A Prototypical In-Car Entertainment Setup Using Software Defined Radio and Ethernet/IP-Based In-Vehicle Communication

Lothar Stolz, Kay Weckemann, and Hyung-Taek Lim
BMW Group
Research and Technology
Munich, Germany
{*Lothar.Stolz,Kay.Weckemann,Hyung-Taek.Lim*}*@bmw.de*

Walter Stechele
Institute for Integrated Systems
Technical University of Munich
Munich, Germany
walter.stechele@tum.de

***Abstract*—We introduce a novel approach to in-vehicle radio entertainment platforms. Using a prototypical automotive setup for car-radio reception, we evaluate a Digital Audio Broadcasting receiver implemented as a Software-Defined Radio. The radio runs as an application on a low-cost x86 device. Furthermore, we distribute the digital audio stream and control information via a standard Internet Protocol/Ethernet link instead of a proprietary automotive field bus technology. Thus, the car demonstrator distinguishes from today's approaches to automotive radio entertainment by increased flexibility on the computation as well as on the communication side, together with potential savings on costs.**

***Keywords*-DAB, SDR, IP, Ethernet, in-vehicle networking.**

I. Introduction

Radio entertainment receivers were introduced into cars as a mass product in the 1960s. For broadcasting, the scenario of FM and AM analog radio reception kept almost stable for almost 50 years. However, with the advent and progress of digital communication, the number of worldwide broadcasting standards increases steadily. New standards arise quickly and Application Specific ICs (ASICs) for radio hardware undergo revisions in short intervals of a few months. On the other hand – especially compared to consumer electronic devices – the product life cycle of automotive Electronic Control Units (ECUs) is much longer. Software-Defined Radio (SDR) describes the paradigm of implementing most of the signal processing of a radio device on a reprogrammable compute platform [1]. Hereby, a high degree of flexibility is promised as the device can adopt its signal processing to potentially any introduced transmission scheme as long as its compute resources are sufficient.

We will achieve the highest grade of flexibility, when we co-use the automotive entertainment unit's application processor instead of having a dedicated decoder IC within the platform. So, we investigate the feasibility of implementing the whole signal processing on the head unit, exemplified by a low-cost general-purpose CPU. We demonstrate an all software-defined radio receiver for digital radio broadcast being compliant to the Digital Audio Broadcasting (DAB) standard.

The in-vehicle network is currently changing due to the increasing bandwidth demand of Advanced Driver Assistance Systems (ADAS) [2]. There is a paradigm shift from the proprietary automotive bus systems to standard technology such as Ethernet to realize a communication very cost-effective. The introducing of two-wire unshielded cables with adaptive physical layer of 100 Mbit/s Ethernet makes Ethernet more interesting for the in-vehicle communication, which is considered by the OPEN ALLIANCE [3].

In this research work, we communicate over an Ethernet link instead of a proprietary automotive field bus technology. For distributing multimedia content in the automotive domain, the proprietary Media Oriented Systems Transport (MOST) field bus is employed today. We demonstrate a distributed in-vehicle audio system relying on Internet Protocol (IP)/Ethernet-based communication – a proven consumer electronics technology recently coming into focus for demanding automotive use cases [4]. For real-time audio streaming, the Real-Time Protocol (RTP) is used. The control flow is done using an IP-based Remote Procedure Call (RPC) framework. Therefore we adapted the open source solution Apache Etch [5] [6].

The remainder is structured as follows: we first introduce the three main concepts used for the components of the demonstrator and establish the relation to the presented use case from automotive domain. Afterwards, we report on the prototypical implementation of the overall system.

II. Software Defined Radio

SDR-like functionality for the mass markets are implemented mostly on dedicated Digital Signal Processors (DSPs). Shifting the signal processing to the central application processor will require a more powerful CPU. This is due to additional workload as well as to the fact that the CPU now is a general-purpose processor rather than an application specific one. However, integrating two processor systems into one will potentially save costs, and also save space on the printed circuit board.

 ISBN: 978-1-61208-241-7

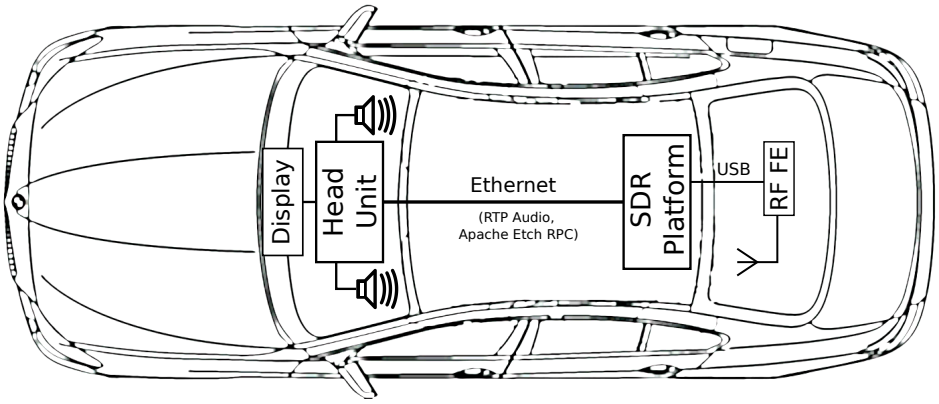

Figure 1. The test vehicle system architecture.

A. Digital Audio Broadcasting

The DAB system was introduced by the Eureka 147 working group in order to replace the analogue FM radio transmissions. It is standardized in ETIS EN 300 401 [7]. DAB is a typical representative of an OFDM-based transmission scheme. The DAB RF channel has an RF bandwidth of 1536 kHz. One channel carries an ensemble being comprised of multiple audio and/or data services. Hereby, audio data is grouped into payload chunks of 24 ms each and subsequently compressed using an MPEG 1 Layer II coder. Lately, the standard was extended to support also MPEG HE AAC as a second codec, providing state-of-the-art source encoding at reduced bit rates. This is known as the DAB$^+$system [8]. Having lots of techniques in common, DAB can be seen as an audio/high-mobility companion to the digital television system Digital Video Broadcasting Terrestrial (DVB-T),

B. The Software Demodulator

For the development of a DAB-compliant OFDM baseband signal processing chain, we follow the standard [7] and the algorithmic concepts outlined in [9] [10], and bring them to a software defined radio implementation. By extensive optimization, especially by applying hand-tuned assembler code using x86 signal processing SSE instruction set extensions, we obtain a highly optimized software radio processing chain [11]. As input, the digitized raw complex baseband is expected at a sampling rate of 2048 kSa/s. Having complex-valued samples at 16 bit each, this results in a data rate of 64 Mbit/s, which has to be provided by the RF frontend. We chose the USB interface to bridge this system component to the compute platform. Furthermore, the bus allows a distant placement of the frontend, for instance, close to the antenna, in order to minimize RF cabling losses. Table I gives the CPU load shares for the SDR part deployed to the test platform being introduced later.

Software receiver component	CPU Load*)
DAB signal processing	7.4%
RF baseband acquisition via USB	2.9%
Operating system / threading overhead	1.5%
Total	11.8%

*) at DAB service bit rate $r = 192$ kbit/s

Table I
CPU LOAD DISTRIBUTION FOR THE DAB SOFTWARE RECEIVER ON THE SELECTED LOW-COST TEST PLATFORM.

III. IP-BASED AUDIO STREAMING

Each 24 ms, one complete MPEG frame of encoded audio gets available from the DAB receiver core. The resulting payload size p in byte can be calculated by equation

$$p = \frac{1}{8} \cdot 24\,\text{ms} \cdot r \qquad (1)$$

where r is the bit rate of the received DAB audio service stream. Assuming an exemplary bit rate of 192 kbit/s for instance, we will encounter a consistent MPEG frame size of 576 byte.

The Real-Time Protocol is being defined in RFC3550 [12]. Basically, it adds time stamp data to allow the information sink to provide on-time presentation

Protocol	Header Size
Ethernet*)	18 byte
IP	20 byte
UDP	8 byte
RTP	12 byte
Total	58 byte

*) no VLAN-tag.

Table II
ABSOLUTE DATA OVERHEAD INTRODUCED BY RTP/IP-BASED AUDIO STREAMING OVER ETHERNET.

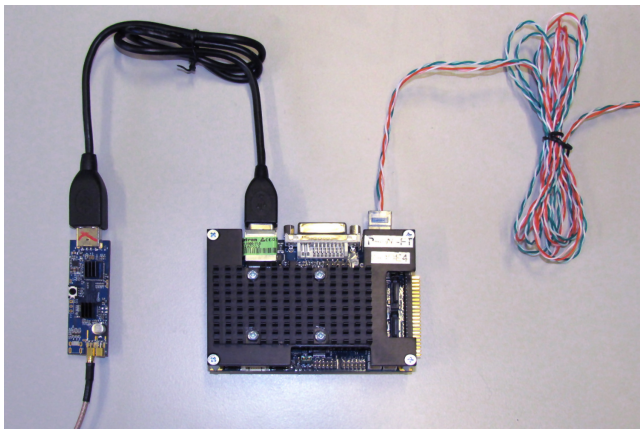

Figure 2. The SDR platform connects via USB to the RF-preamplifier/frontend, and via Ethernet over unshielded twisted pair to the head unit emulator.

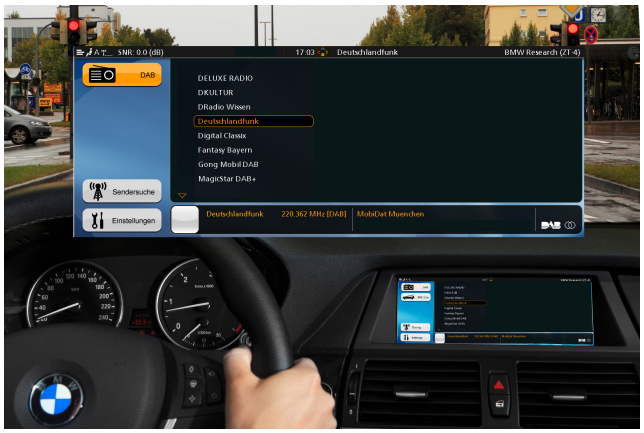

Figure 3. The head unit emulator uses an embedded PC similar to the SDR platform. The graphical user interface is integrated to the car HMI.

Audio Rate r	Overhead
64 kbit/s	30.2%
96 kbit/s	20.1%
128 kbit/s	15.1%
192 kbit/s	10.1%
256 kbit/s	7.6%

Table III
RELATIVE DATA OVERHEAD INTRODUCED BY RTP/IP-BASED AUDIO STREAMING OVER ETHERNET FOR TYPICAL DAB AUDIO BIT RATES r.

of content and to cope with loss of data. Due to the IP protocol stack hierarchy, overhead is introduced by the packet headers of all corresponding layers: Ethernet, IP, UDP and RTP headers together add 58 byte to each packet (see Table II), which increases the effective total streaming rate r_{total} on Ethernet frame layer by 19.33 kbit/s.

$$r_{\text{total}} = r + \frac{58 \cdot 8\,\text{bit}}{24\,\text{ms}} = r + 19.33\,\text{kbit/s} \qquad (2)$$

Again assumed for an exemplary audio rate of 192 kbit/s, the IP-based audio transport approximately adds 10% of bandwidth due to protocol overhead (Table III).

Such overhead could potentially be decreased by grouping packets of multiple 24 ms MPEG frames, however adding latency in exchange. For an audio service rate $r = 256$ kbit/s being presumed as the maximum rate, r_{total} is 275 kbit/s. This is well feasible on an 100 Mbit/s Ethernet link.

Ethernet frames are limited in size by the so-called Maximum Transmission Unit (MTU). The payload must have no more than 1500 byte, otherwise it must be fragmented on higher protocol layers. For r =256 kbit/s, a complete 24 ms audio packet will have a size of p =768 byte plus the headers of IP, UDP, RTP (40 byte). This is well below the MTU value, thus no fragmentation is needed.

IV. IP-BASED MIDDLEWARE

Besides the streaming data, the prototypical receiver implementation has to announce station lists to and receive tuning commands from a distant controller – in general the Human Machine Interface (HMI). Middleware abstracts the communication from the application. Following a definition from Issarny et al. [13], middleware defines an Interface Description Language (IDL), a high-level addressing scheme, a coordination model based on an interaction paradigm and semantics, a transport protocol, and a naming and discovery protocol including conventions to publish and discover resources. The concept of Remote Procedure Calls as interaction paradigm fulfills the required functionality well. For the prototype, we used the open source RPC framework Apache Etch [5]. As the existing Apache Etch bindings primarily target fully-featured workstations, we implemented our own light-weight binding that is targeted at embedded computing requirements [14].

V. CAR RADIO RECEIVER PROTOTYPE AND PRELIMINARY RESULTS

As we focus on low-cost platforms, the compute platform selected for the prototype is based on the Intel Atom single core CPU (Table IV). As working memory, 512 MB of RAM are mounted. A prototypical RF frontend peripheral is located close to the antenna and connected point-to-point via USB bus to the SDR platform. The antenna input signal is mixed down to baseband, digitized and sent to the SDR host.

To demonstrate the distribution between a head unit ECU as sink and an external tuner ECU as source, we use Ethernet over a pair of unshielded twisted pair cables (U/UTP). Within this setup, the associated head unit is emulated by an identical x86 platform. Via an LVDS-converter the in-car display is connected. Furthermore, the car's main haptic controller *iDrive* is attached by a Controller Area Network

 ISBN: 978-1-61208-241-7

Platform	Intel Atom
CPU micro architecture	x86 in-order ("Bonnell")
No. of cores	1
Core clock	1600MHz
DRAM type	DDR2-667, single ch.

Table IV
TEST PLATFORM USED FOR PROFILING.

System Component	Occupied Resources
CPU Load	
DAB SDR Receiver (Tuner)	up to 15%*)
MPEG Audio Playback (Head Unit)	up to 6%*)
Bus Load	
Audio Streaming Bus (Ethernet)	up to 275 kbit/s*)
RF Frontend Bus (USB)	64 Mbit/s

*) depending on actual DAB service bit rate $r \leq 256$ kbit/s

Table V
DEMAND OF RESOURCES FOR THE PRESENTED DEMONSTRATOR SETUP.

(CAN) adapter. To receive, decode and playback the MPEG audio stream, the head unit setup relies on the Videolan VLC framework. Analog audio output is brought by auxiliary line input to the in-car sound amplifier.

Table V summarizes the required system resources for the prototypical setup of this use case.

VI. CONCLUSION AND FUTURE WORK

We presented a test vehicle-integrated prototype of a real-time DAB receiver as a novel approach to automotive in-vehicle entertainment: rather than using custom protocols from the automotive domain and proprietary application specific ICs, software solutions and open standards from consumer electronics were evaluated. We proved that real-time/on-air radio signal processing using software-based demodulation turns out to be possible at reasonable load on a general-purpose CPU for a digital broadcasting receiver. Furthermore, we showed that stable in-vehicle distribution of audio and control relying on IP and Ethernet is feasible. The results gained in the entertainment/broadcast audio domain encourage the evaluation of other use cases. Ongoing activities in the field of future IP-based in-vehicle network architecture will be continued. Studies on SDR evaluation will be intensified and extended from unidirectional broadcast radio to bidirectional data communications to derive further in-depth knowledge.

REFERENCES

[1] W. Tuttlebee, *Software-defined radio: facets of a developing technology*, IEEE Personal Communications, Vol. 6, pp. 38–44, 1999.

[2] H.-T. Lim, B. Krebs, L. Völker, and P. Zahrer, *Performance Evaluation of the Inter-Domain Communication in a Switched Ethernet Based In-Car Network*, 36th Annual IEEE Conference on Local Computer Networks (LCN), pp. 101–108, Bonn, Oct. 2011.

[3] OPEN Alliance, *OPEN Alliance SIG*, [Online]. Available: http://www.opensig.org/about.php [Mar. 2012].

[4] L.L. Bello, *The Case for Ethernet in Automotive Communications*, Special Issue on the 10th International Workshop on Real-Time Networks, Vol. 8. No. 4, 2011.

[5] Apache Etch community, *Apache Etch*, [Online]. Available: http://incubator.apache.org/etch [Feb. 2012].

[6] K. Weckemann, H.-T Lim, and D. Herrscher, *Practical experiences on a communication middleware for IP-based in-car networks*, Proceedings of the International Conference on Communication System Software and Middleware (COMSWARE), Verona, Italy, 2011.

[7] ETSI, *Radio Broadcasting Systems: Digital Audio Broadcasting to mobile portable and fixed receivers* ETSI EN 300 401 V1.4.1, European Telecommunications Standards Institute, Jan. 2006.

[8] ETSI, *Digital Audio Broadcasting (DAB); Transport of Advanced Audio Coding (AAC) Audio*, ETSI TS 102 563 V1.2.1, European Telecommunications Standards Institute, May 2010.

[9] F. van der Laar, N. Philips, and J. Huisken, *Towards the next generation of DAB receivers*, EBU Technical Review Summer 1997, pp. 46–59.

[10] K. Taura, M. Tsujishita, M. Takeda, H. Kato, M. Ishida, and Y. Ishida, *A digital audio broadcasting (DAB) receiver*, IEEE Transactions on Consumer Electronics, Vol. 42, No. 3, pp. 322–327, Aug. 2002.

[11] L. Stolz, M. Feilen, and W. Stechele, *An Optimized Software Defined DAB Receiver for x86 Platforms*, Proceedings of the Workshop on Software Radio, Vol. 8, Karlsruhe, Mar. 2012.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550, Internet Engineering Task Force, July 2003.

[13] V. Issarny, M. Caporuscio, and N. Georgantas, *A Perspective on the Future of Middleware-based Software Engineering*, Proceedings of the Future of Software Engineering, 2007.

[14] K. Weckemann, F. Satzger, L. Stolz, D. Herrscher, and C. Linnhoff-Popien, *Lessons from a Minimal Middleware for IP-Based In-Car Communication*, to appear in: Proceedings of the IEEE Intelligent Vehicles Symposium, 2012.

 ISBN: 978-1-61208-241-7