computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

WEB2.0 TECHNOLOGY FOR AN EMBEDDED WEB-BASED GATEWAY PLATFORM FOR SPATIALLY DISTRIBUTED WIRELESS NETWORKS

Axel Sikora

Baden-Württemberg Cooperative State University Lörrach, Hangstraße 46-50. 79539 Lörrach, Germany
sikora@dhbw-loerrach.de, http://www.dhbw-loerrach.de

Abstract: *A novel communication platform is presented, which helps in the monitoring and deployment of distributed wireless networks. Its major part is based on distributed embedded web servers connected to RF-communication heads. The web servers act as distributed communication hubs and exchange data via XML-feeds with web2.0-capable clients.*

The first implementations are concentrating on the monitoring direction, as this approach eases supervision of spatially distributed wireless networks, and also allows seamless remote monitoring. But it is also capable to feed data frames into the wireless network.

The platform already supports protocols like Wireless M-Bus and EnOcean Radio Protocol (ERP), but is flexible to integrate arbitrary protocols. In addition, this is - to the very best knowledge of the authors - the very first AJAX implementation on a very lean embedded web server.

Keywords: *Wireless Networks, Sniffer, Web2.0, AJAX, Embedded Internet, Gateway*

1. INTRODUCTION

The test of wireless protocols is always a hurdle, because of the wide variety of states and parameters at the different nodes to be tested. This especially holds true, if the main objective of the test is in the distributed functionality of the wireless network. Those networks might come not only with tens, but potentially with hundreds of individual nodes, which might be the case with wireless sensor networks for ambient intelligence [1]. In this case, problems may occur not only during programming, but also during commissioning and operation [2].

Sniffer and commissioning tools are widely available. There can be found three categories of those tools:

1. proprietary tools with support for only one protocol. Those are mostly delivered from the manufacturers of proprietary wireless protocols. Examples are available from [3] or [12], although not being listed as standard products.
2. commercial tools with support for one more than one standard protocol [5].
3. open-source tools with generic support for arbitrary protocols. Best and presumably most widespread example is Wireshark [6].

In addition, we see different layers, where those tools work:

1. There are tools working on the physical level to support physical dislocation of the nodes.1
2. Other tools perform logical decoding and support network analysis.

All those tools are based on fat client architectures. Thus they require a PC or PDA equipped with an RF-frontend, where the sniffer tool is installed. Consequently, mostly local monitoring can be supported.

2. STATE OF THE ART

The In the field of wireless testbeds, we are aware of mainly two approaches. On the one hand, a variety of commercial testbeds concentrate on the RF-characteristics of the nodes and the channel. They allow prototype measurements and production tests. On the other hand, academic publications in the field of testbeds mainly deal with the provision of generic hardware platforms, e.g. [7] [8] [9].

However, these approaches do not sufficiently target the communication aspects of the network nodes, but concentrate on the sensing and processing aspects.

There is only a small number of publications

[1] The development of this platform is partially supported from the Federal Ministry of Economics and Technology (BMWi) within the DEMAX project („Decentralized Energy and Network Management Using Flexible Energy Prices“), 16IN0594.

known to the authors, that monitor and control the architectural aspects of distributed wireless networks:

- [10] presents moteLab, an environment using networked "backchannel" interfaces for remote reprogramming and monitoring the sensor nodes, and a centralized web server.
- [2] describes SNIF, which uses a deployment support network (DSN) [11], a wireless network that is temporarily installed alongside the actual sensor network during the deployment process, selection algorithms, and a graphical user interface.

3. OBJECTIVES & REQUIREMENTS

Our system shall fulfill the following requirements:

- It shall reliably support spatially distributed monitoring and commissioning with a direct and bidirectional access to the wireless nodes.
- It shall be low cost in terms of hardware and installation efforts.
- It shall support long-term monitoring with or without online connectivity.
- It shall support remote access via wide-area networks.
- It shall be comfortable to use and make all available functionality be easily accessible.

4. SOLUTION

Physical Testbed

Our testbed installation includes the following elements, which are shown in Fig. 1:

- management nodes (*M* in Fig. 1), which allow to monitor and control the network. All kinds of management can be supported: passive observation (passive sniffing), semi-active observation (additional management frame exchange with the network node) and active injection of management or data frames into the network. The management nodes come with a novel web approach described below.
- server nodes (*S*' and *S* in Fig. 1), which collect the data from the management node, or the other server nodes,
- a client computer (*C* in Fig. 1) that accesses the input from the management nodes. As management traffic is pure http and XML, there is only a single requirement to the client computer: It shall be capable to run a JavaScript enabled web client. Tests were performed with PC platforms, but also with portable communication devices, such as PDAs and iPhones.

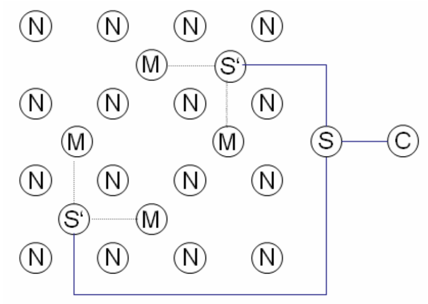


Fig. 1 – Monitoring nodes (M) are connected to servers (S), which collects the data exchanged between the wireless nodes (N)

During the development phase, two types of installations were used:

- a physical test bed with spatially distributed nodes, where the communication channels are guided along wired RF-links and controlled by attenuation elements. Fig. 2 shows the test bed, which was used for prototyping and automated test runs.
- real life test installations in various office buildings.

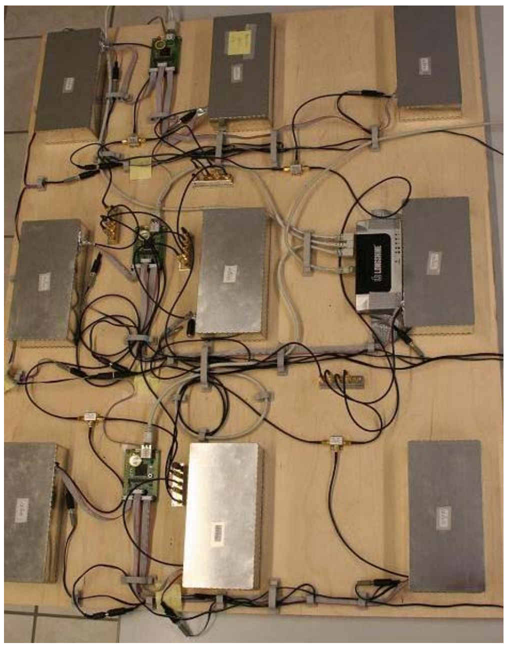


Fig. 2 – Physical testbed for automated verification of routing mechanisms

For ease of development and testing, two further elements were established

- automatic test scripting in support of regression testing. Those scripts are included into the firmware of the network nodes.
- a rich-client management tool to observe and to control the distributed nodes. The functionality of this tool is now transferred to the web based management nodes. However, this tool already was connected to the distributed management nodes via TCP/IP, but uses a proprietary

application protocol.

AJAX for Embedded

Although, web servers on embedded devices are around for more than one decade [14] [15], it is still worth mentioning that the integration into a standardized client server architecture allows decisive advantages. This is mainly due to the fact that a thin web client can be used and all the functionality can be downloaded from the server without further installations at the client computer.

It is especially this advantage, which is extended with client-side scripting, as not only html-code, but functional scripts can be transferred online to the client and executed there. Thus all performance intensive computing is transferred onto the client platform while remaining completely platform independent.

In addition, web2.0 comes with further advantages, which makes it suitable for use in embedded internetworking. From a technical point of view, Web2.0 is based on AJAX (Asynchronous JavaScript and XML) and the architecture shown in Fig. 3. It additionally allows dynamic loading of single XML-fields via XMLHttpRequest and thus significantly may help to reduce traffic volume. Usually Ajax toolkits are used to accomplish these tasks.

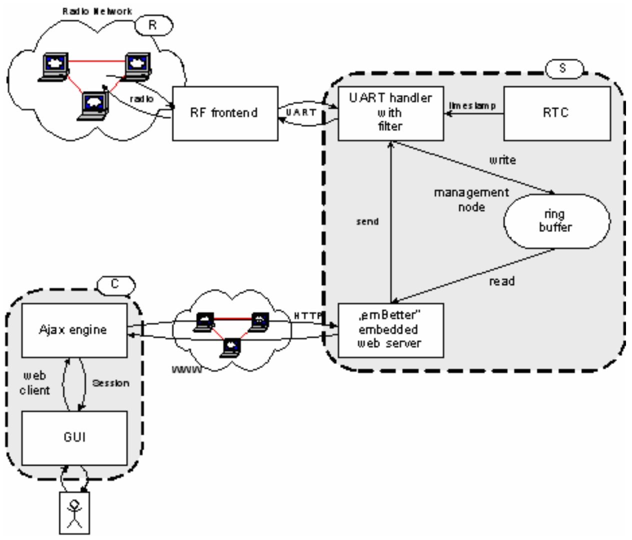


Fig. 3 – Software architecture of embedded web2.0 web server in radio networks

During a product scouting for a suitable JavaScript toolkit [16], which would be lean enough to be located at the embedded web server and compatible with the JavaScript interpreters of modern web browsers, two toolkits were identified:

- MooTools [17] [18] is a compact, modular, object-oriented JavaScript framework,
- jQuery [19] is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Both showed good performance, while allowing compiled libraries sizes of around 17 kBytes (minified and compressed). On the other hand these libraries provide a lot of functionality that is not needed. The last and most important argument for not taking advantage of existing JavaScript toolkits is their size. Although 17 kB for a JavaScript library (like JQuery or MooTools) is small, it would have more than doubled the size of our whole JavaScript code.

Therefore, it was decided implement small custom methods. They are designed to be easily adaptable to new protocols and concentrate around sorting and filtering.

Consequently, cost can be kept low, and web technology can be used at every single management node, and not only at one central database server, as compared to [6].

Hardware Platform

A PCB was developed to operate the system. It is shown in Fig. 4 and contains the following elements:

- a ColdFire MCF52235 microcontroller [20], which features 256KB Flash, 32KB RAM, a 10/100Mbit/s Ethernet and three RS232 interfaces. It has a clock rate of 60MHz and a Real-Time-Clock with a resolution of 31.25 µs, which is high enough to generate a unique timestamp for every telegram in the ring buffer. The ring buffer is also implemented on this controller as a temporary storage for up to 256 telegrams.

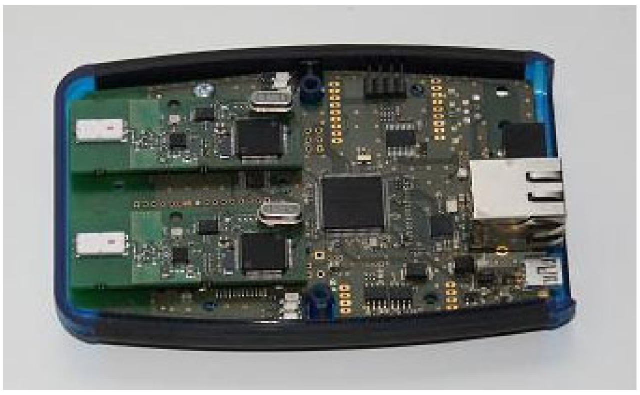


Fig. 4 – PCB for gateway platform

- Interfaces to wireless modules to get access to the wireless network. The interfaces are connected via SPI. As for some wireless protocols, more than RF-frequency, as for example with EnOcean Radio Protocol [12] or Wireless M-Bus [13]. Therefore, two wireless modules can be connected to act simultaneously, which can also be seen in Fig. 2.
- Serial interfaces can also be used to connect to DSN network. Available modules include WLAN for easy-to-install local infrastructure and

GPRS for remote and wide-area-network access.
- USB on the-go allows “infinite” local storage of monitored data.

Software Architecture

The software architecture from the Coldfire MCU contains the following elements:
- The heart is the emBetter embedded web server [21];
- A UART-handler which reads and writes the data to the wireless modules.
- The web server software gains access to these telegrams via an exposed API, which allows for retrieval of specific telegrams as well as initialization of the buffers and deletion of the content.
- The data is retrieved from a web-client via HTTP. Whereas the whole web page including the Java Script Libs has to be downloaded at the first run, after that it is only the XML-Feeds that follow. Thus, the communication channel can be kept very lean. It should be highlighted that all functionality is performed on the client, i.e. display, sorting, filtering, storing.

Figs. 5, 6 and 7 give an impression on the functionality of the generated web pages. Additional features, such as filtering, sorting and export of captured frames are supported within the JavaScript-engine of the client and leave the embedded web-server without this additional load.

The practical experience during the use of the testbed and in various test campaigns was very positive. It allowed excellent visibility of the states and frames during the development of firmware for various wireless protocols, and enabled extended and remote monitoring capabilities during the operation of the networks.

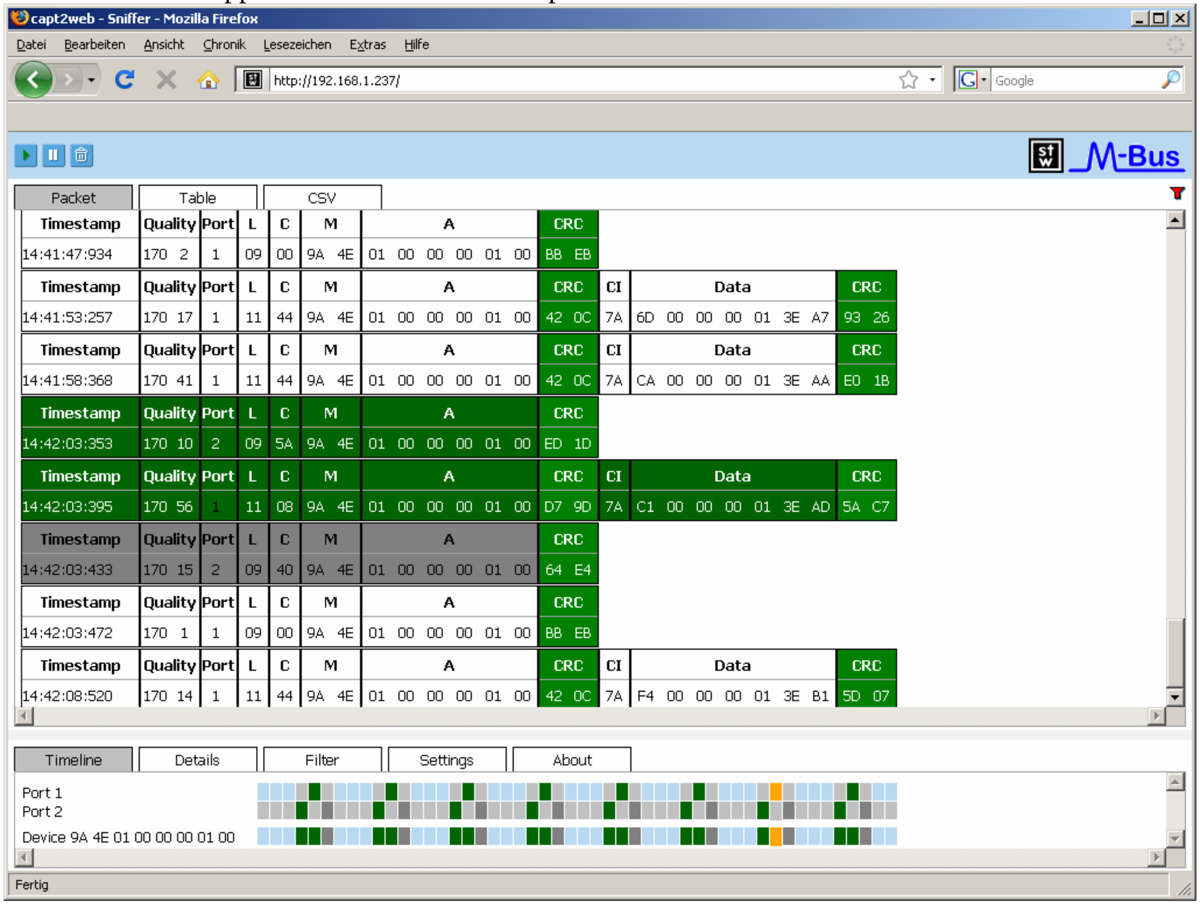
In addition, it could be observed that the data rate of the connections amongst the sniffers was as low as anticipated, i.e. data frames plus http and xml overhead.

5. OUTLOOK

The gateway platform is now being used in real installation for Wireless M-Bus and EnOcean Radio Protocol. Further work will include extension to other protocols, use of WLAN for the DSN, and time synchronization for improved filtering in spatially distributed networks.

The main challenges remain with the security of the intellectual property of protocols, as those details will be readable from the JavaScript code.

In addition, JavaScript does not allow access to multiple servers for security reasons. Consequently, a JavaScript based portal server is not at hand.

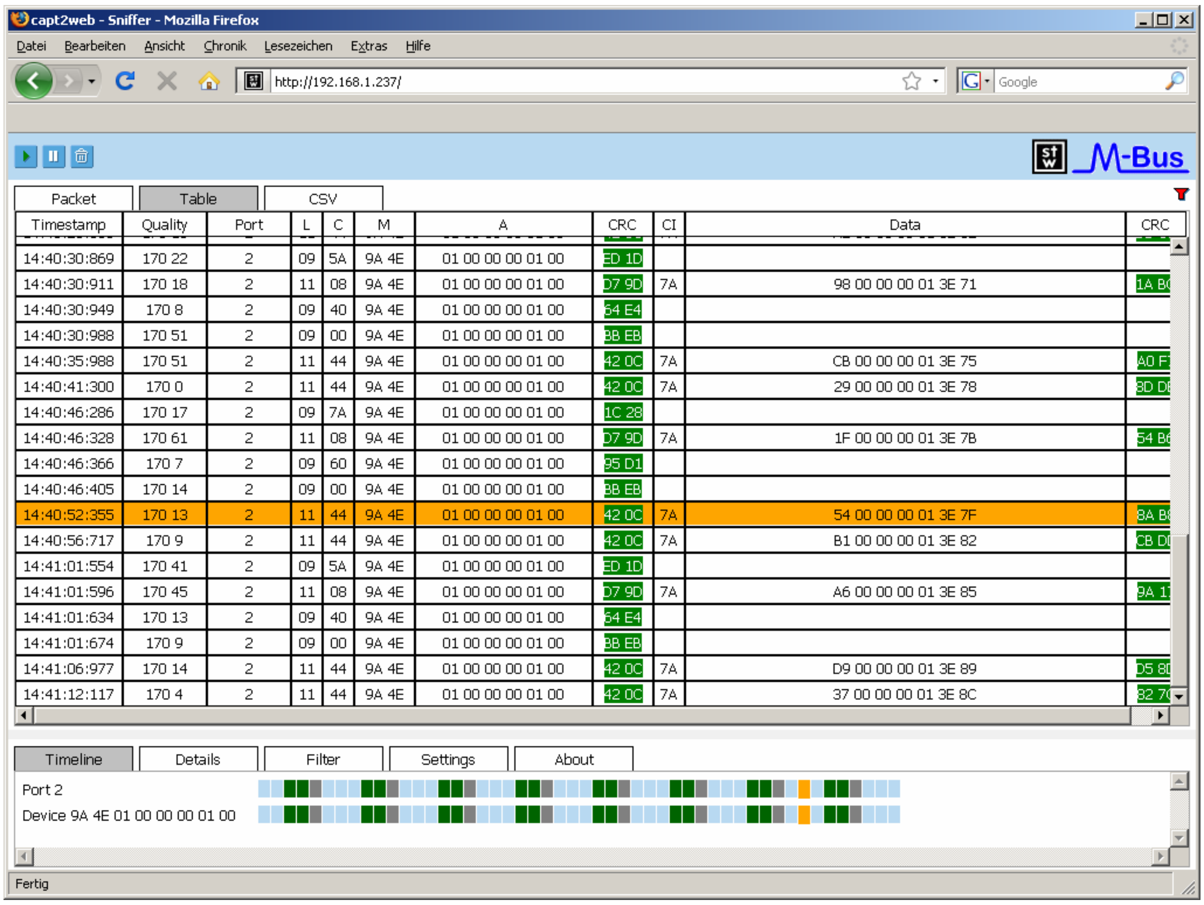


capt2web - Sniffer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://192.168.1.237/
Google
M-Bus
Packet
Table
CSV
Timeline
Details
Filter
Settings
About
Fertig

Timestamp	Quality	Port	L	C	M	A	CRC	CI	Data	CRC
14:41:47:934	170 2	1	09	00	9A 4E	01 00 00 00 01 00	BB EB			
14:41:53:257	170 17	1	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	6D 00 00 00 01 3E A7	93 26
14:41:58:368	170 41	1	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	CA 00 00 00 01 3E AA	E0 1B
14:42:03:353	170 10	2	09	5A	9A 4E	01 00 00 00 01 00	ED 1D			
14:42:03:395	170 56	1	11	08	9A 4E	01 00 00 00 01 00	D7 9D	7A	C1 00 00 00 01 3E AD	5A C7
14:42:03:433	170 15	2	09	40	9A 4E	01 00 00 00 01 00	64 E4			
14:42:03:472	170 1	1	09	00	9A 4E	01 00 00 00 01 00	BB EB			
14:42:08:520	170 14	1	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	F4 00 00 00 01 3E B1	5D 07

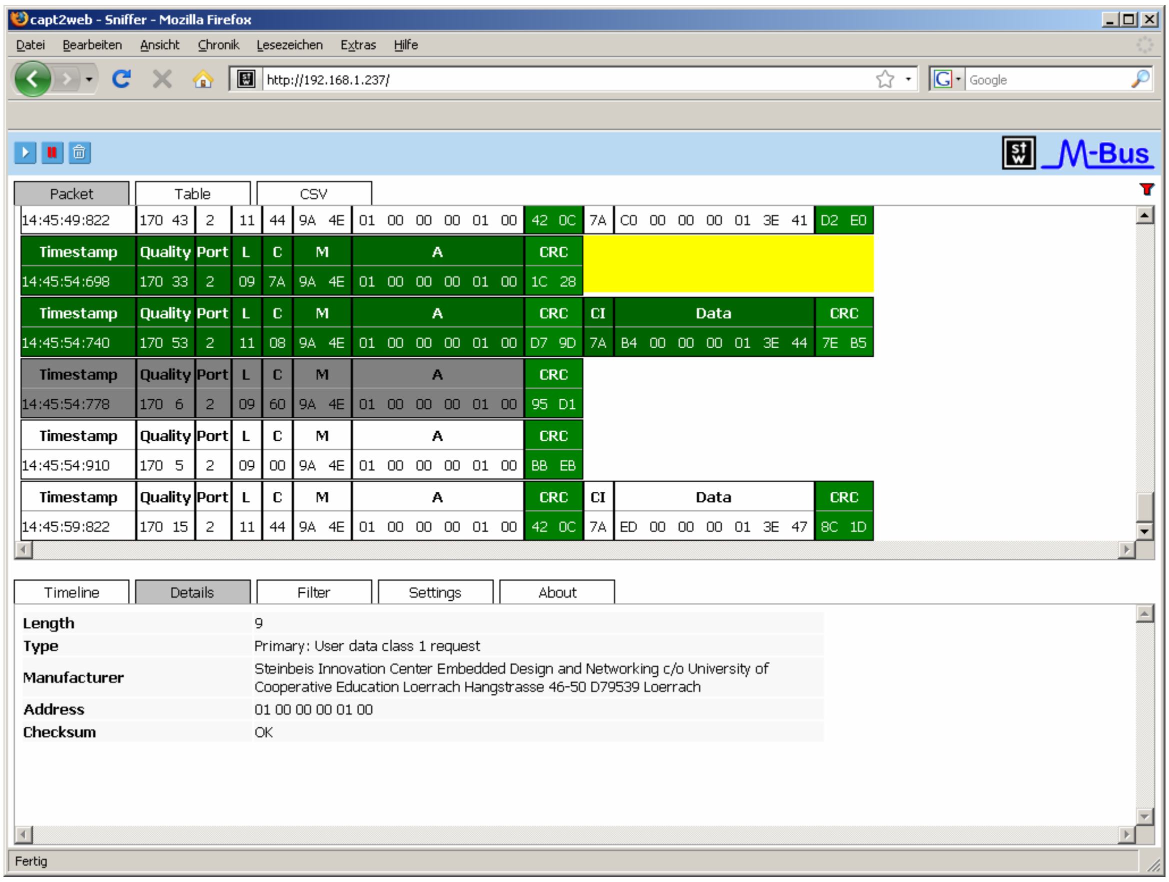
Port 1
Port 2
Device 9A 4E 01 00 00 00 01 00

Fig. 5 – Screenshot of the packet mode web-site for Wireless M-Bus protocol


capt2web - Sniffer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://192.168.1.237/
Google
M-Bus
Packet
Table
CSV
Timeline
Details
Filter
Settings
About
Port 2
Device 9A 4E 01 00 00 00 01 00
Fertig

Timestamp	Quality	Port	L	C	M	A	CRC	CI	Data	CRC
14:40:30:869	170 22	2	09	5A	9A 4E	01 00 00 00 01 00	ED 1D			
14:40:30:911	170 18	2	11	08	9A 4E	01 00 00 00 01 00	D7 9D	7A	98 00 00 00 01 3E 71	1A B
14:40:30:949	170 8	2	09	40	9A 4E	01 00 00 00 01 00	54 E4			
14:40:30:988	170 51	2	09	00	9A 4E	01 00 00 00 01 00	BB EB			
14:40:35:988	170 51	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	CB 00 00 00 01 3E 75	A0 F
14:40:41:300	170 0	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	29 00 00 00 01 3E 78	8D D
14:40:46:286	170 17	2	09	7A	9A 4E	01 00 00 00 01 00	1C 28			
14:40:46:328	170 61	2	11	08	9A 4E	01 00 00 00 01 00	D7 9D	7A	1F 00 00 00 01 3E 7B	54 B
14:40:46:366	170 7	2	09	60	9A 4E	01 00 00 00 01 00	95 D1			
14:40:46:405	170 14	2	09	00	9A 4E	01 00 00 00 01 00	BB EB			
14:40:52:355	170 13	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	54 00 00 00 01 3E 7F	8A B
14:40:56:717	170 9	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	B1 00 00 00 01 3E 82	CB D
14:41:01:554	170 41	2	09	5A	9A 4E	01 00 00 00 01 00	ED 1D			
14:41:01:596	170 45	2	11	08	9A 4E	01 00 00 00 01 00	D7 9D	7A	A6 00 00 00 01 3E 85	9A 1
14:41:01:634	170 13	2	09	40	9A 4E	01 00 00 00 01 00	54 E4			
14:41:01:674	170 9	2	09	00	9A 4E	01 00 00 00 01 00	BB EB			
14:41:06:977	170 14	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	D9 00 00 00 01 3E 89	D5 8
14:41:12:117	170 4	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	37 00 00 00 01 3E 8C	82 7

Fig. 6 – Screenshot of the table mode web-site for Wireless M-Bus protocol


capt2web - Sniffer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://192.168.1.237/
Google
M-Bus
Packet
Table
CSV
Timeline
Details
Filter
Settings
About
Fertig

14:45:49:822	170 43	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	C0 00 00 00 01 3E 41	D2 E0
Timestamp	Quality	Port	L	C	M	A	CRC			
14:45:54:698	170 33	2	09	7A	9A 4E	01 00 00 00 01 00	1C 28			
Timestamp	Quality	Port	L	C	M	A	CRC	CI	Data	CRC
14:45:54:740	170 53	2	11	08	9A 4E	01 00 00 00 01 00	D7 9D	7A	B4 00 00 00 01 3E 44	7E B5
Timestamp	Quality	Port	L	C	M	A	CRC			
14:45:54:778	170 6	2	09	60	9A 4E	01 00 00 00 01 00	95 D1			
Timestamp	Quality	Port	L	C	M	A	CRC			
14:45:54:910	170 5	2	09	00	9A 4E	01 00 00 00 01 00	BB EB			
Timestamp	Quality	Port	L	C	M	A	CRC	CI	Data	CRC
14:45:59:822	170 15	2	11	44	9A 4E	01 00 00 00 01 00	42 0C	7A	ED 00 00 00 01 3E 47	8C 1D

Length	9
Type	Primary: User data class 1 request
Manufacturer	Steinbeis Innovation Center Embedded Design and Networking c/o University of Cooperative Education Loerrach Hangstrasse 46-50 D79539 Loerrach
Address	01 00 00 00 01 00
Checksum	OK

Fig. 7 – Screenshot of the packet mode web-site, showing details using "mouse-over" for Wireless M-Bus protocol

5. ACKNOWLEDGEMENTS

The author thanks his team of Steinbeis Innovation Center Embedded Design and Networking [22] for the brilliant work around the capt²web hardware and software platform.

The author is also grateful for the support from BMWi within the DEMAX project.

6. REFERENCES

[1] "Pervasive Computing: Trends and Impacts", Federal Office for Information Security, ISBN 3-922746-76-4, 2006.

[2] M. Ringwald, K. Römer, "*Deployment of Sensor Networks: Problems and Passive Inspection*", Proc. of the 5th Workshop on Intelligent Solutions in Embedded Systems (WISES '07), Madrid, June 2007.

[3] http://www.elero.de/en.htm

[4] http://www.enocean.com/

[5] http://www.daintree.net/

[6] http://www.wireshark.org

[7] P. Havinga, S. Etalle, H. Karl, C. Petrioli, M. Zorzi, H. Kip, T. Lentsch, "*EYES - Energy Efficient Sensor Networks*", in: IFIP-TC6 8th International Conference on Personal Wireless Communications, (PWC), Sep. 2003, Venice.

[8] U. Kumar, A. Ranjan, V. Jalan, P. Mundra, P. Ranjan, "*CENSE : A modular sensor network testbed*", National Conference on Embedded Systems Feb 2006, Mumbai (India).

[9] E. Mackensen, W. Kuntz, C. Muller, "*Smart wireless autonomous microsystems (SWAMs) for sensor actuator networks*", Sensors for Industry Conference, Proc. ISA/IEEE, 2004.

[10] G. Werner-Allen, P. Swieskowski, M. Welsh, "*MoteLab: A Wireless Sensor Network Testbed*", in: Proc. 4th Int'l Conf. on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS), April 2005.

[11] J. Beutel, M. Dyer, L. Meier, L. Thiele, "Scalable Topology Control for Deployment-Sensor Networks", in Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05), IEEE, Piscataway, NJ, April, 2005.

[12] F. Schmidt, G. Scholl, A. Anders, H.-J. Körber, H. Wattar, „*RF-Embedding of Energy-Autonomous Sensors and Actuators into Wireless Sensor Networks*". In: *Multifunctional Structures / Integration of Sensors and Antennas*", Meeting Proceedings RTO-MP-AVT-141, Paper 3. Neuilly-sur-Seine, France.

[13] T. Gubisch, A. Sikora, "*New Developments for Wireless M-Bus*", Embedded World Conference 2009, Nuremberg, Germany.

[14] B. DeMuth, "*Designing Embedded Internet Devices*", Newnes, 2002.

[15] P. Brügger, A. Sikora, "*Using Embedded Web-servers in Industrial Applications*", embedded world 2005 Conference, Nuremberg.

[16] L. Möllendorf, „*Implementierung eines Sniffers für EnOcean Funknetze*", Practical Thesis, Dpmt. Information Technology, DHBW Lörrach, 2008.

[17] A. Newton, "*MooTools Essentials: The Official MooTools Reference for JavaScript™ and Ajax Development*", Apress, 2008.

[18] http://mootools.net/

[19] http://jquery.com/

[20] http://www.freescale.com/files/dsp/doc/prod_brief/ MCF5272PB.pdf?fsrch=1

[21] http://www.embetter.de

[22] http://www.stzedn.de

***Axel Sikora**, holds a diploma degree in electrical engineering and a diploma degree in business engineering, both from RWTH Aachen Technical University, Germany. His Ph.D. was in the field of digital circuit design at Fraunhofer Institute IMS, Duisburg, Germany.*

After several positions in telecommunications and semiconductor industry, he was appointed Professor at Baden-Württemberg Cooperative State University Lörrach in 1999. There, he is now heading the Information Technology department.

In 2002, he founded Steinbeis Innovation Center Embedded Design and Networking (sizedn), which concentrates on algorithm development, protocol development, protocol implementation, simulation and test in the field of wired and wireless networking.