midOSC: a Gumstix-Based MIDI-to-OSC Converter

Sébastien Schiesser
Institute for Computer Music and Sound Technology
Zurich University of the Arts
Baslerstrasse 30, 8048 Zurich, Switzerland
sebastien.schiesser@zhdk.ch

Abstract

A MIDI-to-OSC converter is implemented on a commercially available embedded linux system, tighly integrated with a microcontroller. A layered method is developed which permits the conversion of serial data such as MIDI to OSC formatted network packets with an overall system latency below 5 milliseconds for common MIDI messages.

The Gumstix embedded computer provide an interesting and modular platform for the development of such an embedded applications. The project shows great potential to evolve into a generic sensors-to-OSC ethernet converter which should be very useful for artistic purposes and could be used as a fast prototyping interface for gesture acquisition devices.

Keywords: MIDI, Open Sound Control, converter, gumstix

1. Introduction

The Institute for Computer Music and Sound Technology (ICST) has been working for many years with Ambisonics: a set of recording and replay techniques for multichannel audio [8]. It has conducted research in advanced higher order Ambisonics algorithms and published spatialisation tools for Csound and Max/MSP [11]. It also supports concerts and organizes residencies for composers who want to use its Ambisonics facilities.

The ICST has developed the mobile Ambisonics equipment (mAe) for concert venues, which is able to play sound files or process live audio in a setup with up to 64 channels [2]. Due to the large size of this equipment and the fan noise caused by the processing computers, many devices are situated outside a concert hall and have to be remote-controlled via MIDI [9].

To avoid having an excessive amount of MIDI cables and to overcome their limitations in length [1], control signals are converted into Open Sound Control (OSC) format [14], sent to the remote-controlled devices location and converted back to MIDI. Until now, this has been done at each conversion point through a Max/MSP patch running on a computer connected to a MIDI interface. This is very demanding in terms of hardware: in the backstage system of the mAe, a computer is dedicated to conversion purposes only. And when MIDI devices are present on stage, an additional laptop with interface is required.

The mAe is intended to be modular and to support several "I/O hubs", where audio and control data are collected and dispatched. In order to avoid dependence on a converting computer at each hub, it seemed appropriate to use a dedicated converter which can run independently, be stacked in a rack and which would be considerably less expensive than a computer with MIDI interface.

Several devices with this capability already exist. Most of them can do much more than MIDI-to-OSC conversion, and thus are too expensive [7][5] or not commercially available [1][3][4]. A project of the University of Applied Sciences of Upper Austria was about a dedicated MIDI-to-OSC converter, but it had some drawbacks (no DHCP capabilities, no maintained drivers) and has never been completed [6]. The midOSC converter fills this gap by providing a solution, which is economical, flexible and open for future enhancements and modifications.

2. midOSC v1

Different possibilities have been evaluated in order to build a dedicated MIDI-to-OSC converter: microcontrollers, field-programmable gate arrays (FPGA) or embedded computers. The Gumstix embedded computers [2] offer the required features in terms of connectivity, as well as a versatile framework, which allows development of other tools beyond this specific application. Furthermore, working with a maintained operating system (OS) gives access to regular updates of drivers or communication protocols, which makes the product easier to use and to develop.

2.1. Hardware

Gumstix embedded computers are all based on the same principle: a motherboard with a given CPU speed, installed

[1] The MIDI Manufacturers Association recommends a maximum length for DIN cables of 15 meter, while OSC works with twisted pairs ethernet cables, which can be up to 100 meter long.

[2] http://www.gumstix.com [2009 Apr 3]

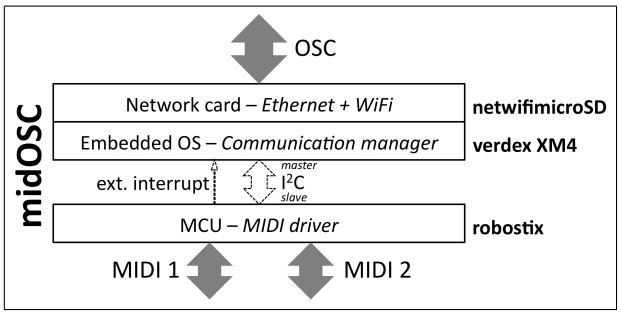

Figure 1. Gumstix cards and communication layers used in the midOSC converter

OS and which can be connected to a number of extension cards for specific purposes: network, sound, Global Positioning System (GPS), touchscreen, sensors... Many drivers are already installed in the base image and the OS (a Linux distribution, but Windows CE exists as third-party software) can be accessed and configured over a serial or ethernet connection.

The cross-compile environment OpenEmbedded (OE) [3] is used as a development platform to create and set up custom packages. A developer site, as well as a mailing list provide the necessary user documentation [4] .

The midOSC converter is based on a Gumstix *verdex XM4* motherboard with a 400 MHz PXA270 processor, connected to two extension cards: *netwifimicroSD* on one side for the ethernet connection and *robostix* on the other side for serial communication at custom baud rate. The robostix works independently with an Atmel ATmega 128 microcontroller unit (MCU) and communicates with the motherboard as a slave on an Inter-Integrated Circuit (I^2C) bus at 400 kHz (see Figure 1).

Since the ATmega 128 provides only two Universal Asynchronous Receiver/Transmitters (UARTs), the first midOSC version works with two MIDI ports. A second version is also planned with a custom developed serial interface, based on a ATmega 640 MCU, which provides four UARTs (more details about developments in Section 4).

Connectivity is provided on the OSC side by the network card through a standard ethernet socket, and on the MIDI side through a custom board, connected with a flexible flat cable to the robostix UART pins. The complete setup is shown in Figure 2.

2.2. System design

The implemented design is based on an interrupt and callback method between the MCU firmware (MIDI driver) and the linux user-space daemon written in C (Communication Manager).

[3] `http://www.openembedded.org` [2009 Apr 3]

[4] Developer site: `http://www.gumstix.net`; mailing-list: `http://www.nabble.com/Gumstix-f22543.html` [2009 Apr 3]

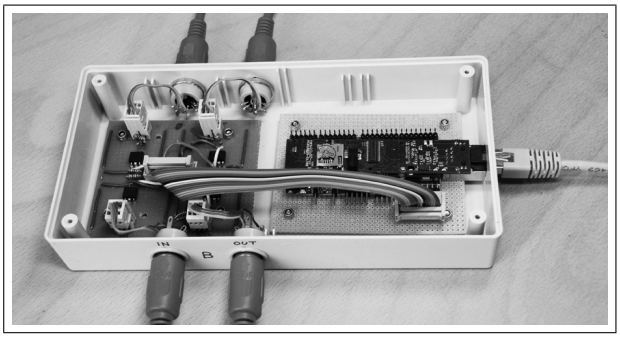

Figure 2. Packaging of the midOSC v1 converter with the Gumstix (right) and the custom built circuit board

The MIDI driver can be interrupted either by a byte reception on a MIDI port or by a I^2C call, when the Communication Manager (CM) has received some OSC data and needs to forward it to MIDI.

The CM consists of two threads, which react either to an incoming OSC message, or to an interruption on a General Purpose In/Out (GPIO) pin of the verdex board, signifying that the MIDI driver needs to send some MIDI data.

Since the I^2C protocol works on a master-slave configuration, the MIDI driver (slave) cannot directly interrupt the CM (master). It also has to use an external interruption on a GPIO line to ask the CM to start a data transfer.

Several other applications run on the lower levels of the OS, like the network and I^2C drivers, the GPIO interruption module and a webserver, which is used for remote configuration purposes. These applications are provided as standard packages in the Gumstix distributions and are used as is.

2.3. Communication

Currently, the midOSC setup works in a node configuration, *i.e.* each device is client of a router/DHCP server and has a MIDI port offset value, assigned by a rotary switch, which tells him the number of its first MIDI port.

A basic communication scheme in this configuration consists of the following sequence:

1. A MIDI message is received on a UART of one device and the communication manager is interrupted by the MIDI driver on a GPIO line
2. The CM sends an I^2C read request to the MIDI driver, which uploads its message
3. The CM creates an OSC message containing the MIDI port number and data, and sends it to the network
4. The OSC message is received by the other devices, interpreted and – if necessary – downloaded to their robostix, which assign it directly to their corresponding UART

Figure 3 shows the standard configuration of a port-following transmission: an incoming MIDI message is received on port 1 of device 1 and is broadcast with its port number to the OSC network. Every midOSC receives the

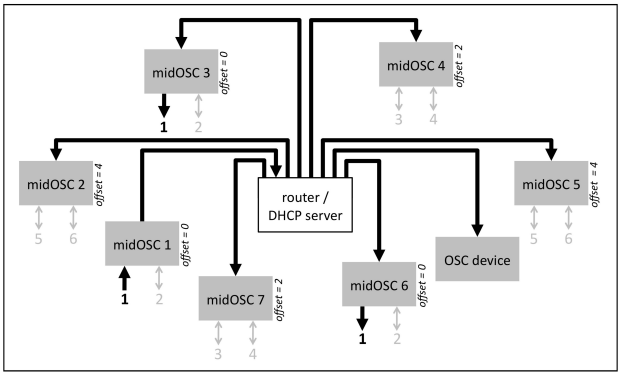

Figure 3. Port-following communication on a midOSC network: MIDI data is received on port 1, broadcast and forwarded by the other devices providing a MIDI port 1

data, but only those providing the same MIDI port number forward them.

Other types of routing can be achieved: general broadcasting (all ports of all devices are addressed), device broadcasting (all ports of a specific device) or strict routing (one specific port of a given device). The OSC address patterns are shown in Figure 4. Naturally, other OSC-enabled devices can communicate in a midOSC node, assuming they match the correct patterns. Furthermore, some improvements in connectivity will be discussed in Section 4.2.

3. Timing characteristics

One of the most critical features of a real-time system is its latencies. Therefore, timing measurements have been completed as first setup characterization.

Scope results of measurements (see Figure 5) show a communication sequence between two devices connected *via* a router. The latency – as defined by M. Nelson [10] and J. Wright [13] – for a midOSC setup is the sum of the incoming MIDI message time, GPIO interruption delay, message upload between MIDI driver and CM of the first device, network delay and message download between CM and MIDI driver of the second device. For a 3-byte MIDI message, the average latency is 4 ms, with a jitter of 0.4 ms and a standard deviation of 0.1 ms.

One drawback of this setup is due to the fact, that a message upload take part only when the complete MIDI sequence has been received. This can cause big latencies for long system-exclusive (SysEx) messages and is subject to buffer size limitations. However, the system is very competitive for common messages like note on/off or control/program changes. Furthermore, the improvements discussed in Section 4.1 will correct these both problems.

4. Discussion

In order for the NIME community to profit from midOSC, it is planned to make the project available as open hardware and electronics by sharing schematics, circuits and source

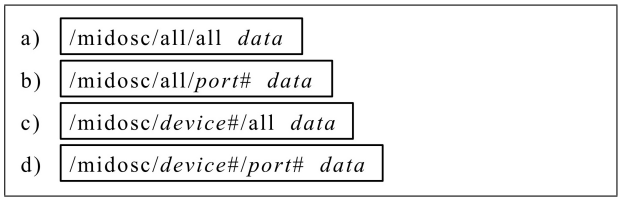

Figure 4. OSC address patterns: a) broadcasting b) port-following c) device broadcasting d) strict routing

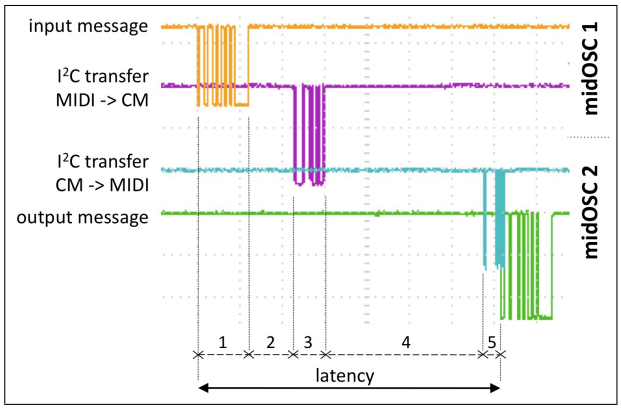

Figure 5. Timing sequence of a two-byte MIDI message. The latency is the sum of the incoming MIDI message time (1), GPIO interruption delay (2), message upload (3), network delay (4) and message download (5).

code on the ICST web site (`http://www.icst.net`). To reach this goal, some improvements have to be implemented.

4.1. Timing

The timing characteristics emphasised that the system latency can become too high for long MIDI messages. The best way to overcome this limitation is to transfer each byte directly after reception without delaying the system. And that for all available MIDI ports.

In order to achieve such timings, a faster communication protocol than I^2C has to be used. Both the ATmega MCUs and the PXA270 microprocessor support the Serial Peripheral Interface (SPI) bus, which has a much higher throughput than I^2C (e.g. up to 8 MHz baud rate for a MCU with a 16MHz clock frequency.). In addition, the currently used GPIO interruption method has to be accurately controlled temporally or replaced by another communication scheme.

When both tasks are implemented, the system latency will remain constant and is estimated to be about the same than the network delay, which in the current configuration is approximately 1 millisecond. Then the buffer size will not be a concern anymore.

4.2. Connectivity

For the I/O hubs of the mobile Ambisonics equipment, it is necessary to provide four MIDI ports.

As replacement for the robostix, a custom board, based

on a ATmega 640 MCU will be developed. This MCU provides four UARTs and a SPI interface, which will allow to optimize the timing characteristics in order to take full advantage of the four MIDI ports.

In order to be independent from any DHCP server, the Zeroconf/Bonjour [12] implementation already existing in the Gumstix base distribution will be used. Then it will be possible to connect two midOSC devices directly to each other or many devices to a simple ethernet switch without DHCP capabilities.

4.3. Perspectives

Beyond MIDI-to-OSC conversion, the Gumstix embedded systems offer a great range of possibilities for the development of custom applications. Furthermore, the robostix expansion card has many I/O pins exposed on its headers cleverly arranged with one supply and one earth for each channel. This provides standardized connections for in- and output from external devices and sensors.

Based on the midOSC framework, generic modules will be developed to enable low-latency bidirectional transmission of sensor data in OSC over an ethernet connection. This tool could be used as data-gathering and normalizing node for network performances, interactive installations or exhibitions.

The pedagogical value of this project should also be emphasized since this system will be used as a quick prototyping tool to interface gesture acquisition devices, thus providing a time and resource-saving experimentation platform.

5. Conclusion

midOSC shows the implementation of a low-latency bidirectional MIDI to ethernet converter based on a commercially distributed embedded linux-system which is coupled with a microcontroller. MIDI messages are converted to and from OSC formatted network packets, that can be transmitted in a variety of network morphologies.

For further developments, the current midOSC setup show potential pointing into two interesting directions. On the one hand it will be further optimized as low-latency four-port MIDI-to-OSC converter to cover the needs of conversion demanding applications like the mAe.

On the other hand, a generic and modular sensor-to-OSC ethernet converter will be implemented, which, by taking advantage of the available robostix connectivity, will provide a quick prototyping tool for pedagogical and artistic purposes.

6. Acknowledgements

Thanks to Jan Schacher for fruitful conversations and improvement ideas. Thanks also to Peter Faerber for the introduction to the mobile Ambisonics equipment and the whole ICST staff for support, brainstorming and office sharing.

References

[1] R. Avizienis, A. Freed, T. Suzuki, and D. Wessel. "Scalable Connectivity Processor for Computer Music Performance Systems," in *Proc. of the Interfnational Computer Music Conference (ICMC)*, Berlin, 2000

[2] P. Faerber. "The Mobile Ambisonics Equipment", [Web site] 2006, [2009 Apr 3], Available: `http://www.icst.net/index.php?show=163`

[3] E. Fléty. "EtherSense : sensors-to-OSC digitizing interface," [Web site] 2009, [2009 Apr 3], Available: `http://recherche.ircam.fr/equipes/temps-reel/movement/hardware/index.htm`

[4] E. Fléty. "The Wise Box: a Multi-performer Wireless Sensor Interface using WiFi and OSC," in *Proc. of the Conference on New Interfaces for Musical Expression (NIME)*, Vancouver, 2005, pp. 266–267

[5] A. Fraietta. "The Smart Controller Workbench," in *Proc. of the Conference on New Interfaces for Musical Expression (NIME)*, Vancouver, 2005, pp. 46–49. See also: `http://www.smartcontroller.com.au` [2009 Apr 3]

[6] G. Gessert. "mOSCito: multiple OSC performance controller," [Web site] 2006, [2009 Apr 3], Available: `http://wwwhsse.fh-hagenberg.at/Projekte/mOSCito`

[7] S. Kartadinata. "the gluion. advantages of an FPGA-based sensor interface," in *Proc. of the Conference on New Interfaces for Musical Expression (NIME)*, Paris, 2006, pp. 93–96. See also: `http://www.glui.de` [2009 Apr 3]

[8] D. G. Malham. "Ambisonics - A Technique for Low Cost, High Precision Three-Dimensional Sound Diffusion," in *Proc. of the International Computer Music Conference (ICMC)*, Glasgow, 1990, pp. 118–120. See also: `http://www.ambisonic.net` [2009 Apr 3]

[9] MIDI manufacturers association. "The complete MIDI 1.0 detailed specifications," [Web site] 2008, [2009 Apr 3], Available: `http://www.midi.org/techspecs/index.php`

[10] M. Nelson, and B. Thom. "A Survey of Real-Time MIDI Performance," in *Proc. of the Conference on New Interfaces for Musical Expression (NIME)*, Hamamatsu, 2004, pp. 35–38

[11] J. C. Schacher, P. Kocher. "Ambisonic Spatialization Tools for Max/MSP," in *Proc. of the International Computer Music Conference (ICMC)*, New Orleans, 2006

[12] D. H. Steinberg, S. Cheshire, *Zero Configuration Networking: The Definitive Guide*, O'Reilly, 2005

[13] J. Wright. "System-Level MIDI Performance Testing," in *Proc. of the International Computer Music Conference (ICMC)*, Havana, 2001

[14] M. Wright, A. Freed, and A. Momeni. "OpenSound Control: State of the Art 2003," in *Proc. of the Conference on New Interfaces for Musical Expression (NIME)*, Montreal, 2003, pp. 153–159