

CHALMERS



Extraction of Streaming Audio Data

Development of a MOST analysing software application

Master of Science Thesis

JAKOB VALINDER

Department of Signals and Systems
Division of Communication Systems and Information Theory
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2011
Report no. EX065/2011

Extraction of Streaming Audio Data
Development of a MOST analysing software application
JAKOB VALINDER

© JAKOB VALINDER, 2011

Master's Thesis no. EX065/2011

Department of Signals and Systems
Division of Communication Systems and Information Theory
Chalmers University of Technology
SE-41296 Göteborg
Sweden

Tel. +46-(0)31 772 1000

Reproservice / Department of Signals and Systems
Göteborg, Sweden, 2011

MASTER'S THESIS no. EX065/2011

Extraction of Streaming Audio Data

Development of a MOST analysing software application

JAKOB VALINDER

Department of Signals and Systems
Division of Communication Systems and Information Theory
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2011

Abstract

Media Oriented System Transport (*MOST*) is a high-speed bus system that is able to transport multimedia data between the different multimedia components within a vehicle. The communication over the network can consist of both control messages, data packets and streaming data simultaneously. Streaming data is transported in the so called synchronous part of the protocol, and could for example be audio sent from audio source devices to the amplifier with its connected speakers and headphones. *MCBuster* is a MOST hardware device from *FYI Communications* that monitors the data sent over the network. It can forward messages, packets and streams over USB to a computer or send messages over a CAN network.

The possibilities and requirements for a streaming MOST data software application are discussed in this thesis. This thesis is also focused around the development of such application to the *MCBuster* hardware. The application must be able to read streaming audio data and present it on a computer, either by speakers or as an audio file. Software for the *MCbuster* to monitor packet data and control messages was already available for use in the project, which then could be focused on creating the audio streaming parts.

To distinguish valid audio channels from others, allocation table is read. There are various ways to do this, but for a passive node, the best option is to monitor the control channel where the allocation table is sent once every 64 frames. Validating audio is also needed to be able to process the audio properly, possibly with the right codecs, such as Dolby digital. The developed software is able to play PCM audio or make a wave file at the computer. Filtering and post-processing is done to the audio samples to obtain audio in the right format.

Preface

This document is a report of a Master of Science thesis work in the programme of Communication Engineering at Chalmers University of Technology, 2011. It was initiated by Björn Bergholm at *Broccoli Engineering* and was executed at their facilities under supervision by the Department of Signals and Systems at Chalmers University of Technology. Examiner is Alexandre Graell i Amat, and advisor at Chalmers is Lotfollah Beygi, both within the Department of Signals and Systems.

Special thanks to all of the above mentioned, all of which made this thesis possible in their own way. Also thanks to anyone else that has contributed to the making of this thesis, including Tomas Forslund at *FYI Communication*, the supporting co-workers at Broccoli, and the opposing David and Matthieu. Final thanks to my wife Anna for proof-reading and support.

Contents

Abstract	i
Preface	ii
Contents	iii
Abbreviations	iv
1 Introduction	1
1.1 Background	1
1.2 Purpose	1
1.3 Objective and Scope	1
2 Method	3
3 Theory	4
3.1 Media Oriented Systems Transport (MOST)	4
3.1.1 MOST Cooperation	4
3.1.2 MCBuster	4
3.2 DirectX	5
3.2.1 DirectSound	5
3.3 Audio formats	6
3.3.1 Dolby Digital	6
4 System description	7
4.1 Overview	7
4.2 MOST hardware	7
4.3 User Interface	8
4.4 Functional software	9
4.4.1 Playback	9
4.4.2 Save file	9
5 Results	10
5.1 Prototype development	10
5.2 Digital processing	10
6 Discussion	11
6.1 Audio formats	11
6.2 Validating data	11
6.2.1 Network functions	11
6.2.2 Data analysis	12
6.3 File sizes	13
6.4 Further development	13
7 Conclusion	14
References	15

Abbreviations

CAN — Controller area network

DD — Dolby Digital

DLL — Dynamic-link library

GiB — Gibibyte

GPL — GNU General Public License

GUI — Graphical User Interface

LFE — Low-frequency effects

LPCM — Linear PCM

MiB — Mebibyte

MOST — Media Oriented Systems Transport

TDMA — Time division multiple access

PCM — Pulse Code Modulation

1 Introduction

1.1 Background

Modern cars have an increasing demand of internal data communication. Vehicle systems are designed to involve interaction between an increasing number of subsystems. Along with the increase of throughput data rate demands, traditional communication protocols, such as CAN (Controller area network), must be exchanged or complemented with protocols that can handle the high load demands of today.

One such system that is designed to enable cross-device communication is MOST (Media Oriented Systems Transport). It is a high-speed bus system, developed especially with the multimedia and infotainment areas in mind.

1.2 Purpose

The purpose of this master thesis is to develop a software program for extracting streaming audio from a MOST bus in a vehicle. Data extracted should be able to be played through a connected computer's speakers or be saved into an audio file on the hard drive.

The streaming data properties should not need to be known in advance, but rather be extracted from the streaming and the control data. The software might use filtering algorithms for detecting valid audio data channels, but must be able to select any of several audio data streams from different sources on the bus.

The solution should be written to use the MCBuster hardware and the software should be tested on an existing MOST set-up.

1.3 Objective and Scope

The objective of this thesis is a working piece of software with suggestions for further developments. The prototype should be able to be used for demonstrations of features and general ideas and give a "proof of concept" to streaming audio reception. Future hardware development projects should be able to use the program as a base.

This thesis is trying to figure out what is needed in terms of software communication in order to extract streaming MOST data onto a computer. Other issues it tries to address are the amount of audio format properties that needs to be known in advance, and how valid audio source data channels can be distinguished from non-valid channels.

The scope of this thesis includes software design, implementation and testing. It is not intended to include hardware design or implementation. The algorithm was

not intended to be fixed to one audio setting, but consider different audio formats, for example multi-channel PCM or AC3.

2 Method

This Master of Science thesis work consisted of three phases: research, implementation and test.

The research phase was about performing a pre-study to get some basic knowledge about the system properties. Reading of the official MOST specifications was of vast importance, as well as reading other sources that deals with the subjects of MOST or audio streaming. Also investigation of the MCBuster hardware functionality was a requirement for the implementation phase.

The main focus of the thesis project was the actual software development in the implementation phase, which therefore took most of the project's time. It was executed iteratively in the form of prototyping methodology. Prototyping is the concept of construction of prototypes; early working versions of the final result (Lichter, 1994). It is used to show some of the wanted features of the final product, which can be tested, evaluated and developed early in the process. The way Lichter describes the various kinds of this method, this development project could be said to be a *presentation prototype* as the final version of this project does not necessarily lead to a finalised product.

In the process, existing code snippets taken from other sources were regarded as useful reference material, without actually contribute to the software in the project. Such assistance were mainly accessed within the scope of code with free or some kind of open license, such as GNU General Public License (GPL) or similar (Free Software Foundation, Inc., 2007).

In the testing phase, use case tests for the software prototypes were performed in the existing test set-up to confirm certain functionality, such as connectivity and the playing of PCM audio. Test failure did occasionally send back the prototype to the implementation phase, as part of the iterative work flow.

3 Theory

This chapter gives a basic introduction to MOST, the protocol over which the audio data is obtained. There are also some sections about DirectX from Microsoft, which is the software platform used for playing audio on the computer. Finally, some different audio formats used for audio data are presented. See Section 4.1 for a system overview.

3.1 Media Oriented Systems Transport (MOST)

MOST is a high-speed, synchronous vehicle transport protocol, developed mainly with multimedia demands in mind (MOST Cooperation, 2010). It is usually implemented as a fibre optics based ring topology network, and has built-in plug-and-play functionality. Data is multiplexed using TDMA (Time division multiple access), which allows MOST communication to have dedicated channels for streaming data, packet data and control data.

The MOST technology currently exists in several versions: MOST25, MOST50 and MOST150. They mainly differ in their data transfer rates, but also in their configuration and functionality. MOST25 is preferred by the European and Korean markets, while US and Japanese markets use the copper-wire based MOST50. The most recent version, MOST150, is not so frequently used due to its novelty, but is favourable because of the 150 Mbps data rate and the built-in Ethernet support.

3.1.1 MOST Cooperation

The MOST standard is developed by the MOST Cooperation; an assembly of car-makers, suppliers and other companies in the business (MOST Cooperation, 2010). It was established in Germany in 1998. The resulting MOST protocol has become a de-facto standard for in-vehicle infotainment and is used in over a hundred car models.

3.1.2 MCBuster

The company *FYI Communication* has developed a product named *MCBuster*, which is a MOST25 analysis tool (See Figure 1). The hardware is passing MOST data on to a host using USB, RS232 or CAN, and can be configured to filter out the data that you are interested in. Software for the MCBuster to monitor packet data and control messages was already from the beginning available for use in the project, which allowed the focus of the project to be the creation of the audio streaming parts.



Figure 1: The MCBuster hardware used for MOST analysis. FYI communication www.fyi.se

3.2 DirectX

DirectX was chosen as the interface for the playback feature of the developed application. It was chosen since it had good code reference material available from Microsoft, as well as the vast range of possibilities it exhibits.

DirectX is a programmer interface included in windows, initially developed for game development. It consists of a series of developing components for different applications such as 3D rendering (*Direct3D*), input device handling (*DirectInput*) or font display (*DirectWrite*). It is intended for Microsoft Windows (included as standard since Windows 95 OEM SR2), but is also used as the foundation for Xbox.

3.2.1 DirectSound

The playback and recording device for wave format audio data in DirectX is called *DirectSound*. It has a tight connection to the sound card, which at the time of its introduction in 1995 was a quite revolutionary feature. DirectSound is used to create sound buffers, in which audio samples are piped to an output sink. There exists an addition for 3D sound called *DirectSound3D*, and since DirectX 8, both of them have been merged into *DirectX Audio*. DirectSound was developed mainly with games in

mind, and any sound could therefore be filtered, altered or otherwise processed to create special effects, such as 3D-positioned sounds or Doppler-effects.

3.3 Audio formats

Audio data has several ways to be expressed technically. For sending in an audio network, such as the MOST network, the audio data samples are most conveniently presented as raw, uncompressed data samples.

One such encoding method is Linear Pulse Code Modulation (LPCM), which is a linearly quantized representation of the amplitude of the audio waveform at evenly spaced discrete time instances. The method is the most commonly used method when the wave file format is used to save data as an audio file on a computer, and could be used both for mono, stereo or multichannel audio data. One advantage of the simplicity of the LPCM format is the possibility to easily get the output as an analogue audio signal, just by sending the data through a simple linear D/A converter. Even though other quantization methods exist, PCM is often implied to use the linear one, and hence have become somewhat synonymous with LPCM. The notation of PCM will be used in this document, implying linear PCM.

More difficult to handle, but also very common, are digital formats such as Dolby Digital or DTS. Here all audio channels are blended together and sent as an encoded data stream, to be decoded near the speakers. These formats are most used when sending multichannel data, such as 5.1 surround sound, but has also the ability to be used for stereo or mono sound.

There also exist many other different encoding formats, including *mp3* and various manufacturer-specific formats.

3.3.1 Dolby Digital

The Dolby Digital format is a digital encoding technology from Dolby Laboratories which follows the ATSC A/52 standard (Dolby Laboratories, 2010). The format is used as one of the standards for audio on DVD-Video discs as it encapsulates up to 5 discrete audio channels and one LFE audio channel into a digital bit stream.

The A/52 standard is also known as AC-3. It sends data in blocks, each representing 256 samples of data per channel, compressed and administered. There also exists a file format with the file extension *.ac3* which is the bit stream saved in a file. The bit stream is identified as an AC-3 digital audio stream as it, like many other data streams, has an identifier "sync word" in the beginning of each block, in this case with the value of 0x0B77. (ATSC, 2010).

4 System description

This section describes the system that has been the result of this project, including the hardware set-up used and the software components that have been developed.

4.1 Overview

The setting is based on an MCBuster hardware unit. It is connected to a MOST network, as well as to a USB interface of a standard PC. The USB interface has multiple uses; configuration of the MCBuster hardware, sending instructions e.g. start or stop, as well as receiving control messages and audio data. On the software side, an existing library with controlling functions for the hardware was used to communicate with the MCBuster and receive data. What was developed was the functions to collect audio data from the existing library, process and present that audio, and also an example of user interface. For a graphical overview, see Figure 2 which shows the connections between the developed software and the existing system.

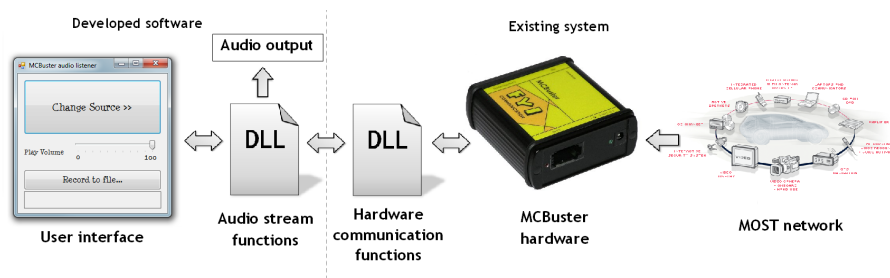


Figure 2: The system overview flow chart.

The MOST network in the test environment consists among other things of a display unit, an amplifier including speakers, DVD player/FM radio media unit and a controlling unit. The normal data flow over the network is that audio data is sent over the network's synchronous channels. When an audio source has been initiated to play, audio data is from the source to the amplifier with its attached headphones or speakers. These audio information are intercepted by the MCBuster and sent to the PC for processing.

4.2 MOST hardware

The actual audio data in the MOST network is intercepted by the MCBuster (See Section 3.1.2). The hardware-near functions initialises the MCBuster, opens a connection and selects the data channels that are in use for audio transfer. The received audio data

bits are sent over USB to the computer for playing and, if needed, saving as a sound file.

4.3 User Interface

For the purpose of using user input for controlling the system, a simple graphical user interface (GUI) application was created; see Figure 3. The functionality lies however not in the GUI *per se*, but in stand-alone software, which makes it possible to use different interfaces according to taste or level of functionality needed. This particular implementation is based on a Microsoft .NET 4 Windows form but other solutions, including hardware interfaces, are possible to use with the controlling software.

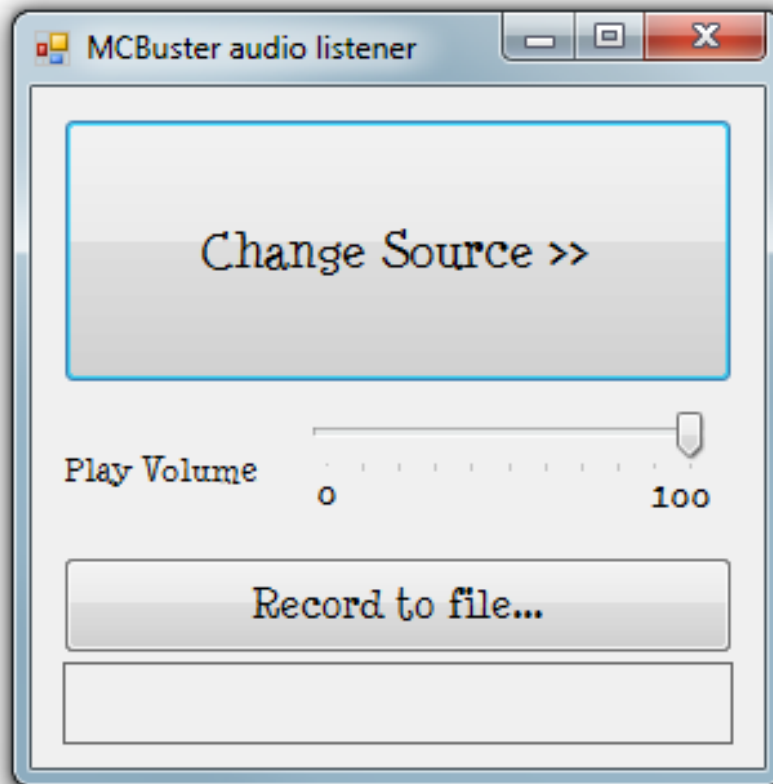


Figure 3: The simple graphical user interface (GUI) for controlling the connection.

4.4 Functional software

The software for controlling the connection with the hardware is presented as a dynamic link library (DLL). Functions exist for starting and initialising the device, and of course for changing channels. The software also presents an audio playback device as well as a mechanism to save a sound file on the computer.

4.4.1 Playback

The software that is used for playing the sound on the windows computer is based on the DirectSound part of the DirectX interface, see Section 3.2.1.

4.4.2 Save file

As a complement to playing the audio data through the computer speakers, the user also has the option to save the data as an audio file on the computer. The save file option asks for a file name through a save file dialogue, and the received data will be written into a file using a file stream writer. The file format for saving is *wav*, which basically contains the raw data samples, preferably in stereo. These files can, if needed, be converted into other file formats, such as *mp3*, by external third-party software.

5 Results

5.1 Prototype development

Audio data from the MOST network has in test settings successfully been transferred to a computer via the system. The sources have been both stereo PCM from the FM receiver, as well as Dolby Digital 5.1 audio from a DVD player. As no AC-3 decoder has been included in the software design, no tests could be performed on the Dolby Digital compressed data. This was somewhat compensated for by the transport protocol of the DVD audio, as it transferred 2 channels stereo PCM along with the compressed surround sound. The PCM audio was successfully played in the speakers of the computer, and at the press of a button also saved to the hard drive as a wave audio file.

5.2 Digital processing

The received audio samples is not in a format which can be presented right away without having any processing applied. First of all, the byte order is big-endian, which means that the bits with the highest significance (greatest impact on the total value) are sent first. The problem is that this is opposite to the little-endian format that the wave audio files and the play algorithm use, so every low-significance byte and high-significance byte had to exchange place. Second, there was a problem of extreme-valued noise that sometimes was added to the sound data. Raw data analysis showed eventually that it was caused by some semi-regular bit-shift of audio data in the hardware. As the bit-shift does not lie in the software, it could not be fixed. The only thing is to some extent compensate for it by a detection algorithm. This works partly due to the regularity, but it also uses the fact that waveform audio at high sample rates is continuous whilst bit-shifted samples might have sudden differences in sign and amplitude if the sign bit happens to change.

6 Discussion

6.1 Audio formats

The algorithm has a lot to live up to if it should be completely generic. Audio formats such as Dolby ProLogic II, DolbyDigital, 2 channels stereo PCM or manufacturer-specific proprietary formats puts special needs to the software. Other issues like Intel or Motorola byte order, or little or big endianness also has to be addressed. What the algorithm can detect in its current state is the presence of bit-shifted samples, while the rest of the bytes from allocated channels is assumed to contain valid PCM data samples. As digital bit stream audio usually has unique identifiers, the inclusion of other codecs along with stream identifications could be done relatively easy. This is even more so if the syntax of each format is parsed for consistency, but such functionality is not included in this version.

In case of a wrongly configured system, channels that are read might be channels that are unused by the system, or used in another way than you think. An example of this was when audio data samples were 16 bits words and the channel configuration offset happened to be one byte wrong. In such cases, the low byte is mistaken as a high byte and the resulting sound was nothing but white noise. It is fairly possible to introduce a check for white noise channels and thus detect such alignment errors, of course with an exception if the audio data itself resembles white noise.

6.2 Validating data

The received data is supposed to be audio data, but can that be verified? To know whether the data received really is playable audio or not, you must validate the data. There are two ways of finding which of the MOST channels that are in use for audio data; either active queries to the network or data analysis.

6.2.1 Network functions

The first method uses the fact that the procedures of allocating and maintaining synchronous channels in MOST are utilising control data messages. In case you have an active MOST node, the approach is easy and straightforward. You can yourself send control messages and receive information about which node is sending what, to whom and on which channels. For this purpose every network has a connection master that stores allocation information that every node in the network at any time can obtain by sending a request message. This is however not possible with passive Most spy nodes such as the MCBuster, which has to rely only on extracting data from the network.

Control messages could however be useful even if your node is passive and thus

cannot send messages of its own. By just listening to the traffic over the network, the passive node could intercept information about the streaming data channels. All allocation initiations could be recorded, as well as any other node's information requests and corresponding answers. This approach depends however on full data perception and luck with control requests.

The last, but not least important, MOST25 network feature is the regular transmission of the allocation table. Within the designated bytes for control messages, 2 out of 64 are reserved for the allocation table, which means that the network sees an updated allocation table nearly a hundred times per second. This method does however not tell you anything about the source or the contents of each channel, just the usage status of each channel and the affinity amongst them.

6.2.2 Data analysis

To really know if a bit stream is valid, you must analyse it somehow. Some encoding formats have special markers as indicators of their respective formats, which must be found in order to engage the appropriate decoder. If no such markers is to be found, it could be an idea to assume that the data at hand is uncompressed audio data. Such a case does not give the audio data properties automatically, but there are certain things that you nevertheless know about the data, or at least can find out.

One indication is the amount of channels used for the connection. A group of 4 bytes is probably not a 32 bit mono PCM, but rather 16 bit stereo or at least dual mono data stream. If more than 4 channels are allocated, that probably means that the data is digital audio, for instance 5.1 surround sound from DVD player. The test set-up I have been using, transmits PCM alongside the digital stream. If the sound is in stereo, left first - then right is the convention.

The PCM values themselves could also yield information about the configuration. As previously mentioned, the least significant bits are statistically random for audio data, and could be used as an indicator of channel allocation. One can also use the fact that audio typically is continuous if sampled fast enough, or that stereo audio channel typically are highly correlated from time to time.

Although the synchronous part MOST is capable of streaming all kinds of data, it is mainly used for audio purposes. The data transfer rate per streaming channel is approx. 0,35 Mbps, and the number of used channels for an application should be limited to ensure network reliability. This means that communication with higher rate demands, such as DVD video, must be achieved using external communication links, while the MOST25 protocol still is used.

6.3 File sizes

When saving data to a file one might consider to use a limitation on file sizes to prevent memory flooding. Raw PCM audio from a CD player over MOST yields $2 * 2 * 48000$ bytes per second, or about 11 MiB per minute. This can be compared to the 2 GiB limit to wav files, which corresponds to approximately 3 hours. If there is a need to record further, one possible solution is to record multiple sequential files, possibly with a limit to the number of files.

6.4 Further development

Not included in this realisation are decoders for other audio formats, such as mp3, DTS or the mentioned Dolby Digital. The digital bit-stream is able to be received, but not yet identified correctly or further processed. To include such decoders would be a grand improvement to the generality of the system. Bit stream identification would be needed, either as general analysing or matching of pre-defined audio characteristics and transmission patterns. A related improvement is the option to save the file in other formats than the wave file format.

Hardware development was not part of this thesis, since this is a software development project. The methods developed here, however, could be used as the software base in hardware projects if needed. The communication and filtering developed for the available MCBuster hardware is the same that is required in a future hardware project, and could hence be reused.

The MCBuster hardware in its current configuration has been introducing artefact errors in the bit stream, and has thus not been optimal as hardware. This could possibly be improved by reconfiguring the system with streaming correctness in mind. Another thing to improve is to change the transmission protocol for synchronous data, for example with a sync header. This will be useful in distinguishing streaming data from other data, as well as getting clearer information about channel positions over the bus for each byte.

7 Conclusion

The software application is prototyped along with a user interface to present the functionality and results. Audio data is in the developed application successfully transferred from the MOST network to the computer. The software is showing results by being able to play the received samples through the speakers, as well as producing wave audio files. The filtering mechanisms are producing audio samples that are playable by analysing samples and reduce deterministic artefacts introduced by hardware.

Using the MCBuster in its current configuration has been seen to introduce artefacts in the stream, which makes the system suboptimal. Later configurations might solve this and use the capabilities of the hardware better.

The validity of MOST audio channels is monitored by the network, and is available at the control channel. To monitor this allocation in a passive MOST node, the allocation table that is periodically sent has to be read, as stated before. In case of an active node, queries could be sent to obtain the information.

It was stated that there is relatively little effort needed to identify various audio format streams as well as some of their properties, for example identify data streams by the unique keyword identifier in the header. Also PCM data with unknown properties could in most cases be characterised by signal analysis.

References

- ATSC. 2010. *Digital Audio Compression Standard (AC-3, E-AC-3)*. Advanced Television Systems Committee (ATSC). Retrieved June 20, 2011.
URL: <http://atsc.org/cms/index.php/standards/published-standards/48-atsc-a52-standard>
- Dolby Laboratories. 2010. *Frequently Asked Questions about Dolby Digital*. Dolby Laboratories. Retrieved June 20, 2011.
URL: http://www.dolby.com/uploadedFiles/Assets/US/Doc/Professional/42_DDFAQ.pdf
- Free Software Foundation, Inc. 2007. "GNU gpl." Retrieved March 30, 2011.
URL: <http://www.gnu.org/licenses/gpl.html>
- Lichter, H. et al. 1994. "Prototyping in industrial software projects-bridging the gap between theory and practice." *Software Engineering, IEEE Transactions on* 20(11):825–832.
- MOST Cooperation. 2010. "Vehicle Count Hits 100 MOST Car Models." *MOST Informative* 6:6–7.