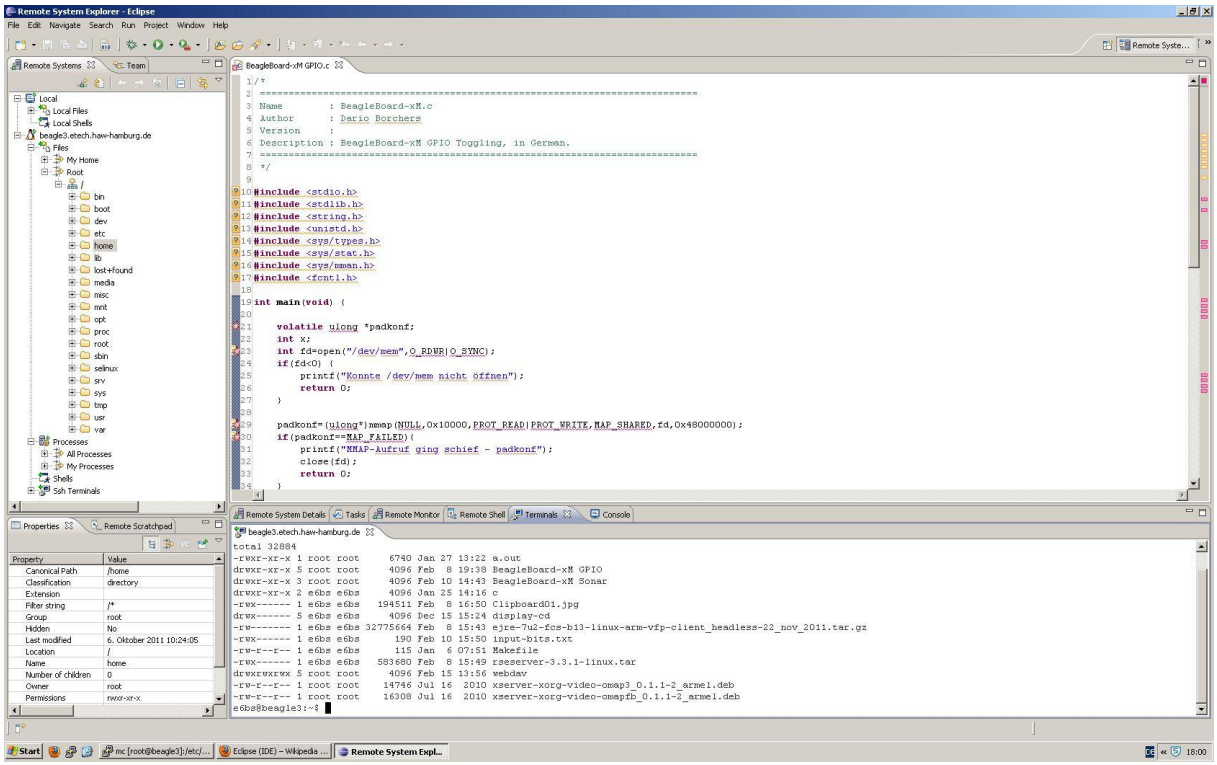

Bild 17: Remote System Explorer-Perspektive in Eclipse

Es zeigte sich allerdings, dass die allgemeine Performance der Dateiübertragung mittels des durch das RSE-Plugin registrierten RSEFS sehr gering ist. Daher empfiehlt es sich, auf andere Methoden, wie den Samba-Server oder das NFS, auszuweichen.

Man kann nun also ein Projekt auf dem Board anlegen und den Quelltext schreiben. Allerdings ist damit noch nicht der Prozess des Kompilierens, Linkens, und des Ausführens auf das Board verlagert worden. D.h. Eclipse versucht immer noch, einen lokal installierten Compiler zu starten. Findet die Entwicklung unter Windows statt, würde eine PE-Datei für MS Windows erzeugt werden, wenn kein Cross-Compiler installiert ist.

Um den Prozess des Kompilierens und des Linkens automatisch auf dem Board ausführen zu können, muss eine Netzwerkverbindung dorthin aufgebaut werden. Für Windows existiert das Programmpaket „Putty“ [18]. Dieses beinhaltet neben dem SSH-Client Putty einige weitere

Zusatzprogramme. Zuerst erstellt man sich mit dem Programm „Puttygen“ ein Schlüsselpaar, um ein passwortloses Login per SSH auf dem Board zu ermöglichen. Für die genaue Vorgehensweise zur Einrichtung der SSH-Keys sei auf [19] verwiesen. Ist das passwortlose Login korrekt konfiguriert worden, wird beim Aufbau der SSH-Verbindung nichtmehr nach einem Passwort gefragt:

```
Using username "e6bs".
Authenticating with public key "rsa-key-20120208"
Last login: Mon Mar  5 13:44:27 2012 from ds43.etech.haw-hamburg.de
e6bs@beagle3:~$
```

Listing 13: Passwortloses Login am SSH-Server mit dem Putty-Client

Die so in Putty konfigurierte Sitzung kann nun mit dem Komandozeilenprogramm „plink.exe“ verwendet werden. Plink.exe ist sehr ähnlich zum ssh-Kommando unter Linux. Mit dem Parameter „–load“ wird angegeben, welche Sitzung geladen werden soll. Hier würde also die zuvor mit den SSH-Keys definierte Sitzung eingetragen werden. Als zweiter Parameter folgt das Kommando, welches nach dem erfolgreichen Verbindungsaufbau ausgeführt werden soll. Idealerweise würde hier dann ein Aufruf des make-Kommandos stehen. Diese Kombination lässt sich dann als sog. „Builder“ in den Eigenschaften des Eclipse-Projektes eintragen, womit nun auch Eclipse die Funktionalität hat, auf entfernten Rechnern kompilieren zu können.

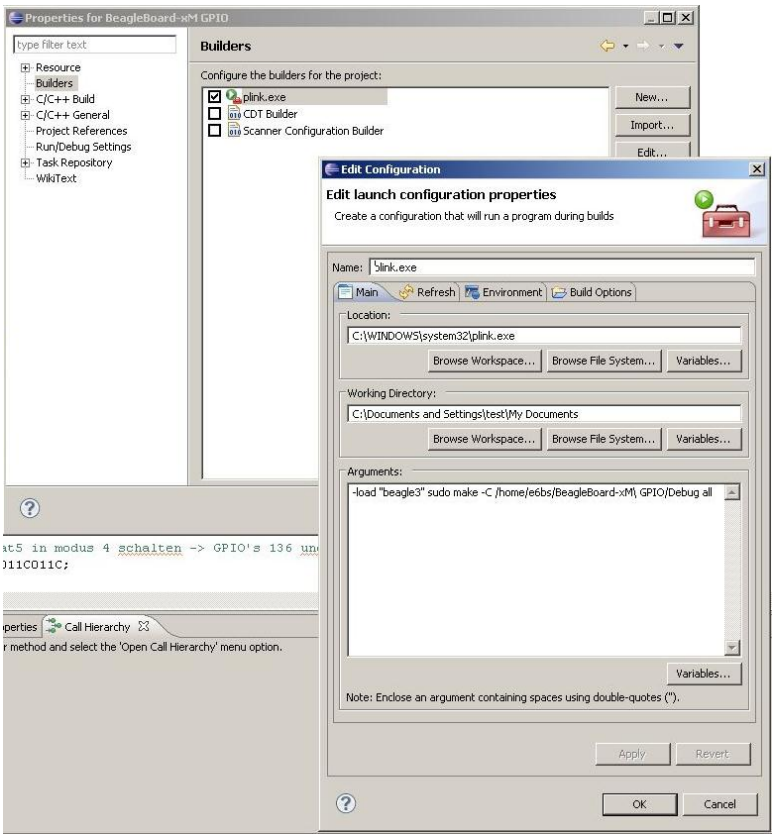

Bild 18: Konfiguration der „Builder“ eines Projektes in Eclipse

Ein Debugging kann mit dem „gdbserver“ stattfinden, wofür im Internet zahlreiche Anleitungen existieren. Dadurch, dass man die Konfiguration der „Builder“ verändert hat, funktioniert das „Parsing“ des Quelltextes nicht mehr, sodass die IDE keine Aussage mehr treffen kann, ob eine Zeile Quelltext korrekt ist oder nicht. Dies muss dann im Ausgabefenster des Compilers mitverfolgt werden.

Aufgrund der Einfachheit der Konfiguration und der guten Verzahnung mit den verwendeten Programmen soll für diese Arbeit Netbeans verwendet werden. Der Konfigurationsaufwand, der bei Eclipse bei jedem neuen Projekt erneut durchgeführt werden muss, ist um ein vielfaches höher, als der Aufwand, welcher investiert werden muss, um bei Netbeans gleiche Resultate zu erzielen.

4. Ultraschall

In diesem Kapitel wird zuerst ein kurzer Überblick über die theoretischen Grundlagen des Ultraschalls und des darauf beruhendem Prinzips der Entfernungsmessung gegeben. Anschließend werden die zur Verfügung stehenden Sensoren vorgestellt. Dabei handelt es sich zum einen um ein Fertiges Modul mit eigenem MCU und zum anderen um eine Kombination aus einem Eigenbau der HAW sowie einem digitalem Temperatursensor. Das Messen der Temperatur ist notwendig, um die Schallgeschwindigkeit berechnen zu können.

4.1 Schallwellen

„Schallwellen sind mechanische Longitudinalwellen. Ausgehend von der Schallquelle, einem Schwingendem Körper, breiten sie sich in Festkörpern, Flüssigkeiten und Gasen in der Form von Druckschwankungen (Druckwellen) aus.“ [12]

Das menschliche Gehör kann Frequenzen zwischen ca. 16Hz und ca. 20kHz wahrnehmen [12]. Unter der Wahrnehmungsschwelle liegende Frequenzen werden als Infraschall, darüber liegende als Ultra- bzw. Hyperschall bezeichnet. Ultraschall umfasst einen Frequenzbereich von ca. 20kHz bis 10GHz, Hyperschall umfasst die Frequenzen zwischen 10GHz und 10THz. Schallquellen können alle schwingenden Körper sein die Schallwellen abstrahlen können, ganz gleich in welchem Aggregatszustand sie sich gerade befinden.

Treffen Schallwellen auf ein Hindernis, d.h.: auf den Übergang zweier Medien mit verschiedenen Dichten, wird ein Teil der Energie der Schallwellen vom Hindernis absorbiert, und die Schallwellen werden in ihrer Amplitude gedämpft reflektiert. Für den Hörer entstehen sog. Spiegelquellen. Am Punkt des Auftreffens der Schallwellen gilt der aus der Optik bekannte Satz „Einfallswinkel gleich Reflexionswinkel“ (Reflexionsgesetz, [12]), sofern die Oberfläche (Reflexionsfläche) im Vergleich zu der Wellenlänge groß ist.

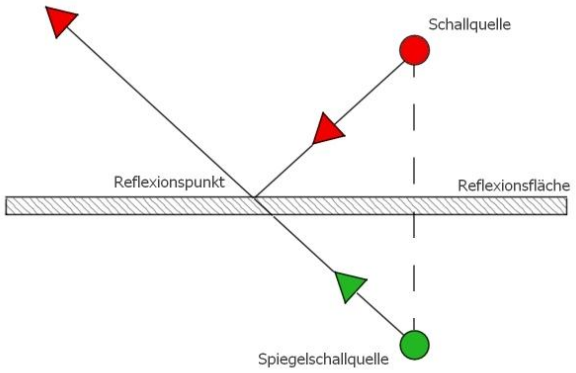

Bild 19: Reflexion von Schallwellen [13]

4.2 Schallgeschwindigkeit

Die Geschwindigkeit, mit der sich die Schallwellen in Gasen ausbreiten, ist abhängig von der Temperatur des Gases, von einer Gaskonstante und dem sog. Isentropenexponenten. Die Schallgeschwindigkeit in einem Gas ist allerdings nicht von dessen Druck abhängig.

$$c_0 = \sqrt{\varkappa R_i T} \qquad \textbf{Gl. 3}$$

wobei gilt:

$\varkappa$ = Isentropenexponent aus Tabellenwerk, beinhaltet den Gasdruck

R_i = spezielle Gaskonstante aus Tabelle

T = Temperatur des Gases in Kelvin

Wenn man in Gl. 3 die Werte für trockene Luft bei 0° Celsius einsetzt, erhält man:

$$c_0 = \sqrt{1{,}4 \cdot 287 \frac{J}{kg \cdot K} \cdot 273K} = 331{,}2 \frac{m}{s} \qquad \textbf{Gl. 4}$$

Die Feuchtigkeit der Luft hat lt. [12] allerdings nur einen sehr geringen Einfluss auf die Ausbreitungsgeschwindigkeit von Schallwellen.

Aus dieser Gleichung lässt sich die Beziehung zwischen Änderung der Schallgeschwindigkeit und Änderung der Temperatur des Gases Ableiten:

$$\frac{c}{331{,}2 \frac{m}{s}} = \frac{\sqrt{\varkappa R_i T}}{\sqrt{\varkappa R_i 273K}}$$

$$c = 331{,}2 \frac{m}{s} \cdot \sqrt{\frac{T}{273K}}$$

mit $T_0 = 273K$ und $T = T_0 + \Delta t$ folgt:

$$c = 331{,}2 \frac{m}{s} \cdot \sqrt{1 + \frac{\Delta t}{273K}}$$

Diese Gleichung lässt sich lt. [12] mit hinreichender Genauigkeit zu

$$c = (331{,}2 + 0{,}6 \cdot \frac{\Delta t}{°C}) \frac{m}{s} \qquad \text{Gl. 5}$$

vereinfachen. Für trockene Luft und einer Raumtemperatur von 20°C ergibt sich demnach eine Schallgeschwindigkeit von

$$c = \left(331{,}2 + 0{,}6 \cdot \frac{20°C}{°C}\right) \frac{m}{s} = 343{,}2 \frac{m}{s} \qquad \text{Gl. 6}$$

Die Temperatur des Gases, in dem sich die Schallwellen ausbreiten hat also erheblichen Einfluss auf die Ausbreitungsgeschwindigkeit. Daher sollte die Temperatur nicht lediglich angenommen, sondern gemessen werden.

4.3 Messverfahren

Um eine Entfernung mit Hilfe von Ultraschall bestimmen zu können, muss die aktuelle Schallgeschwindigkeit bekannt sein. Da Schallwellen von Hindernissen reflektiert werden, reicht es aus, einige Schallwellen zu erzeugen und die Zeit zu messen, nach der die reflektierten Wellen, das Echo, eintreffen.

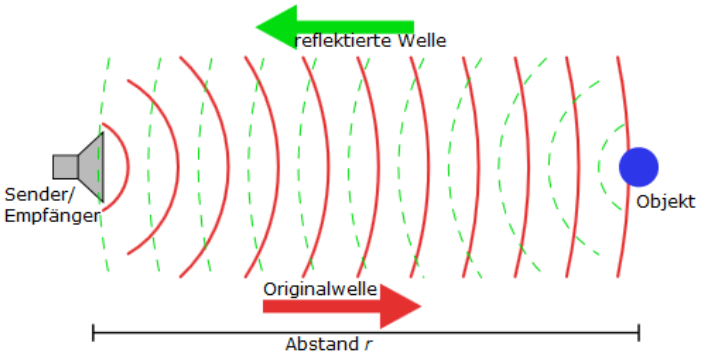

Bild 20: Prinzip der Entfernungsmessung [15]

Da die gemessene Laufzeit der Wellen den Weg zum Reflektor und zurück beinhaltet, muss sie durch 2,0 dividiert werden, um die Laufzweit zwischen Sender und Reflektor zu erhalten. Um die hinlaufende und die dazugehörige, rücklaufende Welle identifizieren zu können, werden einzelne Pulse ausgesendet. Man misst also die Pulslaufzeit.

$$r = \frac{c \cdot t}{2} \qquad \text{Gl. 7}$$

wobei gilt:

r: Abstand zwischen Sender und Reflektor

c: Schallgeschwindigkeit

t: gemessene Laufzeit der Schallwellen

4.4 Sensoren

4.4.1 LV-MaxSonar-EZ1

Das LV-MaxSonar-EZ1 ist ein komplettes Sensormodul, welches ohne weitere äußere Beschaltung sofort Einsatzbereit ist. Das Modul verfügt über einen eigenen MCU, welcher die Messungen vornimmt. Dieser wurde für den Innenbereich konzipiert und verwendet Ultraschall mit einer Frequenz von 42 KHz. Die Betriebsspannung kann zwischen 2,5V und 5,5V variieren [17].

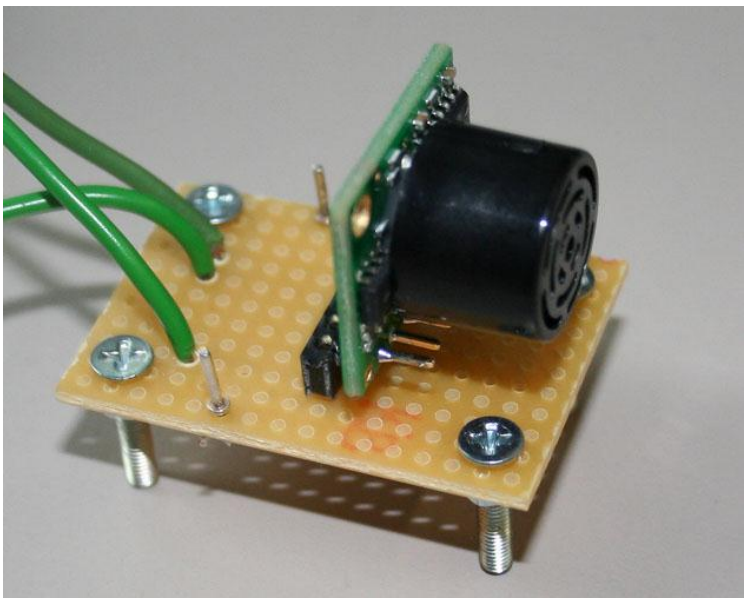

Bild 21: LV-MaxSonar-EZ1 auf einer Trägerplatine

Als Ausgänge stehen eine RS232-Schnittstelle (mit CMOS-Pegeln, trotzdem wird kein Pegelwandler benötigt), eine analoge Spannung mit einer Auflösung von $\frac{V_{cc}}{\frac{512}{Zoll}}$ (ugs.: „V_{cc} Fünfhundertzwölftel pro Zoll“), sowie eine Pulsweitenmodulation mit einer Auflösung von $147\frac{\mu s}{\text{Zoll}}$ zur Verfügung. Wird das Modul im freilaufenden Modus betrieben, wird alle 50 Millisekunden eine Messung ausgeführt und das Ergebnis steht nach der Auswertung an den

Ausgängen bereit. Misst man also eine Pulsweite von 4,998ms, so entspricht dies einer Entfernung von 34 Zoll oder 0,864 Metern.

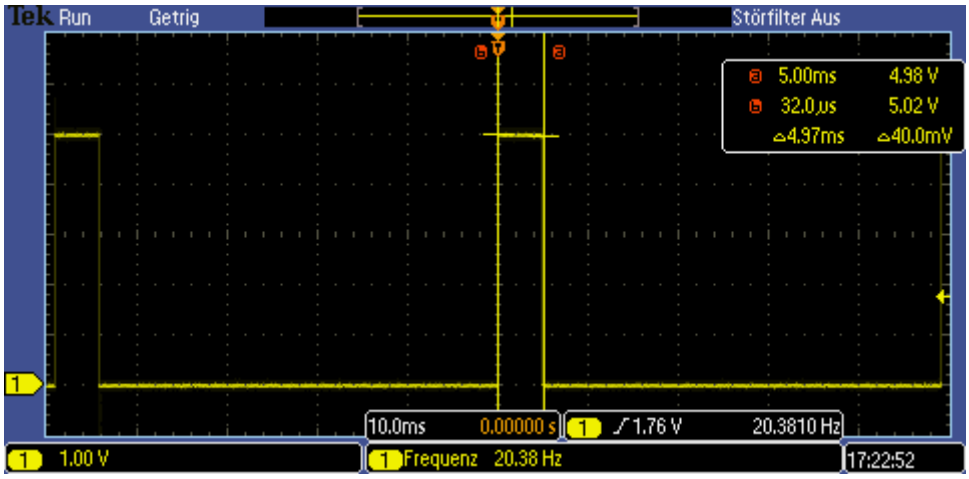

Bild 22: Ausgangssignal des Sensormoduls

Um das Ausgangssignal dieses Moduls auszuwerten, muss also lediglich im µS-Raster zwischen steigender und der fallenden Flanke des Signals die verstreichende Zeit mitgezählt werden und dieses Ergebnis dann durch 147,0 dividiert werden, um die Distanz zu bestimmen. Ggf. muss das Ergebnis noch einmal mit dem Faktor 2,54 multipliziert werden, um das Ergebnis in Zentimeter anstelle in Zoll zu erhalten.

4.4.2 Eigenbau der HAW Hamburg

Als zweite Variante steht für die Ultraschallmessungen eine Eigenbaulösung der HAW Hamburg zur Verfügung. Auf der Sendeseite besteht diese aus einem als frei laufendem Oszillator konfiguriertem NE555 Zeitgeberbaustein, einer Sendekapsel und einer Ladungspumpe.

Da im Labor der HAW ausschließlich 5,0V-Logik verwendet wird, sollten auch diese Module mit dieser Betriebsspannung funktionieren (gefordert ist also Laborkompatibilität), damit das Modul auf einfache Weise an die bestehende Peripherie im Labor für Informationstechnik angeschlossen werden kann.

Bild 23: Ultraschall-Sender und -Empfänger der HAW

Die Ladungspumpe erzeugt aus den angelegten 5,0V zuerst 15,0V, die als Betriebsspannung für den NE555 verwendet werden. Dies ist notwendig, damit das Ausgangssignal eine ausreichende Amplitude hat, um eine möglichst große Auslenkung der Membran im der Sendekapsel zu erzeugen. Die Frequenz, die der NE555 erzeugt, ist über ein Potentiometer manuell auf die Eigenfrequenz der Sendekapsel abgeglichen worden. Zwar erzeugt der NE555-Baustein Rechteckimpulse, diese werden aber dennoch ungefiltert, also mit den Harmonischen der Grundfrequenz, auf die Sendekapsel gegeben.

Auf der Empfangsseite befindet sich ebenfalls eine solche Kapsel, die durch den Sender zum Schwingen angeregt wird. Das so entstandene Signal wird zunächst auf einen aus zwei Operationsverstärkern bestehenden, schmalbandigen Bandpass gegeben, welcher als Mittenfrequenz die Eigenfrequenz der Sendekapsel hat. Dadurch ist sichergestellt, dass der Empfänger nicht durch Nebengeräusche angeregt werden kann. Nachgeschaltet ist eine Gleichrichterschaltung, welche das Signal gleichrichtet und glättet. Sowie der Empfänger korrekt angeregt wird, entsteht hier also ein High-Pegel. Dieser wird auf einen Spannungsteiler gegeben, um ihn auf 5,0V-TTL-Pegel zu bringen.

Das Messprinzip ist also, den Sender mit einem Puls zu aktivieren, und die Zeit zu messen, nach der der Empfänger von Low auf High umschaltet. Da es sich hier nicht um eine reflektierte Schalwelle handelt, braucht die gemessene Zeit auch nicht durch den Faktor 2,0 dividiert zu werden. Diese Zeit kann also direkt in Gleichung 7 eingesetzt werden.

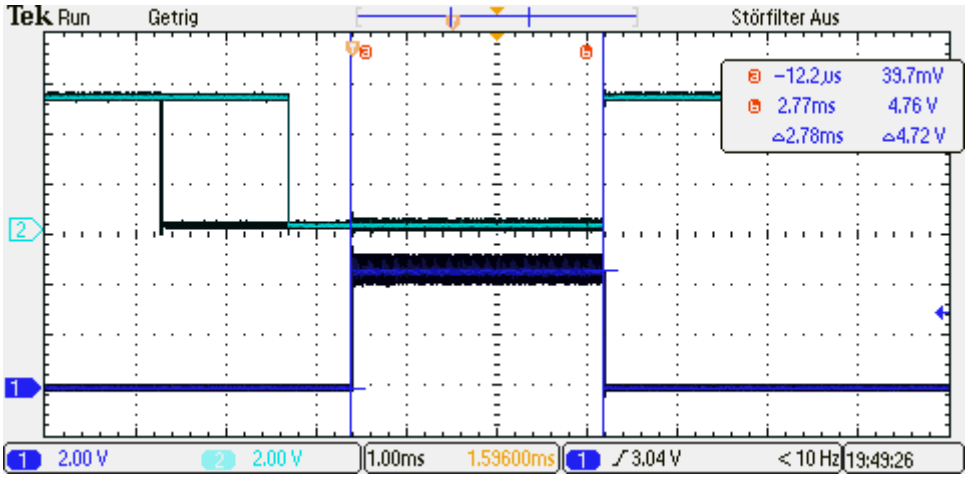

Bild 24: Messung der Verzögerung

4.4.3 Temperatursensor SHT15

Der SHT15-Sensor der Firma Sensiron AG ist ein digitaler Temperatur- und Luftfeuchtesensor. Zur Kommunikation werden eine Takt- und eine Datenleitung benötigt. Lt. [23] kann man ihn zusammen mit anderer Peripherie an den I²C-Bus anschließen, ohne dass es zu Komplikationen kommt. Allerdings verwendet dieser Sensor kein I²C-Protokoll. Daher lässt er sich auch nicht über das Kernel-Interface des Linux-Betriebssystems auswerten. Stattdessen muss das Kommunikationsprotokoll manuell implementiert und am Bus zwischen diesem und dem I²C-Protokoll umgeschaltet werden.

Bild 25: SHT15-Sensor auf einer Trägerplatine [24]

Es existiert ein Beispielprogrammtext von der Firma Sensiron AG für einen 80C51-MCU, der einen solchen Sensor auswertet. Dieser wurde im Rahmen dieser Arbeit auf das BeagleBoard-xM portiert. Zunächst wird der Sensor neu gestartet, danach erfolgt eine „Transmission-Start-Sequence“. Im Anschluss daran wird das Kommando 0x3 geschrieben. Dieses Kommando startet die Temperaturmessung mit 14 Bit Auflösung. Während der Messung bleibt die Datenleitung auf „high“, außerdem setzt der Takt während der Messung aus.

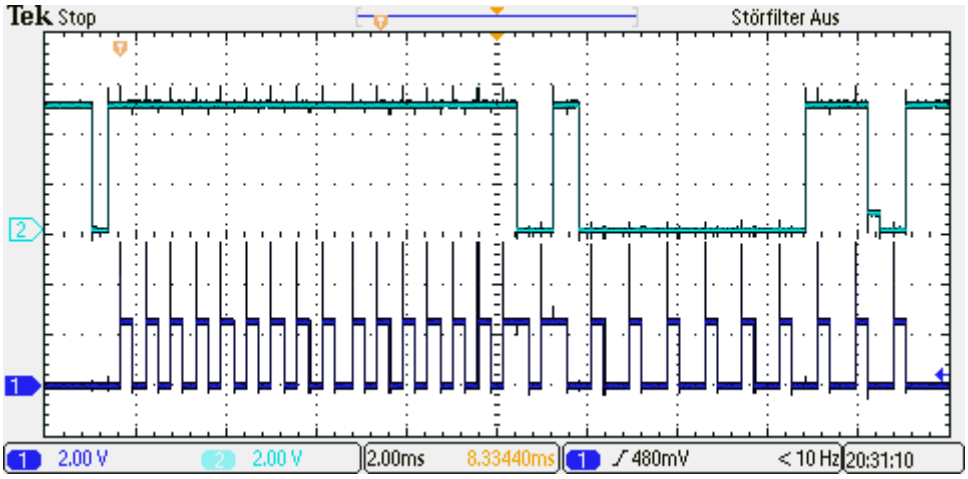

Bild 26: Reset, Transmission Start und Kommando 0x3

Die Messung dauert lt. [23] ca. 80ms. Wenn die Messung durch den Sensor beendet wurde und das Ergebnis bereit steht, wechselt der Sensor den Pegel der Datenleitung auf „low“. Nun setzt der Takt wieder ein. Mit jedem Takt wird nun ein Bit des Ergebnisses auf die Datenleitung gelegt, das höchstwertigste Bit (MSB) zuerst, das niederwertigste Bit (LSB) zuletzt. Alle acht Bit muss ein sog. Acknowledge-Bit gesendet werden. Das dritte Byte, das auf diese Art abgefragt werden kann, ist eine Prüfsumme für die beiden zuvor gesendeten Bytes.

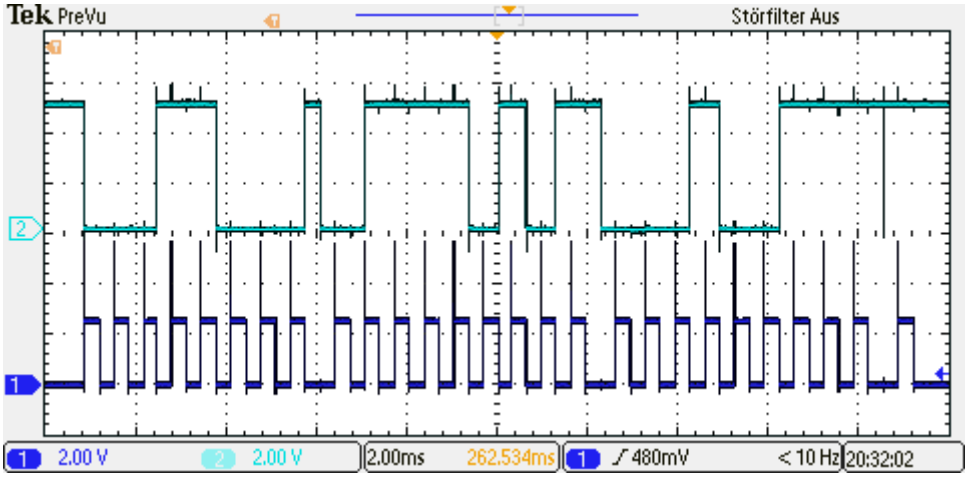

Bild 27: Auslesen des Ergebnisses

Für die genaue Vorgehensweise sei auf das Datenblatt des Sensors ([23]), den Beispielprogrammtext der Firma Sensiron AG ([25]) und den erstellten Programmtext im Anhang A1.4 verwiesen.

Nach Auswertung dieses Sensors steht nun die aktuelle Temperatur zur Verfügung, die in Gl.6 eingesetzt werden kann, um die genaue Schallgeschwindigkeit zu bestimmen, die in Gl.7 eingesetzt werden kann. Dadurch, dass die Raumtemperatur nicht mehr angenommen wird, sondern gemessen werden kann, ist eine Temperaturkompensation der Messwerte möglich.

5. Programmierung der digitalen Ein- und Ausgänge

In diesem Kapitel wird beschrieben, auf welche Art die digitalen Ausgänge (GPIO‘s) am Expansion-Header des BeagleBoard-xM’s programmiert werden können. Der Linux-Kernel stellt über ein Modul eine Schnittstelle dafür bereit, andererseits existiert die Möglichkeit, direkt die entsprechenden Register zu manipulieren. Im ersten Schritt wird die Umsetzung der Pegel des Boards auf die im Labor für Informationstechnik verwendeten 5,0V-Pegel erläutert. Im zweiten Schritt wird ein grundlegendes Verständnis für die Funktionsweise der GPIO’s geschaffen. Im dritten Schritt werden schliesßlich die beiden genannten Möglichkeiten der Programmierung miteinander verglichen.

5.1 Pegelumsetzer

Da das BeagleBoard-xM auf der Leiterplatte nicht mit 5,0V- oder 3,3V-Pegeln, sondern mit 1,8V-Pegeln arbeitet, ist ein Pegelumsetzer notwendig, wenn man die in den Laboren der HAW Hamburg verwendete 5,0V-Peripherie an das BeagleBoard-xM anschließen möchte. Von der HAW wurde eine Leiterplatte mit Pegelumsetzern von der Firma TI gestellt (Siehe Bild 1, rechts im Bild). Unter keinen Umständen darf eine Spannung von $U > 1{,}8V$ an die Pins am Expansion-Header am Board angelegt werden, da dies in sehr kurzer Zeit zu irreparablen Schäden am Board führt.

Auf der Leiterplatte werden für die Umsetzung TXS0104E-Chips der Firma TI verwendet. Diese Chips benötigen lt. [20] kein Signal zur Steuerung der Richtung des Datenflusses. Außerdem haben sie interne PullUp-Widerstände an beiden Eingängen von jeweils 10KΩ. Daher lassen sich auch Open-Kollektor- oder Open-Drain-Ausgänge, wie sie beispielsweise beim I²C-Bus verwendet werden, an diese Pegelumsetzer anschließen. Außerdem muss keine Startreihenfolge eingehalten werden, d.h. es ist gleich, ob V_{CCA} oder V_{CCB} zuerst angelegt wird. Das Board kann also normal gestartet werden und es kann zu einem späteren Zeitpunkt Peripherie angeschlossen, und eine zusätzliche Spannungsquelle aktiviert und mit dem Pegelumsetzer verbunden werden.

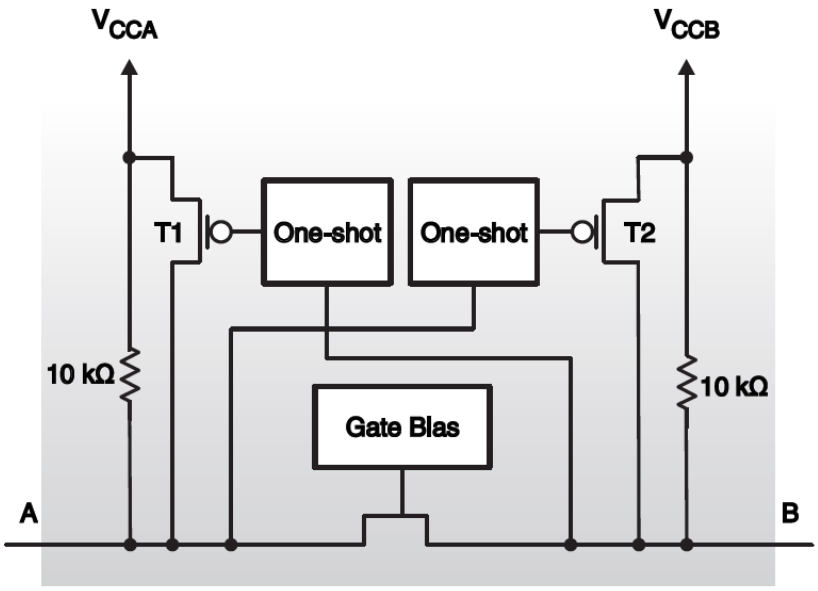

Bild 28: Architektur einer Zelle im TXS0104E [20]

Auf der zur Verfügung gestellten Leiterplatte ist der Output-Enable-Eingang der Chips mit V_{CCA}verbunden, sodass die Ausgänge ohne weitere Konfiguration aktiviert sind.

5.2 GPIO und Pinmux

Moderne Mikrocontroller haben mehr extern nutzbare Signale als Beinchen bzw. Lotkugeln am Gehäuse. Es kommt daher zu einer Mehrfachbelegung der Beinchen oder Lotkugeln mit Signalen. Wenn nun zwei Funktionseinheiten das gleiche Beinchen für verschiedene Signale verwenden wollen, kommt es zum Konflikt (Bsp.: Port C und JTAG am Atmel Atmega16).

Bei im Vergleich zu diesem MCU „kleineren“ Mikrocontrollern können daher undefinierte Zustände auftreten, bei dem hier verwendeten MCU ist dies nicht der Fall. Dieser hat ein sog. „System Control Module“, in welchem die Konfiguration jeder einzelnen Lotkugel am Gehäuse hinterlegt ist ([16], Kap. 13.4.4). Neben der Konfiguration des Modus lassen sich auch noch PullUp- und PullDown-Widerstände zuschalten. Außerdem lässt sich hier die Ein- und Ausgabe auf diesem Pin gezielt ein- und ausschalten. Das SCM ist mit dem zentralen L4-Bus des MCU verbunden.

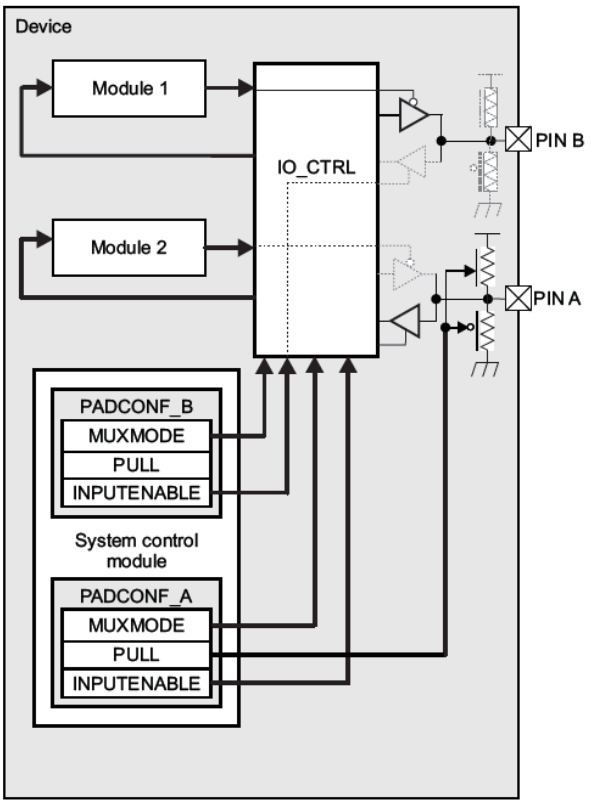

Bild 29: Schematisches Diagramm der möglichen Pin-Konfiguration [16]

Für jede Lotkugel am Gehäuse gibt es ein 16-Bit-Register, in welchem die genaue Konfiguration hinterlegt wird. Jeweils zwei dieser Register werden zu einem 32-Bit-Register zusammengefasst.

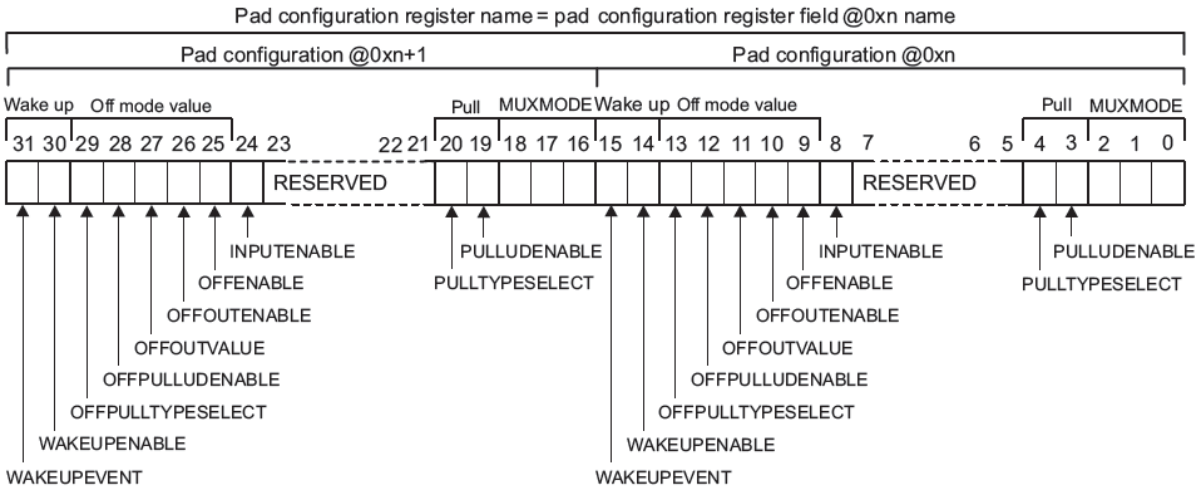

Bild 30: Konfigurationsregister für einzelne Lotkugeln [16]

Um eine Lotkugel als GPIO verwenden zu können, muss diese lt. [22], Tabelle 22 auf Seite 108, in den Modus 4 (0b100) geschaltet werden. Dadurch wird einer der GPIO-Blöcke mit dieser Lotkugel verbunden.

Der MCU hat insgesamt sechs GPIO-Blöcke mit jeweils 32 Pins. Diese Blöcke sind direkt mit dem zentralen L4-Bus verbunden. Jeder einzelne Block kann im GPIO- oder im CPU-Subsystem einen Interrupt auslösen. Zusätzlich zum Haupttakt des Moduls steht noch ein 32KHz-Takt für die Entprellung von Schaltern zur Verfügung.

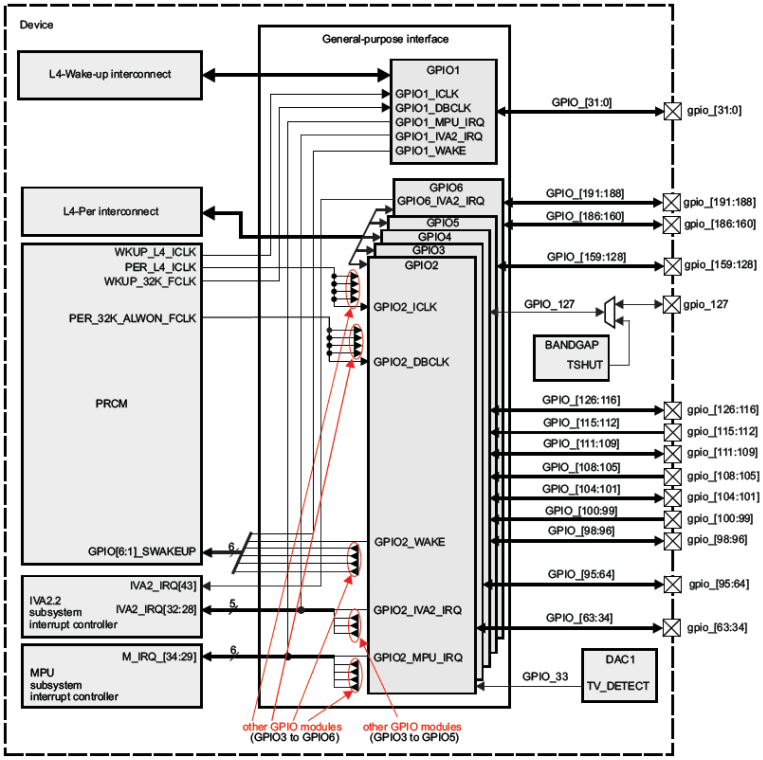

Bild 31: Übersicht über das GPIO-Modul [16]

Um einen Pin verwenden zu können, muss zunächst die Richtung des Datenflusses festgelegt werden (input/output). Dies wird im GPIO_OE-Register des jeweiligen GPIO-Blocks festgelegt. Nach einem (Neu-)Start des MCU‘s steht in diesem Register der Wert 0xFFFFFFFF. Damit sind alle Pins als Ausgang definiert. Um einen Pin als Eingang nutzen zu können, muss das zu diesem Pin gehörende Bit im GPIO_OE-Register auf null gesetzt werden. Wird ein Pin als Eingang verwendet, kann nun das dazugehörige Bit im DataIn-Register des GPIO-Blocks gelesen werden. Um den Pin als Ausgang zu verwenden, gibt es bei diesem MCU eine Besonderheit: man schreibt nicht ein Bitmuster in das DataOut-Register, sondern verwendet stattdessen sog. Set- und Clear-Register, um das DataOut-Register zu manipulieren. Man gibt also explizit an, welche Bits gesetzt bzw. zurückgesetzt werden sollen.

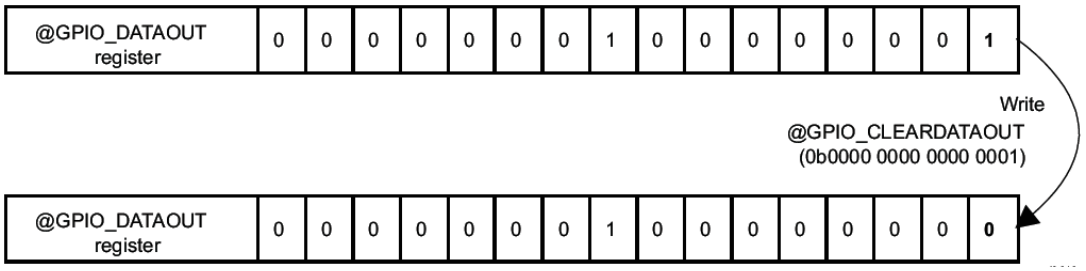

Bild 32: Bit zurücksetzen [16]

5.3 Programmierung der GPIOs

5.3.1 Kernel-Interface im SysFS

Der Linux-Kernel stellt über ein Modul eine fertige Schnittstelle zu den GPIO's in UNIX-Manier („Alles ist eine Datei") im Verzeichniss /sys/class/gpio/ bereit. Dort findet man die sechs GPIO-Blöcke abgebildet auf Dateien:

```
e6bs@beagle3:~$ ls -l /sys/class/gpio/
total 0
--w------- 1 root root 4096 Jan  1  1970 export
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip128 -> ../../devices/virtual/gpio/gpiochip128
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip160 -> ../../devices/virtual/gpio/gpiochip160
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip192 ->
../../devices/platform/omap/omap_i2c.1/i2c-1/1-0049/twl4030_gpio/gpio/gpiochip192
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip32 -> ../../devices/virtual/gpio/gpiochip32
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip64 -> ../../devices/virtual/gpio/gpiochip64
lrwxrwxrwx 1 root root    0 Jan  1  1970 gpiochip96 -> ../../devices/virtual/gpio/gpiochip96
--w------- 1 root root 4096 Jan  1  1970 unexport
e6bs@beagle3:~$
```

Listing 14: Kernel-Interface zu den GPIO's

Um einen GPIO-Pin verwenden zu können, muss dieser zunächst „exportiert" werden. Dies schaltet die entsprechende Lotkugel am SoC in den Modus 4 (siehe oben). Danach muss die Datenrichtung (Input oder Output) festgelegt werden. Nun kann entweder der Pegel gesetzt, oder aber gelesen werden.

```
root@beagle3:/sys/class/gpio# echo "136" > export
root@beagle3:/sys/class/gpio# cd gpio136
root@beagle3:/sys/class/gpio/gpio136# echo in > direction
root@beagle3:/sys/class/gpio/gpio136# cat value
1
root@beagle3:/sys/class/gpio/gpio136# cd ..
root@beagle3:/sys/class/gpio# echo 136 > unexport
root@beagle3:/sys/class/gpio#
```

Listing 15: Verwendung des Kernel-Interfaces

Somit steht eine sehr einfach zu verwendende Schnittstelle zu den GPIO's zur Verfügung. Diese Einfachheit hat allerdings auch zwei sehr schwerwiegende Nachteile.

Liest man in einem Programm fortlaufend den Zustand eines Pins ein, so greifen diverse Zwischenspeichermechanismen (Caching) des Dateisystems. Die Idee dieser Mechanismen ist

es, Lesezugriffe auf Dateien zu beschleunigen, indem häufig gelesene Dateien nicht vom dem angeschlossenem, langsamen Massenspeicher (Festplatte, Diskette, etc.), sondern aus dem schnellen, flüchtigem Arbeitsspeicher (RAM) gelesen werden. Dazu wird eine Kopie dieser Datei in dem RAM abgelegt. Es findet allerdings keine Prüfung statt, ob etwas an der ursprünglichen Datei auf dem Massenspeicher geändert wurde. Wird also die Datei, welche den Zustand eines GPIO-Pins repräsentiert, fortlaufend gelesen, so wird die Kopie im RAM, aber nicht das DATAIN-Register des jeweiligen GPIO-Blocks gelesen. Man bekommt also fortlaufend den Zustand, den der Pin beim ersten Lesen hatte, zurück geliefert. Es spielt keine Rolle, ob sich zwischenzeitlich etwas an dem Zustand des Pins geändert hat. Eine Abhilfe wäre es, sofern man in C Programmiert, den Dateideskriptor zu schließen und neu zu erzeugen. Dies erzeugt eine neue Kopie der Datei, welche den Pin repräsentiert, wobei der Zustand des Pins neu eingelesen wird. Allerdings bedeutet dieses Vorgehen einen erheblichen Geschwindigkeitsverlust.

Der zweite Nachteil ist die allgemeine Performance eines Dateisystemtreibers. Um aus dem Linux-Userland mit einem entsprechenden Befehl einem Pin einen anderen Zustand aufzwingen zu können, sind sehr viele Schichten Software notwendig. In diesen Schichten finden außerdem diverse Prüfungen (Prüfung der Berechtigungen im Dateisystem, Prüfung, ob der Pin bereits exportiert wurde, Prüfung, ob die Datei durch einen anderen Prozess gesperrt ist etc.) statt. Dieses Vorgehen kostet sehr viele CPU-Zyklen. Um die Geschwindigkeit dieser Schnittstelle zu messen, wurde ein Programm in der Sprache C geschrieben:

```
#include "stdio.h"
#include "unistd.h"

int main(){
    //pin "exportieren"
    FILE *fp=fopen("/sys/class/gpio/export","w");
    fprintf(fp,"161");
    fclose(fp);
    //richtung setzen
    fp=fopen("/sys/class/gpio/gpio161/direction","w");
    fprintf(fp,"out");
    fclose(fp);
    //pin toggeln
    fp=fopen("/sys/class/gpio/gpio161/value","w");
    while(1){
       fseek(fp,0,SEEK_SET);
         fprintf(fp,"0");
       fseek(fp,0,SEEK_SET);
         fprintf(fp,"1");
         }
    fclose(fp);
    //pin "deexportieren"
    fp=fopen("/sys/class/gpio/unexport","w");
    fprintf(fp,"161");
    fclose(fp);
    return 0;
}
```

Listing 16: Pintoggling per File-IO

Wenn man dieses Programm mit normaler Prozesspriorität laufen lässt, so ergibt sich das folgende Bild auf dem Oszilloskop:

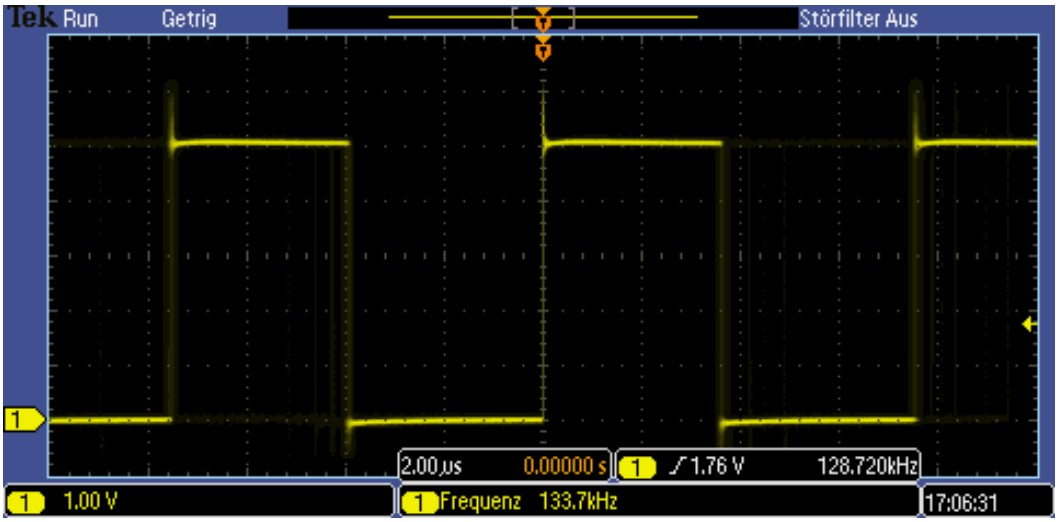

Bild 33: Pintoggling per File-IO

Obwohl die CPU mit 1,0GHz läuft, ist man mit der Kernel-Schnittstelle im SysFS gerade einmal in der Lage, zuverlässig alle 3,9µS einen Pin umzuschalten, was einer Frequenz von ca. 133KHz entspricht.

Sieht man dieses Verhalten in Kombination mit dem Caching des Dateisystems, so lässt sich sagen, dass die Kernel-Schnittstelle nicht nur zu langsam, sondern unbrauchbar ist, um die o.g. Ultraschallsensoren auszuwerten. Allerdings lässt sich auf diese Weise sehr einfach ein Pin schalten, um beispielsweise eine Signallampe an einer Maschine ein- oder auszuschalten.

5.3.2 Direkte Manipulation der Register

Lt. [16] ist der Adressbereich der zentralen L3- und L4-Interconnects (siehe Bild 3) in die Speicherkarte der verwendeten ARM-CPU eingeblendet. Die SFR's der o.g. Module lassen sich daher wie gewöhnliche Speicheradressen auslesen und beschreiben. Ein Prozess im Linux-Userland hat aus Sicherheits- und Stabilitätsgründen erst einmal keinerlei direkten Zugriff auf die Speicherkarte der CPU. Stattdessen werden im Kontext eines Prozesses virtuelle Adressen verwendet, die bei einem Zugriff auf Variablen etc. in reale Speicheradressen umgerechnet werden.

Allerdings existiert die Möglichkeit, mittels der mmap()-Funktion bestimmte reale auf virtuelle Adressen abzubilden. Man erzeugt einen Dateideskriptor auf /dev/mem und ruft anschließend die mmap()-Funktion mit diesem als Parameter auf. Um den Dateideskriptor erzeugen zu können, werden allerdings Super-User-Privilegien benötigt (siehe Systemsicherheit, Kap. 1.3).

```
volatile ulong *padkonf;
int fd=open("/dev/mem",O_RDWR|O_SYNC);
if(fd<0) {
        printf("Konnte /dev/mem nicht öffnen");
        return -1;
        }

padkonf=(ulong*)mmap(NULL,0x10000,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x48000000);
if(padkonf==MAP_FAILED){
        printf("MMAP-Aufruf ging schief - padkonf");
        close(fd);
        return -1;
        }
```

Listing 17: Erzeugen eines Zeigers auf die Speicherkarte

Durch diesen Aufruf bildet man die Speicheradressen 0x48000000 bis 0x48010000 auf die Variable „padkonf“ ab. Idealerweise wählt man hier also ein Fenster, in dem die zu bearbeitenden SFR’s in die Speicherkarte der CPU eingeblendet sind. Die Konfigurationsregister für die Lotkugeln AE4 und AH3 (GPIO 136 und 137) sind an der Speicheradresse 0x48002164 eingeblendet. Durch den Aufruf

```
padkonf[0x2164/4]=0x011C011C;
```

lässt sich der angegebene Wert in das entsprechende Padconf-Register schreiben. Durch diesen Wert werden die beiden Lotkugeln in den Modus vier (GPIO) geschaltet. Auf die gleiche Weise lassen sich auch die Set-Dataout- und Clear-Dataout-Register sowie die OutputEnable-Register der einzelnen GPIO-Blöcke Manipulieren und auslesen.

```
gpios= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
if(gpios==MAP_FAILED){
        printf("MMAP-Aufruf ging schief - gpio");
        close(fd);
        return -1;
        }

x=0|0x00000100;

//permanentes Umschalten der Pins
while(1){
        gpios[0x6090/4]=x; //gpio5-cleardataout-register
        gpios[0x6094/4]=x; //gpio5-setdataout-register
}
```

Listing 18: Pintoggling per SFR

Die hier vorgestellte Methode ist im Vergleich zu der Schnittstelle im SysFS zwar erheblich komplizierter, allerdings auch erheblich schneller im Umschalten der Zustände oder im Abfragen der Pins. Außerdem bietet sie die Möglichkeit, wieder direkt mit Bitmaskierungen zu arbeiten und es greift kein Caching-Mechanismus. Es muss bei dieser Methode allerdings sehr sorgfältig gearbeitet werden, da Teile des Netzwerkchips mit einigen Pins des GPIO-Block 5 verbunden sind. Werden diese Pins umkonfiguriert, kann es vorkommen, dass die Netzwerkschnittstelle nicht mehr betrieben werden kann (Siehe Kap.3).

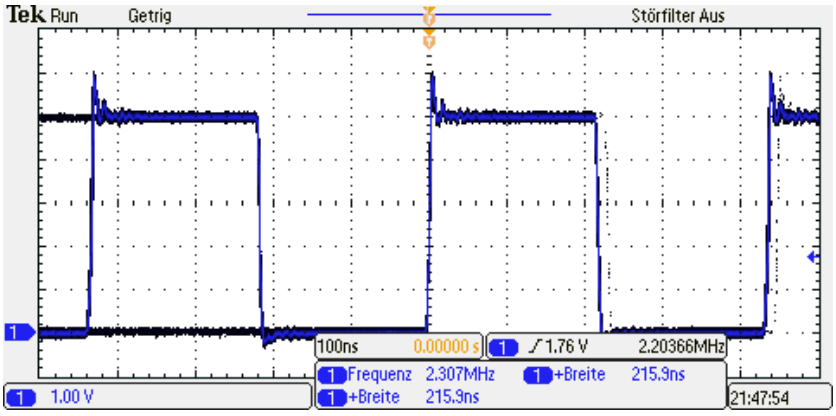

Bild 34: Pintoggling per SFR

Es lässt sich also sagen, dass zwischen den beiden hier vorgestellten Methoden ein Unterschied in der Geschwindigkeit von ca. Faktor 20 liegt. Bei der zweiten Methode ist man in der Lage, ca. alle 216ns einen Pin umzuschalten oder abzutasten. Um einen besseren Vergleich zu ermöglichen, wurde auf einem Atmel AtMega16 mit einem 16MHz-Quarz ebenfalls ein solches Pintoggling implementiert.

```
int main(){
        DDRD=0xff;
        while(1){
                PORTD=0x0;
                PORTD=0xFF;
        }
return 0;
}
```

Listing 19: Pintoggling auf einem AtMega16

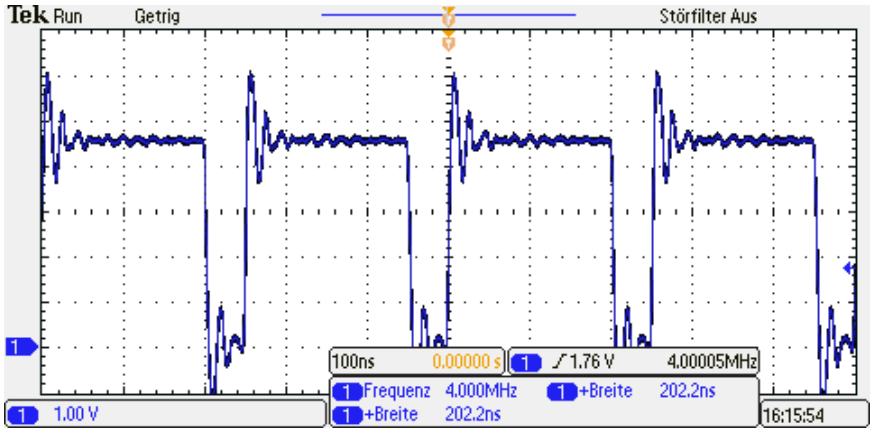

Bild 35: Pintoggling auf einem AtMega16 mit 16MHz

Vergleicht man also die auf dem BeagleBoard-xM erreichte Geschwindigkeit mit kleineren MCU's, wie beispielsweise einem Atmel AtMega16 mit 16MHz, so lässt sich sagen, dass das erreichte Ergebnis plausibel erscheint. Die erreichte Geschwindigkeit ist mehr als ausreichend, um die in Kap. 4 eingeführten Sensoren anzusteuern und auszuwerten. Außerdem greifen hier keinerlei Caching-Effekte mehr. Daher wird diese Methode für die weitere Arbeit verwendet. Allerdings liegt die Vermutung nahe, dass man ohne das Betriebssystem Linux eine deutlich höhere Geschwindigkeit erzielen könnte.

6. Messungen

In diesem Kapitel werden die durchgeführten Messungen aufgezeigt und ausgewertet. Zuvor wird noch auf die maximal erreichbare Auflösung der Messungen und auf die Messfehler eingegangen.

6.1 Auflösung und Genauigkeit der ungestörten Messungen

Dadurch, dass das verwendete Entwicklungsboard digital, also wert- und zeitdiskret, arbeitet, wird ein digitales Raster über die analoge Realität gelegt. Dieses Raster ist bei der hier verwendeten Methode, die GPIO-Pins auf dem Board zu programmieren, 215,9ns breit (siehe Bild 34), sofern die Datain-Register genauso schnell ausgelesen, wie die Set-Dataout- und Clear-Dataout-Register beschrieben werden können. In der Praxis zeigte sich, dass dies nicht der Fall ist, und die Rasterung deutlich gröber ist. Dennoch ist die Rasterung mehr als ausreichend, um die Signale der o.g. Sensoren auszuwerten.

Der Verwendete LV-MaxSonar-EZ1 hat einen eigenen MCU und liefert eine Auflösung der gemessenen Distanz von 1,0 Zoll ($\approx 2{,}54$ cm). Pro Zoll wird am Ausgang ein Puls mit der Länge von 147µs erzeugt. Wird eine größere Distanz gemessen, so wird das Ergebnis entsprechend mit dieser Zeit multipliziert (siehe Kap. 4.4.1). Die Rasterung des Sensormoduls ist also sehr viel gröber, als die Rasterung des Boards, was bedeutet, dass der Fehler, der durch das Raster des Board entstehen kann, theoretisch vernachlässigt werden kann. In der Praxis zeigte sich allerdings ein positiver Effekt: das Sensormodul scheint die gemessenen Werte abzurunden. Dadurch, dass durch das BeagleBoard-xM eine zu lange Zeit abgetastet wird (die gemessenen Zeiten werden in der Software nicht auf ganzzahlige Vielfache von 147µs heruntergebrochen), wird dieses Abrunden teilweise wieder rückgängig gemacht, sodass die aus den Abtastwerten berechneten Distanzen genauer sind, als die per RS232 Schnittstelle vom Sensormodul angelieferten. An dieser Stelle heben sich also zwei Fehler gegenseitig teilweise wieder auf.

Zu beachten ist außerdem, dass das Sensormodul beim Einschalten zwar eine Selbstkalibration durchführt, um das Einschwingverhalten der Sende- und Empfangskapsel bei den Messungen berücksichtigen zu können, allerdings wird die Temperatur mangels Sensor nicht gemessen. Daher sind die Werte, die dieses Sensormodul liefert nicht temperaturkompensiert. Aufgrund der Tatsache, dass das Modul einen Distanzwert liefert und nicht bekannt ist, mit welcher Temperatur intern gerechnet wird, ist eine Temperaturkompensation der Messwerte im Nachhinein nicht mehr möglich.

Gemessen wurde die Distanz Zwischen Sensor und einem Hinderniss mit einem Zollstock und mit dem jeweiligen Sensor. Dabei wurden Vielfache von 10cm verwendet. Aus den Messwerten lässt sich ableiten, dass der relative Messfehler linear kleiner wird, wenn die

Distanz linear größer wird, es besteht also ein invers proportionaler Zusammenhang zwischen dem relativen Messfehler und der Distanz.

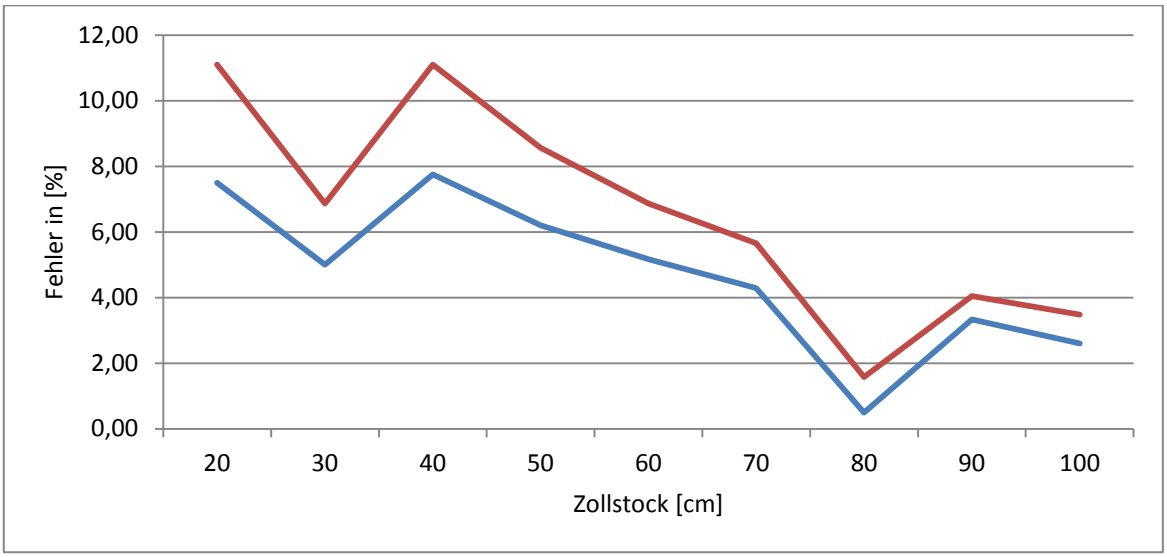

Bild 36: Relativer Messfehler des LV-MaxSonarEZ1 im Normalbetrieb (ungestört), Blau: Fehler der berechneten Werte, Rot: Fehler per RS232

Der Eigenbausensor der HAW Hamburg liefert am Ausgang einen High-Pegel, wenn der Empfänger mit 42kHz angeregt wird. Da dies komplett analog geschieht, existiert auch keine Rasterung des Ergebnisses, wie bei dem anderen verwendeten Sensor. Allerdings wies dieser Sensor einen sehr großen Versatz (Offset) von ca. 4cm auf, was sich in einem sehr starken relativen Messfehler bei kleinen Distanzen äußert. Die Rasterung (Diskretisierung) der Zeit entsteht hier durch das Abtasten des GPIO-Pins (Siehe Kap. 5).

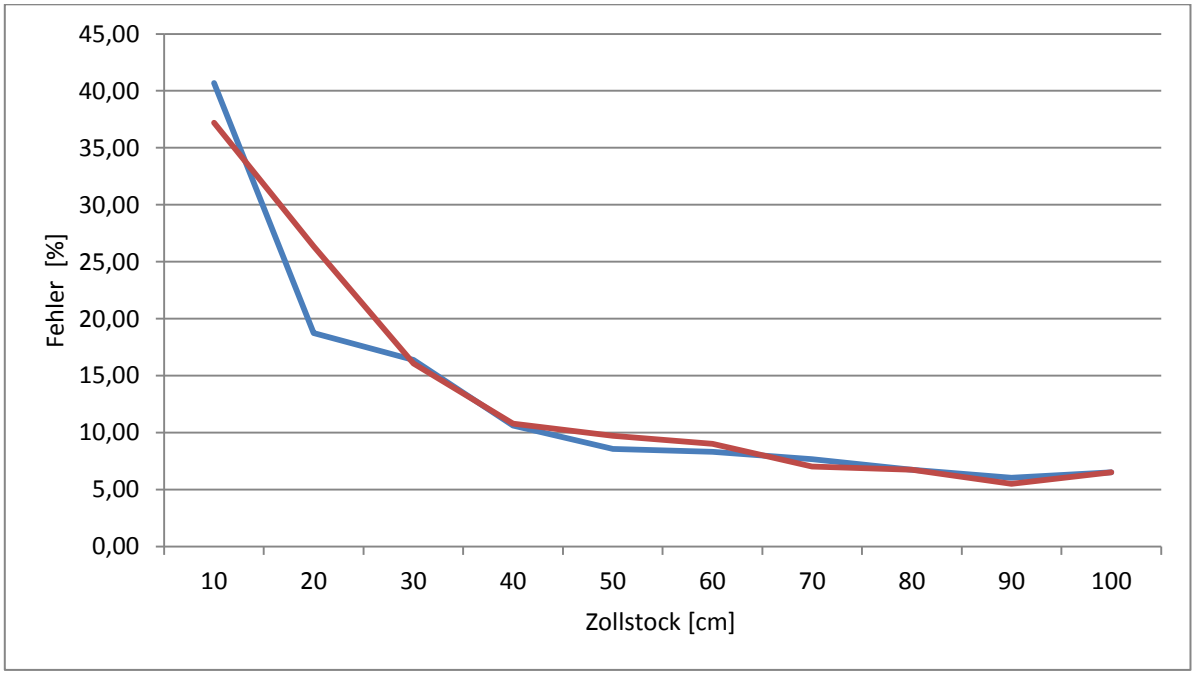

Bild 37: Relativer Messfehler des HAW-Eigenbaus im Normalbetrieb (ungestört). Blau: Per Software gemessene Distanz, Rot: aus der am Oszilloskop abgelesenen Zeit berechnete Distanz

Aufgrund des Versatzes ist der HAW-Eigenbau-Sensor gerade bei kleinen Distanzen absolut ungenau, und kein Vergleich zum Modul LV-MaxSonar-EZ1. Da das fertige Modul beim Starten eine Selbstkalibration durchführt, um solche Versatze erkennen und das Messergebnis entsprechend korrigieren zu können, ist es legitim, auch die Messergebnisse des HAW-Eigenbaus um diesen Versatz zu bereinigen. Dadurch wird ein besserer Vergleich der Messergebnisse ermöglicht. Wird diese Bereinigung durchgeführt, so ergibt sich ein komplett anderes Bild. Trotz des Alters und der Tatsache, dass diese Sensoren von sehr vielen Studierenden in Laborübungen bereits verwendet worden sind, messen sie die Distanz präziser, als das fertige Modul LV-MaxSonar-Ez1.

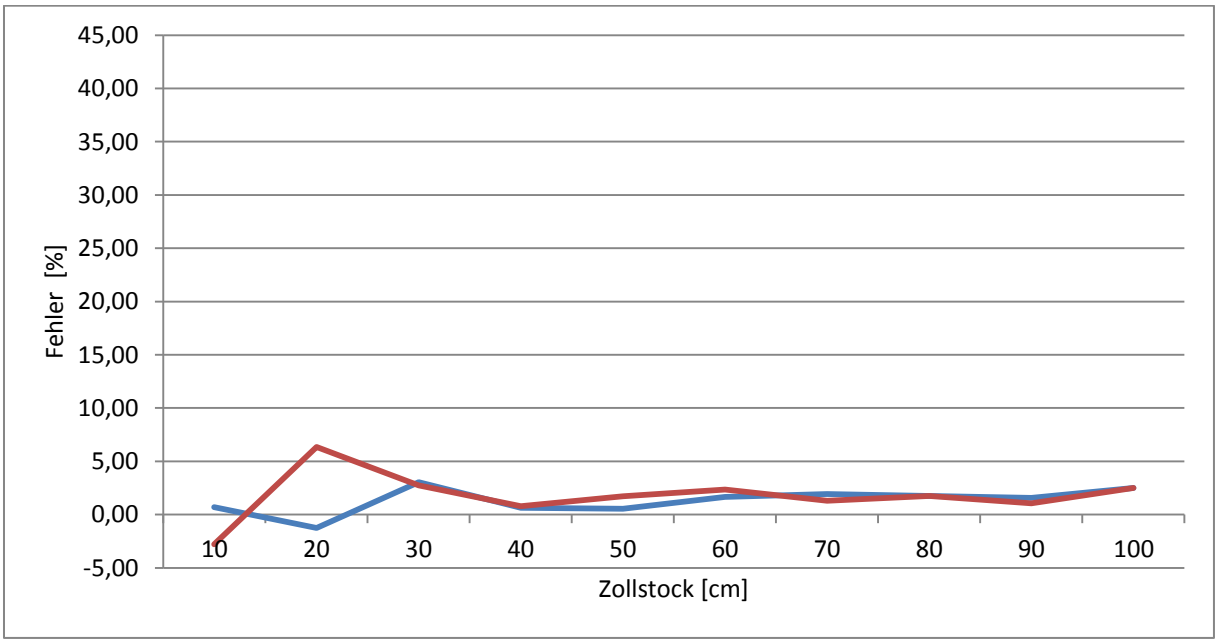

Bild 38: Fehlerbetrachtung mit um den Offset bereinigten Messwerten.

Durch diese Messungen ist außerdem die Funktion der Sensoren und der Messsoftware verifiziert worden. Es kann also davon ausgegangen werden, dass ungestörte Messungen im Rahmen der Möglichkeiten dieser Arbeit exakte Ergebnisse liefern.

6.2 Störung der Messungen

Um die Messungen gezielt zu stören, wurde ein Programm geschrieben, welches in einer kopfgesteuerten Schleife, die nie die Abbruchsbedingung erreicht, zwei Gleitkommazahlen durcheinander dividiert, das Ergebnis allerdings nicht ausgibt (Anhang A1.6). Dieses Programm verrichtet also keine sinnvolle Arbeit, sondern es verbraucht lediglich CPU-Zyklen. Wird dieses Programm gestartet, erfolgt keinerlei Ausgabe, stattdessen wird sofort mit den Berechnungen begonnen, und die CPU zu 100% ausgelastet. Wird nun die Messsoftware gestartet, so erhalten beide Prozesse jeweils ca. 50% der zur Verfügung stehenden CPU-Zeit. Die Idee war nun, das Verhältnis der CPU-Zeiten der Prozesse zueinander durch das Nice-Level, die Priorität der Prozesse, gezielt zu verschieben und

dadurch den Messfehler über die zur Verfügung stehende CPU-Zeit zu messen. Wie sich herausstellte, war es nicht möglich, die verschiedenen CPU-Zeiten durch Prozessprioritäten zu beeinflussen. Ferner scheint es zum Zeitpunkt der Entstehung dieser Arbeit keine Möglichkeit zu geben, unter die dem Betriebssystem Linux einem Prozess zur Verfügung gestellte CPU-Zeit auf einen bestimmten Wert zu begrenzen. Auch der Start eines zweiten Störprozesses brachte keine Änderung der Situation: anstatt, dass jeder Prozess bei gleichem Nice-Level nun 33% der zur Verfügung stehenden CPU-Zeit zugeteilt bekam, bekam der Prozess, welcher die Messungen durchführt, weiterhin 50% der CPU-Zeit, den beiden Störprozessen wurden jeweils 25% zugeteilt.

Die Vertrauenswürdigkeit dieser Angaben aus dem Prozessanzeigeprogramm „top“ wurde mit dem Oszilloskop überprüft. Dafür wurden zunächst alle Prozesse beendet und im Anschluss das Programm, mit welchem die Umschalthäufigkeit der GPIO’s gemessen wurde (Anhang A1.1), erneut gestartet. Die Funktion und das Ergebnis des Programms wurden zunächst auf Korrektheit überprüft. Nun wurde ein Störprozess gestartet und das Verhältnis der Zeiten der Ruhelage und des permanenten Umschaltens auf dem Oszilloskop beobachtet. Auf dem Bildschirm des Oszilloskops war ersichtlich, dass die Zeiten gleich lang waren. Die CPU-Zeit war also, wie vom Programm „top“ angegeben, zu gleichen Teilen über die zwei Prozesse verteilt. Danach wurde ein zweiter Störprozess gestartet. Das Programm, welches die Pins umschaltet, bekam weiterhin 50% der Rechenzeit, die beiden Störprozesse jeweils nur 25% - zusammen also auch 50%. Die Messung mit dem Oszilloskop bestätigte dies, da sich an der Darstellung auf dem Bildschirm nichts änderte. Die von dem Programm „top“ gemachten Angaben sind also vollkommen korrekt.

Dennoch wurden zwei Messreihen mit jeweils sieben verschiedenen Nice-Leveln und 200 Messwerten pro Nice-Level durchgeführt. Dabei zeigte sich, dass die Ergebnisse im Mittel 30% bis 50% zu hoch ausfielen, sich allerdings über die Priorität des messenden Prozesses nicht ändern, was mit Wegnehmen von 50% der CPU-Zeit und dem sich nicht ändernden Verhältnis der CPU-Zeiten zu den Störprozessen zu erklären ist.

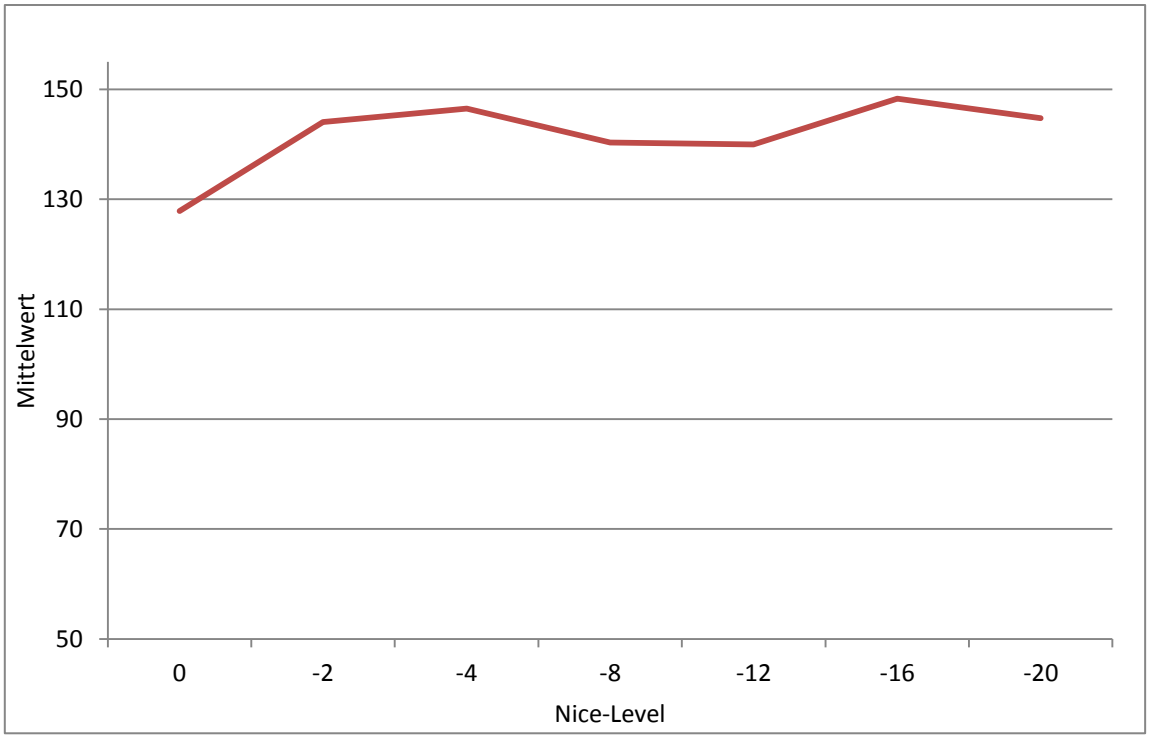

Bild 39: Mittelwert der Messwerte aufgetragen über dem Nice-Level am HAW-Eigenbau, Distanz: 1,0m

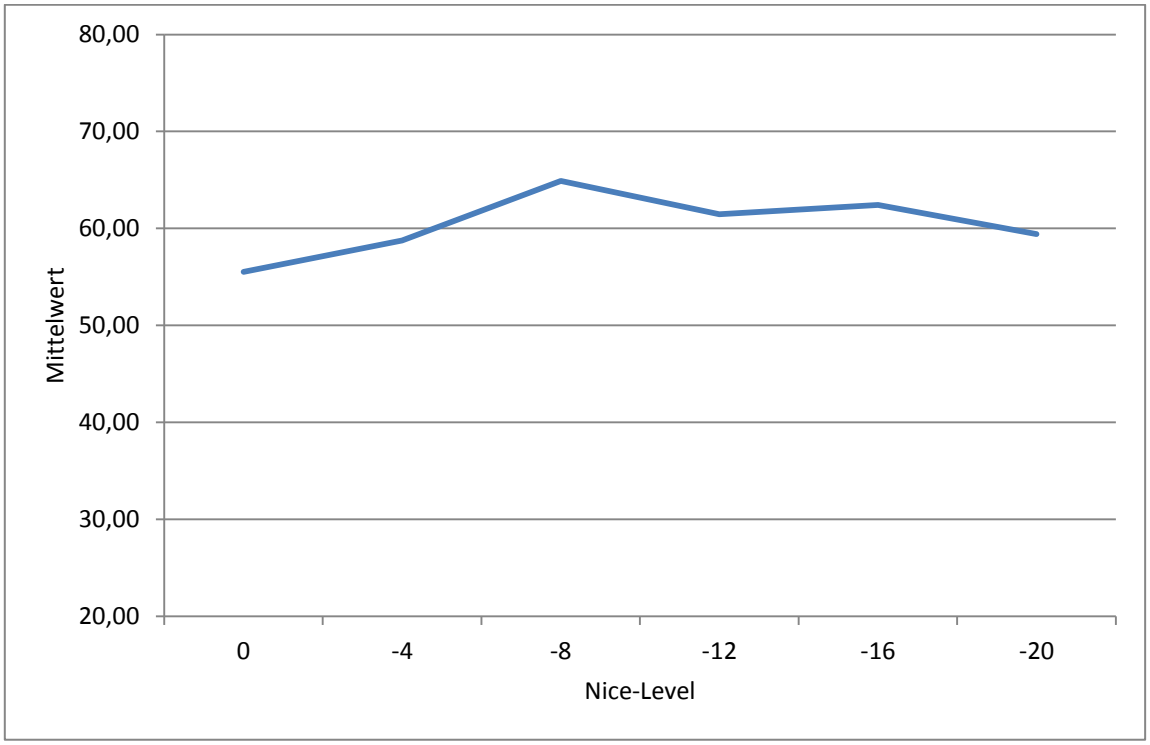

Bild 40: Mittelwert der Messwerte aufgetragen über dem Nice-Level am LV-MaxSonar-EZ1, Distanz: 40cm

6.3 Auswertung und mögliche Bereinigung der falschen Messwerte

Bei den Werten der gestörten Messungen fiel sehr deutlich auf, dass eine hohe Anzahl von Messwerten trotz der Präsenz eines störenden Prozesses weiterhin korrekt war. Diese Werte folgten auch fast immer aufeinander. In dieser Zeit hatte der Prozess also genügend CPU-Zeit. Die Werte, welche den Mittelwert stark veränderten, waren nicht besonders zahlreich, allerdings wichen sie mindestens um den Faktor 3,0 von den vorherigen Messwerten ab. Offensichtlich wurde der messende Prozess während einer Messung von der CPU genommen.

Aufgrund dieser hohen Abweichung sind diese einzelnen, falschen Messwerte (auch für eine Maschine) sehr einfach zu erkennen. Daher gibt es die Möglichkeit, diese bei der Berechnung des Mittelwertes auszulassen oder aber passend zu interpolieren. Wenn eine solche Prüfung der Messergebnisse im laufenden Betrieb stattfindet, darf sich die Distanz zwischen zwei aufeinanderfolgenden Messwerten nicht um den Faktor 3,0 oder mehr ändern. Eine solche Veränderung der Messwerte wurde im Rahmen dieser Arbeit allerdings nicht durchgeführt.

7. Fazit und Ausblick

Im Rahmen dieser Arbeit wurde ein BeagleBoard-xM mit einem berührungsempfindlichen Bildschirm als vollwertiger Linux-Rechner in Betrieb genommen. Außerdem wurden verschiedene Möglichkeiten, Software für diesen Rechner zu entwickeln, evaluiert. Auch wurde die Performance der Kernel-Schnittstelle im Dateisystem für die GPIO‘s mit einer direkten Registermanipulation verglichen. Ferner wurden drei verschiedene Sensoren erstmalig am Board angeschlossen und in Betrieb genommen. Durch Messungen wurden zwei grundsätzlich verschiede, mit Ultraschall arbeitende Distanzsensoren mit einander verglichen und deren Präzision evaluiert. Im Anschluss wurde dann herausgearbeitet, wie stark sich das Messergebnis verändert, wenn dem Prozess, der die Messungen durchführt, die Priorität weggenommen wird. Es zeigte sich, dass die hier durchgeführten Messungen sehr störanfällig sind.

Durch die initiale Inbetriebnahme des Boards, die Evaluation der Möglichkeiten, Software für dieses zu entwickeln und dem Aufzeigen einer Möglichkeit, direkt die Register des MCU’s zu manipulieren, steht der HAW Hamburg nun außerdem eine sehr leistungsfähige Plattform für die Ausbildung der Studierenden im Fach „Betriebssysteme“ zur Verfügung.

In dieser Arbeit fand die Softwareentwicklung ausschließlich unter dem Betriebssystem Windows statt. Interessant wäre sicherlich eine Evaluation der Möglichkeiten, Software für dieses Board unter dem Betriebssystem Linux zu entwickeln. Sollte dies möglich sein, würde man für die Ausbildung im Fach „Betriebssysteme“ keine Windows-Lizenzen mehr benötigen, was einen Kostenvorteil für die HAW Hamburg und damit für die öffentliche Hand darstellt.

Ferner könnte der im Rahmen dieser Arbeit erstellte Programmtext zu Kernelmodulen mit Schnittstelle im Dateisystem umgeschrieben werden, was die Verwendbarkeit der Software weiter steigert. In diesem Zusammenhang könnten die Messungen dann wiederholt werden. Außerdem wurde eine Möglichkeit aufgezeigt, die Messwerte weiter zu bereinigen und damit die Störsicherheit der Messungen zu erhöhen.

8. Quellen

[1] Internetauftritt des BeagleBoard-xM Projektes, http://beagleboard.org/hardware-xM (Stand: 7.1.2012)

[2] http://blogs.msdn.com/b/sfu/archive/2009/03/27/can-i-set-up-user-name-mapping-in-windows-vista.aspx (Stand: 7.1.2012)

[3] James B. D. Joshi, INFSCI 2935: Introduction to Computer Security, Lecture 9, http://www.sis.pitt.edu/~jjoshi/IS2935/ (Stand 8.1.2012)

[4] SMSC Lan9512 Datenblatt, SMSC Corporation, http://www.smsc.com/media/Downloads_Public/Data_Sheets/9512.pdf (Stand: 3.2.2012)

[5] Wikipedia, Artikel S-Video, http://de.wikipedia.org/wiki/S-Video (Stand: 10.2.2012)

[6] Wikipedia, Artikel X Window System, http://de.wikipedia.org/wiki/X_Window_System (Stand: 16.2.2012)

[7] Wikipedia, Artikel X.Org Server, http://de.wikipedia.org/wiki/X.Org-Server (Stand: 16.2.2012)

[8] Prof. Dr. Ing. Reinhard Stolle, ODFM Modulator, Hochschule für angewandte Wissenschaften Augsburg, https://www2.hs-augsburg.de/~stolle/SECURE/100613_NASY_OFDM_Modulator.pdf (Stand: 16.2.2012)

[9] Sergei Griney und Susan Morgan, C/C++ Remote Development – Netbeans IDE 6.9 Tutorial. Juni 2010, Version V6.9-1, http://www.netbeans.org/kb/docs/cnd/remotedev-tutorial.html (Stand: 17.2.2012)

[10] Wikipedia, Artikel Eclipse IDE, http://de.wikipedia.org/wiki/Eclipse_%28IDE%29 (Stand 17.2.2012)

[11] Lilliput Electronics Co., Ltd., Internet: http://www.lilliputweb.net

[12] Horst Kuchling, Taschenbuch der Physik, 18. Auflage, Fachbuchverlag Leipzig, ISBN: 3-446-22883-7

[13] Wikipedia, Artikel Schallreflexion, http://de.wikipedia.org/wiki/Schallreflexion (Stand: 22.2.2012)

[14] Internetauftritt des Eclipse Target Management System Projektes: http://www.eclipse.org/tm/ (Stand: 24.2.2012)

[15] Wikipedia, Artikel Entfernungsmessung, http://de.wikipedia.org/wiki/Entfernungsmessung (Stand: 25.2.2012)

[16] Texas Instruments Incorporated, AM/DM37x Multimedia Device Technical Reference Manual, Silicon Revision 1.x, Version O, Public Version, Doc.ID: SPRUNGN4O

[17] LV-MaxSonar-EZ1 Datenblatt Version 3.0c, MaxBotix Incorporated, http://www.maxbotix.com/documents/MB1010_Datasheet.pdf

[18] Internetauftritt des Putty Projektes, http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[19] HowTo Forge, Artikel „Key-Basierte SSH Logins mit Putty“, http://www.howtoforge.de/anleitung/key-basierte-ssh-logins-mit-putty/ (Stand: 5.3.2012)

[20] TI TXS0104E Datenblatt, Firma Texas Instruments Incorporated, Juni 2006, Doc.Id: SCES651D

[21] Embedded Linux Wiki, Artikel “BeagleBoardDebian”, http://elinux.org/BeagleBoardDebian (Stand: 8.3.2012)

[22] BeagleBoard-xM System Reference Manual, Rev. C.1.0, BeagleBoard.org Community, http://beagleboard.org/static/BBxMSRM_latest.pdf

[23] SHT15 Datenblatt, Sensiron AG, Version 5, Dezember 2011, http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf (Stand: 21.3.2012)

[24] Spakfun Elektronics Inc., SKU SEN-08257, http://www.sparkfun.com/products/8257 (Stand: 21.3.2012)

[25] SHT1X & SHT7X Sample Code, Sensiron AG, Version 2.07, November 2010, http://www.sensirion.com/en/pdf/product_information/Sample_Code_humidity_sensor_SHTxx.pdf (Stand: 21.3.2012)

9. Anhang

A1 Erstelle Programme

A1.1 Umschalten von GPIO 136 und 137 per SFR

```
/*
 ============================================================================
 Name        : BeagleBoard-xM.c
 Author      : Dario Borchers
 Version     : 1.0
 Description : BeagleBoard-xM GPIO Toggling, in German.
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>

int main(void) {

	volatile ulong *padkonf;
	int x;
	int fd=open("/dev/mem",O_RDWR|O_SYNC);
	if(fd<0) {
		printf("Konnte /dev/mem nicht öffnen");
		return 0;
	}

	padkonf=(ulong*)mmap(NULL,0x10000,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x48000000);
	if(padkonf==MAP_FAILED){
		printf("MMAP-Aufruf ging schief - padkonf");
		close(fd);
		return 0;
	}

	//Lotkugeln at4 und at5 in modus 4 schalten -> GPIO's 136 und 137
	padkonf[0x2164/4]=0x011C011C;
	close(fd);

	fd=open("/dev/mem",O_RDWR|O_SYNC);
	volatile ulong *gpio;
	gpio= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
	if(gpio==MAP_FAILED){
			printf("MMAP-Aufruf ging schief - gpio");
			close(fd);
			return 0;
		}

	x=gpio[0x6034/4];
	x=x&0xFFFFF87F;
	gpio[0x6034/4]=x;
	//GPIO's 136 und 137 sind jetzt output

	fd=open("/dev/mem",O_RDWR|O_SYNC);
	volatile ulong *gpios;
	gpios= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
	if(gpios==MAP_FAILED){
			printf("MMAP-Aufruf ging schief - gpio");
			close(fd);
			return 0;
		}
	x=gpios[0x603c/4]; //gpio5-dataout-register
```

```
        x=0|0x00000380;

        //permanentes Umschalten der Pins
        while(1){
                gpios[0x6090/4]=x; //gpio5-cleardataout-register
                gpios[0x6094/4]=x; //gpio5-setdataout-register
        }
        return 0;
}
```

A1.2 Einlesen des Zustandes eines Pins über das Kernel-Interface

```
#include "stdio.h"
#include "unistd.h"

int main(){
    char value;
    int usecs;

    //pin "exportieren"
    FILE *fp=fopen("/sys/class/gpio/export","w");
    fprintf(fp,"137");
    fclose(fp);
    //richtung setzen
    fp=fopen("/sys/class/gpio/gpio137/direction","w");
    fprintf(fp,"in");
    fclose(fp);

    fp=fopen("/sys/class/gpio/gpio137/value","r");

    //ersten wert einlesen
    while(1){
        value=fgetc(fp);
        fseek(fp,0,SEEK_SET);
        printf("%c\r\n",value);
    }

    fclose(fp);
    //pin "deexportieren"
    fp=fopen("/sys/class/gpio/unexport","w");
    fprintf(fp,"137");
    fclose(fp);
    return 0;
}
```

A1.3 Setzen des Zustandes eines Pins über das Kernel-Interface

```
#include "stdio.h"
#include "unistd.h"

int main(){
    //pin "exportieren"
    FILE *fp=fopen("/sys/class/gpio/export","w");
    fprintf(fp,"161");
    fclose(fp);
    //richtung setzen
    fp=fopen("/sys/class/gpio/gpio161/direction","w");
    fprintf(fp,"out");
    fclose(fp);
    //pin toggeln
    fp=fopen("/sys/class/gpio/gpio161/value","w");
    while(1){
        fseek(fp,0,SEEK_SET);
         fprintf(fp,"0");
        fseek(fp,0,SEEK_SET);
         fprintf(fp,"1");
         }
    fclose(fp);
    //pin "deexportieren"
    fp=fopen("/sys/class/gpio/unexport","w");
    fprintf(fp,"161");
```

```
    fclose(fp);
    return 0;
}
```

A1.4 Auswertung der HAW-US Sender und Empfänger, sowie des SHT15

```
/*
 ============================================================================
 Name        : BeagleBoard-xM.c
 Author      : Dario Borchers
 Version     :
 Description : BeagleBoard-xM with HAW-US-Sensors and SHT15 in German.
 Wiring      : Sender: GPIO 136, Empfaenger: 137, SHT15_DATA:138 SHT15_CLK: 139
 ============================================================================
 */

#define SIG SIGRTMIN

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <time.h>
#include <signal.h>

void sht15_sensor_reset(volatile ulong *gpio);
char sht15_read_byte(int ack, volatile ulong *gpio);
void sht15_transmission_start(volatile ulong *gpio);
int sht15_write_byte(char byte, volatile ulong *gpio);
double sht15_calc_temperature(char msb, char lsb, char checksum);
double calc_schall_geschindigkeit(double temperatur);

void sigh(int);

int main(void) {

        volatile ulong *padkonf;
        int i=0;
       int x;
        unsigned char val, c, msb, lsb, checksum;
        double temperatur, schallgeschwindigkeit, distanz,zeit;

        clockid_t clockid=CLOCK_MONOTONIC;
        struct timespec timer1, timer2;
        clock_getres(clockid,&timer1);

        signal(SIGALRM,sigh);
        int fd=open("/dev/mem",O_RDWR|O_SYNC);
       if(fd<0) {
              printf("Konnte /dev/mem nicht ÃƒÂ¶ffnen");
              return 0;
       }

       padkonf=(ulong*)mmap(NULL,0x10000,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x48000000);
       if(padkonf==MAP_FAILED){
              printf("MMAP-Aufruf ging schief - padkonf");
              close(fd);
              return 0;
       }

       //Lotkugeln in modus 4 schalten -> GPIO's 136-139
       padkonf[0x2164/4]=0x011C011C;
        padkonf[0x2168/4]=0x011C011C;
       close(fd);

       fd=open("/dev/mem",O_RDWR|O_SYNC);
       volatile ulong *gpio;
       gpio= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
       if(gpio==MAP_FAILED){
              printf("MMAP-Aufruf ging schief - gpio");
              close(fd);
```

```
                return 0;
        }

         sht15_sensor_reset(gpio);
         sht15_transmission_start(gpio);
         sht15_write_byte(0x3,gpio);
         x=gpio[0x6034/4];
        //x|=0x100;
         //x|=0x200;
         x|=0x400;
         //x|=0x800;
        gpio[0x6034/4]=x;

        //Data jetzt input
         while(gpio[0x6038/4]&0x400); //sensor misst, auf data=0 warten
         //sensor ist fertig, wert abholen.

         msb=sht15_read_byte(1,gpio);
         lsb=sht15_read_byte(1,gpio);
         checksum=sht15_read_byte(0,gpio);
         temperatur=sht15_calc_temperature(msb,lsb,checksum);
         schallgeschwindigkeit=calc_schall_geschindigkeit(temperatur);

         x=gpio[0x6034/4];
         x&=0xFFFFFEFF; //gpio 136 ausgang
         x|=0x200; //gpio 137 ist eingang
         gpio[0x6034/4]=x;
         gpio[0x6090/4]=0x100;

         while(1){
             clock_gettime(clockid,&timer1);
             gpio[0x6094/4]=0x100;
             while(!(gpio[0x6038/4]&0x200));
             clock_gettime(clockid,&timer2);
             gpio[0x6090/4]=0x100;
             while(gpio[0x6038/4]&0x200);
                 zeit=(((timer2.tv_nsec-timer1.tv_nsec)/1000.0)/1000.0); //millisekunden
                 distanz=(schallgeschwindigkeit*zeit)/10.0; //rechnung in cm
                 printf("Temperatur: %2.2f°C Zeit: %1.3fms Distanz:
%2.2fcm\r\n",temperatur,zeit,distanz);
               }

        return 0;
}

void sht15_sensor_reset(volatile ulong *gpio){
    int i,x=0;
    x=gpio[0x6034/4];
    x=x&0xFFFFF3FF;
    gpio[0x6034/4]=x;
    gpio[0x6090/4]=0x400;//sck=0 data=0
    usleep(1);
    gpio[0x6094/4]=0x400;//sck=0 data=1
    usleep(1);
    for (i=0;i<15;i++){
       gpio[0x6094/4]=0x800;//sck=1 data=1
       usleep(1);
       gpio[0x6090/4]=0x800;//sck=0 data=1
       usleep(1);
    }
}

char sht15_read_byte(int ack, volatile ulong *gpio){

    unsigned char i, val;
    int x=0;

    x=gpio[0x6034/4];
    x=x|0x400; //data ist jetzt input
    gpio[0x6034/4]=x;

    for (i=0x80;i>>0;i/=2){
        gpio[0x6094/4]=0x800;//sck=1
        usleep(1);
        if (gpio[0x6038/4]&0x400) {
            val|=i;
        }
        gpio[0x6090/4]=0x800;//sck=0
```

```
        usleep(1);
    }

    x=gpio[0x6034/4];
    x=x&0xFFFFF3FF;
    gpio[0x6034/4]=x;

    if(ack) {
        gpio[0x6090/4]=0x400;
    } else  {
        gpio[0x6094/4]=0x400;
    }
    usleep(1);

    gpio[0x6094/4]=0x800;
    usleep(1);
    gpio[0x6090/4]=0x800;
    usleep(1);
    //gpio[0x6090/4]=0x400;
    return val;
}

void sht15_transmission_start(volatile ulong *gpio){
    int x=0;
    x=0|0x00000400;
    x=x|0x00000800;
    gpio[0x6094/4]=x; //sck=1 data =1
    usleep(1);
    gpio[0x6090/4]=0x400;//sck=0 data=0
    usleep(1);
    gpio[0x6090/4]=0x800;//sck=0 data=1
    usleep(1);
    gpio[0x6094/4]=0x800;//sck=1 data=0
    usleep(1);
    gpio[0x6094/4]=0x400;//sck=1 data=1
    usleep(1);
    gpio[0x6090/4]=0x800;//sck=0 data=1
    usleep(1);
}


int sht15_write_byte(char byte, volatile ulong *gpio){
    int x;
    unsigned char c;
    int result=0;
    for (c=0x80;c>>0;c/=2){
            if (c&byte) gpio[0x6094/4]=0x400;
            else gpio[0x6090/4]=0x400;
            usleep(1);
            gpio[0x6094/4]=0x800;
            usleep(1);
            gpio[0x6090/4]=0x800;
            usleep(1);
    }
    gpio[0x6094/4]=0x400;//sck=0 data=1
    x=gpio[0x6034/4];
    x=x|0x400; //data ist jetzt input
    gpio[0x6034/4]=x;
    usleep(1);
    gpio[0x6094/4]=0x800;
    usleep(1);
    gpio[0x6090/4]=0x800;
    usleep(1);
    result=(gpio[6038/4]&0x400);
    x=gpio[0x6034/4];
    x=x&0xFFFFF3FF;
    gpio[0x6034/4]=x;
    return result;
}

double sht15_calc_temperature(char msb, char lsb, char checksum){
    long temperatur;
    double result;
    temperatur=(long)(256*msb);
    temperatur+=lsb;
    result=(double)temperatur;
    result/=100;
    result-=40.1;
```

```
    //printf("Debug: Temperatur: %2.2f°C\r\n",result);
    return result;
}

double calc_schall_geschindigkeit(double temperatur){
    double result=331.2+(0.6*temperatur);
    //printf("Debug: Schallgeschwindigkeit: %2.2f m/s\r\n",result);
    return result;
}

void sigh(int signo)
{
}
```

A1.5 Auswertung des LV-MaxSonar-EZ1

```
/*
 ============================================================================
 Name        : BeagleBoard-xM.c
 Author      : Dario Borchers
 Version     :
 Description : BeagleBoard-xM with LV-Max-Sonar-EZ1, in German.
 Wiring:     : USB->RS232 @ USB-Port, PW-Out of Sensor to GPIO 136
 ============================================================================
 */

#define SIG SIGRTMIN


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <time.h>
#include <signal.h>
#include <termios.h>

sig_atomic_t ticks;

static void handler(int sig, siginfo_t *si, void *uc);

int main(void) {

        clockid_t clockid=CLOCK_MONOTONIC;
        struct timespec timer1, timer2;
       volatile ulong *padkonf;

       int x,oldx;
       int fd=open("/dev/mem",O_RDWR|O_SYNC);
       if(fd<0) {
              printf("Konnte /dev/mem nicht Ã¶ffnen");
              return 0;
       }

       padkonf=(ulong*)mmap(NULL,0x10000,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x48000000);
       if(padkonf==MAP_FAILED){
              printf("MMAP-Aufruf ging schief - padkonf");
              close(fd);
              return 0;
       }

       //Lotkugeln at4 und at5 in modus 4 schalten -> GPIO's 136 und 137
       padkonf[0x2164/4]=0x011C011C;
       close(fd);

       fd=open("/dev/mem",O_RDWR|O_SYNC);
       volatile ulong *gpio;
       gpio= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
       if(gpio==MAP_FAILED){
                     printf("MMAP-Aufruf ging schief - gpio");
                     close(fd);
```

```
                        return 0;
                }

        x=gpio[0x6034/4];
        //sicher?;
        x=x|0x780;
        gpio[0x6034/4]=x;
        //GPIO's 136 und 137 sind jetzt input

        fd=open("/dev/mem",O_RDWR|O_SYNC);
        volatile ulong *gpios;
        gpios= (ulong*) mmap(NULL, 0x10000, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x49050000);
        if(gpios==MAP_FAILED){
                        printf("MMAP-Aufruf ging schief - gpio");
                        close(fd);
                        return 0;
                }


         //permanentes lesen der Pins
        while(1){
                x=gpios[0x6038/4]&0x100; //gpio5-datain-register, bit 10 -> gpio 136
                if(x && !oldx){ //steigende flanke
                      clock_gettime(clockid,&timer1);
                      //ticks=0;
                }else{
                      if ((!x)&&(oldx)) { //fallende flanke
                          clock_gettime(clockid,&timer2);
                          printf("%3.1f cm\r\n",(((timer2.tv_nsec-
timer1.tv_nsec)/1000.0)/147.0)*2.54);
                          ticks=0;
                      }
                 }
                 oldx=x;
        }
        return 0;
}

static void handler(int sig, siginfo_t *si, void *uc){
    //ticks++;
}
```

A1.6 Prozess zum Stören der Messsoftware

```
#include <stdio.h>

int main(int argc, char**argv) {
    double a = 7.2639;
    double b = 15.29401;
    double c = 0;

    while(1){
        c=b/a;
        //printf("Ergebniss: %f\r\n",c);
    }


    return 0;
}
```

A1.7 Umschalten von Pins auf einem Atmel AtMega16

```
#include <stdlib.h>
#include <avr/io.h>
#include <stdlib.h>


int main(){
        DDRD=0xff;
        while(1){
        PORTD=0x0;
```

```
        PORTD=0xFF;
        }


return 0;
}
```

A1.8 Boot.cmd mit aktiviertem S-Video Ausgang

```
setenv dvimode VIDEO_TIMING
setenv vram 12MB
setenv bootcmd 'fatload mmc 0:1 0x80300000 uImage; fatload mmc 0:1 0x81600000 uInitrd; bootm
0x80300000 0x81600000'
setenv bootargs root=/dev/mmcblk0p5 rootwait ro fixrtc buddy=${buddy} mpurate=${mpurate}
omapfb.mode=tv:pal omapdss.def_disp=tv omapfb.video_mode=700x550MR-24@25
boot
```

A1.9 Boot.cmd für die Verwendung mit einem HDMI-Adapter

```
setenv dvimode VIDEO_TIMING
setenv vram 12MB
setenv bootcmd 'fatload mmc 0:1 0x80300000 uImage; fatload mmc 0:1 0x81600000 uInitrd; bootm
0x80300000 0x81600000'
setenv bootargs root=/dev/mmcblk0p5 rootwait ro fixrtc buddy=${buddy} mpurate=1000
boot
```

A1.10 Init-Skript zur Aktivierung des S-Video-Ausgangs

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          tv-out
# Required-Start:    $local_fs $syslog kdm
# Required-Stop:     $local_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# X-Interactive:     true
# Short-Description: enable/disbale s-video output
### END INIT INFO

set -e

enable() {
    sudo sh -c "echo 1 > /sys/devices/omapdss/display1/enabled"
}

disable() {
    sudo sh -c "echo 0 > /sys/devices/omapdss/display1/enabled"
}

case $1 in
        start)
                echo "Enabling Tv-Output"
                enable
        ;;
        stop)
                echo "Disablig Tv-Output"
                disable
        ;;
        restart)
                echo "Restarting Tv-Output"
                disable
                enable
        ;;
        status)
            cat /sys/devices/omapdss/display1/enabled
```

```
        ;;
        *)
                exit 1
        ;;
esac
```

Abbildungsverzeichnis

Liste der Listings

Gleichungsverzeichnis

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §25(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Ort, Datum Unterschrift