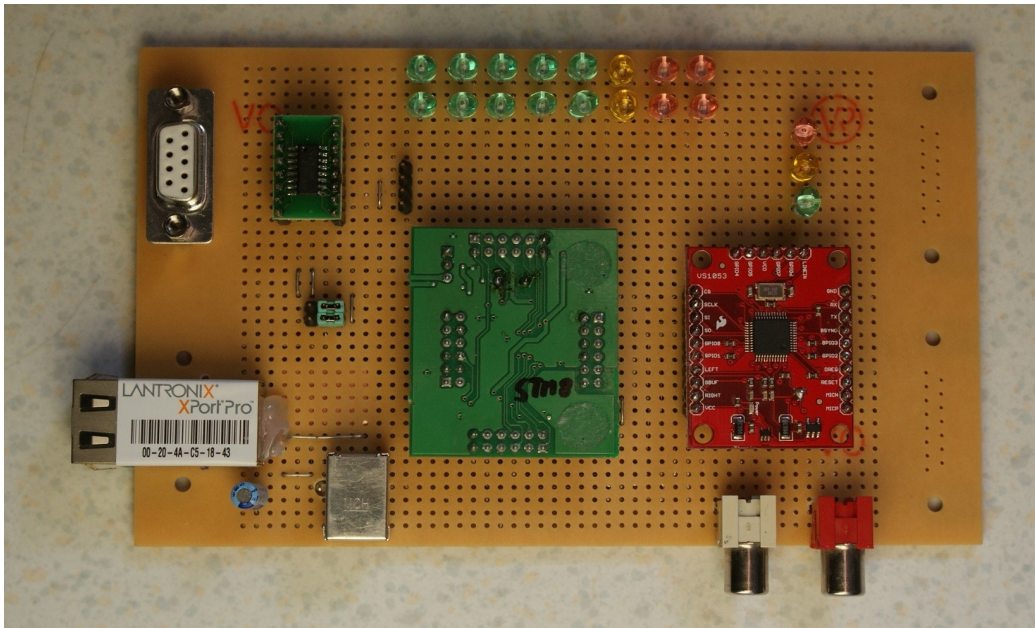


OTELO's



OggStreamer

an Audiostreamingdevices based on the
Lantronix XportPro, VLSI VS1053
and STMicroelectronics STM8S



OTELO's OggStreamer by Georg Ottinger is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/).

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Table of Contents

Functional Description.....	4
Technical Description.....	5
Signal Chain.....	5
VLSI VS1053 Ogg-Encoder.....	6
ST STM8S105C6T6 Micro-Controller.....	6
Lantronix XportPro.....	6
Technical details.....	7
High-speed RS-232.....	7
XportPro to STM8 Protocol.....	7
Getting Started.....	8
Use Scenarios.....	9
Costs of Production.....	9
Futher Optimizations.....	9
Eliminating the intermediate mikro-controller.....	9
Replacing VS1053 by VS8053.....	10
Implementation of an IceCast Server.....	10
Storing the VS1053 Firmware on the XportPro.....	10
Appendix.....	11
Bill of Material.....	11
Schematics.....	12
PCB Print.....	13
Used Software.....	13

Functional Description

The diagram below describes the function of the OggStreamer within a Broadcasting-Setting:

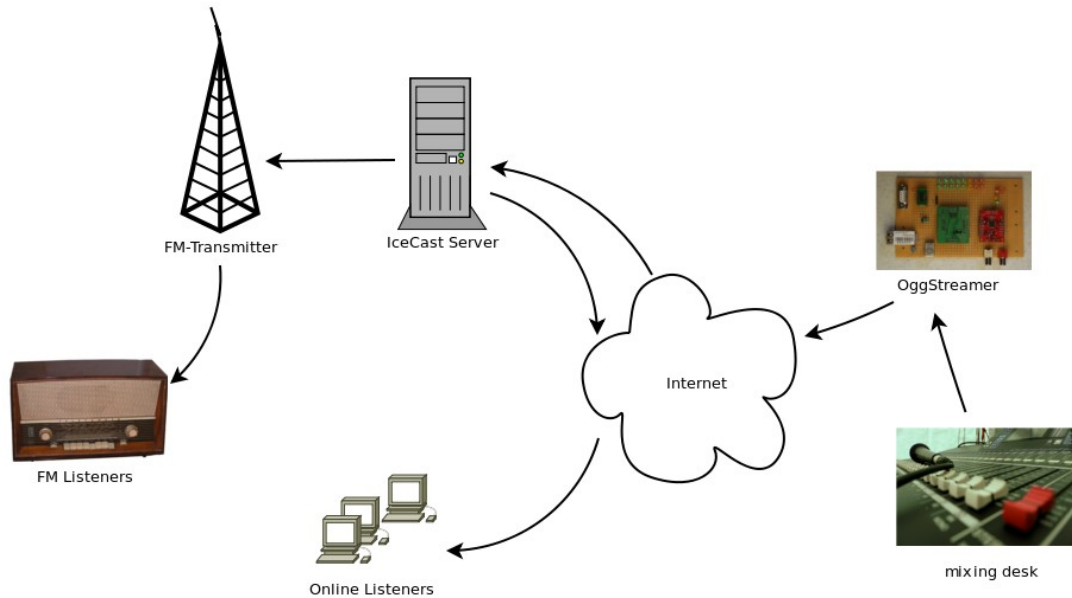


Illustration 1: Functional Description

The audio-signal coming from a mixing desk is fed into the OggStreamer-circuit, which encodes the analog audio-signal into the digital compressed Ogg/Vorbis Format. The Lantronix XportPro encapsulates the compressed audio-data within the Ice Cast-audiostreaming protocol and transmits the audio-data over the Internet to an Ice Cast-server. At the main-studio an Ice Cast-client connects to the Ice Cast-server and puts the audio-signal on Air, which then can be received with conventional FM-radios. Additionally on-line-listeners can connect directly to the Ice Cast-server.

Technical Description

The prototype of the OggStreamer is powered by a standard USB type B plug, which provides the circuit board with +5 Volt DC. On the bottom side of the board a LD1117-33 linear regulator generates +3.3 Volt DC for the components of the board.

Two cinch connectors provide a standard stereo Line-In Input for the analog audio-signal. The Status-LEDs indicate an ongoing recording and possible connection-errors. The VU-meter helps the operator to adjust the volume of the audio-signal, to achieve the maximum dynamic range and to avoid clipping.

The XportPro acts as an gateway to the outer world, namely the Internet, and connects with its Ethernet Port to the Network.

For debugging purpose the prototype also includes a RS-232 DSUB-9 connector.

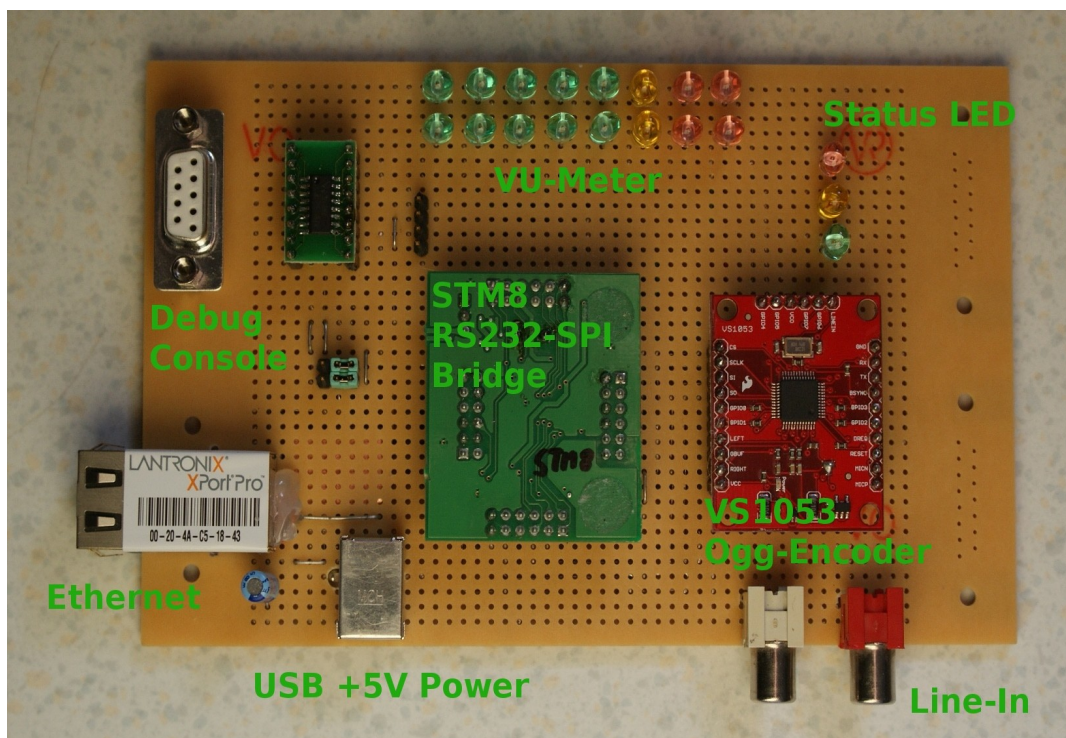


Illustration 2: Components Overview

Signal Chain

The audio-signal enters the board as analog voltage and exits it as compressed audio-data on the Ethernet. In between the signal runs through OggStreamer's three core components:

1. VLSI VS1053 Ogg-Encoder
2. ST STM8S105C6T6 Micro-Controller
3. last but not least the the Lantronix XportPro

VLSI VS1053 Ogg-Encoder

The VS1053¹ chip is a specialized Audio-DSP with integrated Audio-ADCs and Audio-DACs. It is mainly used for PCM, MP3, OGG/Vorbis, MIDI and FLAC playback, but can also be used as an OGG/Vorbis Encoder. It is quite impressive, to me, that a small chip with about 16 kByte of Instruction-RAM and 16 kByte of Data-RAM, is capable of encoding OGG/Vorbis at around 53 Mhz. This chip also provides 8 GPIOs and communication is done via a SPI interface.

The signal enters the VS1053 as analog-audio and exists it via the SPI interface as a already complete Ogg/Vorbis-File. In order for the OggEncoder to work, it is necessary to load a special firmware plug-in² into the VS1053, which is provided from VLSI free of charge.

ST STM8S105C6T6 Micro-Controller

The STM8S105C6T6³ is a 8bit Micro-Controller with 2kByte RAM and 32kByte Flash. Besides numerous other features it also includes one UART- and one SPI-peripherel. This uC acts as bridge between the UART of the XportPro and the SPI interface of the VS1053. The uC first loads the ogg-encoder firmware plug-in into the VS1053 and then streams the encoded audio-data to the XPortPro at a Baudrate of 230400bps. The internal flash of the uC is big enough to also include the VS1053 ogg-encoder firmware, which is about 27kByte in size. Due to this fact the the protocol on the UART-side can be kept very simple. The STM8 also automatically updates the VU-Meter.

Lantronix XportPro

The Lantronix XportPro⁴ with its dma-enabled serial port is the perfect fit for high-speed RS-232 datastreaming. It was very simple to implement in this design. On the hardware-side because of its easy-to-handle 8-pin package. And on the software-side due to the straight forward Linux SDK.

In our application the XportPro connects to an IceCast Server and up-streams the audio-data received from the serial port. As the XportPro is defacto a complete tiny Linux-Computer, it is straight forward to port existing libraries and applications to the XportPro. The current stage of the prototyp is only Proof-of-Concepts and stores the received audio-data as a file on a nfs-server. The next step in the development process will be the capability of direct streaming the audio-data to an IceCast server. Therefore the libraries libvorbis, libogg and libshout will be ported to the XportPro.

1 <http://www.vlsi.fi/en/products/vs1053.html>

2 <http://www.vlsi.fi/en/support/software/vs10xxapplications.html>

3 <http://www.st.com/mcu/devicedocs-STM8S105C6-113.html>

4 <http://www.lantronix.com/device-networking/embedded-device-servers/xport-pro.html>

Technical details

High-speed RS-232

The compressed audio-data streams at an average of 130 kbps. In order to transfer this amount of data a rs-232 connection with 230400 baud is needed. While developing the OggStreamer it proofed necessary to enable the dma-transfer modus of the serial port. But even after enabling dma, the data was not reliable transferred from the STM8 to the XportPro. Taking a look into the mcf_serial.c and mcf_dma.c drivers of the Linux Kernel, it showed that the actual baudrate of the XportPro at nominal 230400 baud is 236742 baud (= $83333333/32/11$; 11 is the used divisor), which represents an error of +2,75%. From a theoretical standpoint an error of +2,75% is acceptable (everything below 10% should be ok). As the STM8 in our implementation uses an 14,7456 Mhz oscillator, the STM8 could be adjusted very accurately to 230400 baud. While transmitting single bytes no problem was observed, but when streaming data-bursts many receiving errors occurred. As a workaround the Baud-rate of the STM8 was set to 237832 baud, which represents an relative error of around +0,46%. In this case the transmission of data-bursts proved to be stable.

In some cases it is desirable that the UART of the MCF5208 inside the XportPro should run as close as possible to nominal rs-232 values (like 115200, 230400, 921600). One way to achive this could be the use of an 14,7456 Mhz crystal and/or changing the PLL-settings of the MCF5208 to a core frequency as close as possible to 147,456 Mhz. This would lead to a slight drop in performance, but should be acceptable in most of the cases.

XportPro to STM8 Protocol

The protocol between the XportPro and the STM8 is kept very simple. It basically consists of one byte commands sent from the XportPro to the STM8 and a streamed OggFile from the STM8 to the XportPro. The current protocol doesn't include error checking, therefore care for an reliable rs-232 connection have to been taken.

To enable the communication with the STM8 the XportPro first has to send the Magic-string "OgGi" consisting of 4 bytes (without termination). This feature has been implemented to prevent the STM8 from being confused by output from the dBug-bootloader or the Linux-kernel.

Currently the command-set of the STM8 includes the following commands:

Command	Description
'b'	Start Streaming
'e'	End Streaming
'0' – '7'	Set Status-LEDs (Bitmask 0-7)

Every time before the STM8 starts streaming the VS1053 firmware-plug-in is transferred, which delays the start for a few 100ms.

By sending "End Streaming" command the state of the communication protocol is reseted, therefore the Magic-string has to be resend in order to communicate with the STM8 again.

Getting Started

The present prototyp represents a proof-of-concept and it is still lacking the functionality to stream the ogg-data directly to an IceCast-Server. Also the web-configuration and a “record”-button still needs to be implemented. Nevertheless transmitting an compressed OggFile to a NFS-server is already possible.

A running **NFS-Server is a prerequisite** needed to start the demonstration.

The OggStreamer is preconfigured to use a static IP-Address (**192.168.0.63**).

The following steps should allow you to stream an encoded OggFile to your NFS-Server.

1. Connect the Ethernet Port of the OggStreamer to your Linux host
2. Connect your favourite Audio-Source (like MP3-Player, etc.) to the Line-In of the OggStreamer
3. Connect the USB-Plug of the OggStreamer to your PC to supply the OggStreamer with +5 Volt DC
4. Make sure the Linux host is within the 192.168.0.0/24 Subnet
`Example: ifconfig eth0 192.168.0.1 up`
5. Make sure the NFS-server on your Linux Host is running
6. Telnet to the OggStreamer
`telnet 192.168.0.63`
7. Login with user: root and password: root
8. Start the portmap daemon
`portmap &`
9. Mount the NFS-server
`Example: mount 192.168.0.1:/tftpboot /mnt`
10. change to the NFS-directory
`cd /mnt`
11. Start saving the Ogg-File
`oggstreamer file.ogg`
12. press “play” on your audio source
13. Check the Volume on the VU-Meter
14. Press CTRL+C to end Ogg Streaming
15. On your Linux host change to the NFS-server directory and use vlc or a similar programm to replay the recorded audio-data.

Use Scenarios

The OggStreamer can be marketed as a standalone device or included into existing devices, as shown in the following non-exhaustive enumeration:

- Radio-station uplink
- Live-concert audio-streaming
- Internet-radio station
- High-end Mixing Desks
- Hifi Systems

The OggStreamer primary design goal was to provide the functionality to stream live-emissions from an far away studio to the main-studio, where then the audio-stream is encoded and broadcasted. But it can also be used to generate an audio-stream from, for examples, live-concerts, where a simple and reliable setup is needed. In case somebody wants to make his/her own Internet-radio, the OggStreamer is an easy-to-use, lowpower device to upstream the audio-data to an IceCast server.

The functionality of the OggStreamer can also be integrated in high-end mixing desks, which would be especially useful in the case of live-events. Or it can be integrated into hifi systems, allowing the streaming of audio-data within or even outside of the private network.

Costs of Production

To estimate the production cost of the OggStreamer-Devices the current schematics are referenced. In quantities of 1k the OggStreamer costs about 13 USD (excluding the XportPro, assembling and housing). See BOM for details.

Futher Optimizations

Eliminating the intermediate mikro-controller

There are basically two ways, how to save the costs of the intermediate mikro-controller. As the mikro-controller is kind of translating unit between the XportPro and the VS1053, with RS-232 on the one side and SPI on the other, a solution would be to make the XportPro “understand” SPI or to output the audio-data via RS-232 on the VS1053. Both approaches are theoretically feasible. On the XportPro there are again two approaches thinkable:

1. Implementation of a bit-banging SPI-Interface
2. A new generation of XportPro supporting the QSPI Interface of the MCF5208

The VS1053 already provides an UART for debugging purpose, which could be theoretically reprogrammed to transmit the audio-data. This would imply the modification of the ogg-encoder plug-in. As the ogg-encoder is publically only available in its binary form, such a modification would have to be carried out by VLSI itself. In this context I want to mention that VLSI recently decided to provide their internal SDK "VSIIDE"⁵ free of charge for the public. So maybe they are interested in open up other parts of their work as well.

However, the intermediate mikro-controller, also serves as a driver for the LEDs. So in cases where more than the 8 GPIOs provided by the VS1053 are used, a mikro-controller still might be a good choice.

Replacing VS1053 by VS8053

VLSI recently announced the availability of the VS8053⁶ chip, a cut-down version of the VS1053, which doesn't support MIDI and MP3 decoding. As the OggStreamer doesn't use either of these two features, the VS8053 can be used as a drop-in replacement. Because this device doesn't support MP3 decoding, VLSI doesn't have to pay fees for MP3 licensing, thus lowering the cost of the chip.

Implementation of an IceCast Server

The current OggStreamer is designed to upstream the audio-data to an IceCast-Server. As we run Linux on the XportPro it could be possible to port the whole IceCast Software to the XportPro. This would allow a small number of listeners (maybe around 5) to directly connect to the XportPro.

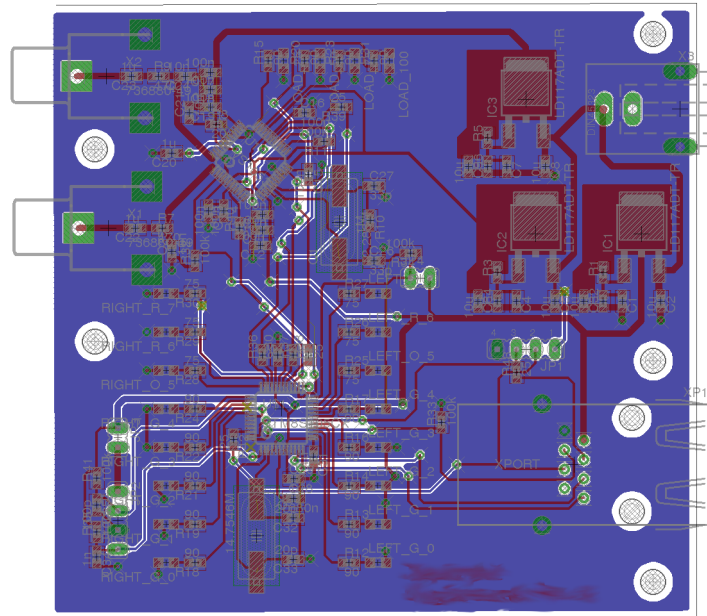
Storing the VS1053 Firmware on the XportPro

The XportPro holds enough Flash to store the VS1053 Firmware-plug-in. Moving the storage place of the VS1053 Firmware-plug-in from the STM8 to the XportPro requires a more elaborated protocol between the XportPro and the STM8. But on the other hand it would allow the usage of a much smaller intermediate micro-controller with about 4 KB of flash, and therefore further reduce cost. On the XportPro there is enough space to store even more than one Firmware-plug-in. Every Firmware-plug-in is hardcoded to a certain qualityprofile. Storing multiple Firmware-plug-ins on the XportPro would enable the user to choose a desired quality- and bandwidthprofile suiting his needs.

5 <http://www.vlsi.fi/en/support/software/vside.html>

6 <http://www.vlsi.fi/fileadmin/datasheets/vlsi/vs8053.pdf>

PCB Print



Used Software

- RS-232 Library
Homepage: <http://www.teuniz.net/RS-232/>
License: GPL Version 2
- STM8S Firmware Library
Homepage: <http://www.st.com/products/support/micro/files/stm8sfwlib.zip>
License:
The enclosed firmware and all the related documentation are not covered by a License Agreement, if you need such License you can contact your local STMicroelectronics office.

THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

- libshout
Homepage: <http://downloads.us.xiph.org/releases/libshout/libshout-2.2.2.tar.gz>
License: GPL Version 2

- libogg and libvorbis
Homepage: <http://downloads.xiph.org/releases/ogg/libogg-1.2.0.tar.gz>
<http://downloads.xiph.org/releases/vorbis/libvorbis-1.3.1.tar.bz2>
License:

Copyright (c) 2002-2008 Xiph.org Foundation

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Xiph.org Foundation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
FOUNDATION
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.