Hochschule Offenburg
University of Applied Sciences

A Web-Based Monitoring Tool for Metering Bus (EN13757-3)

Thomas Matt, Manuel Schappacher, Axel Sikora

***Abstract*—The Metering Bus, also known as M-Bus, is a European standard EN13757-3 for reading out metering devices, like electricity, water, gas, or heat meters. Although real-life M-Bus networks can reach a significant size and complexity, only very simple protocol analyzers are available to observe and maintain such networks. In order to provide developers and installers with the ability to analyze the real bus signals easily, a web-based monitoring tool for the M-Bus has been designed and implemented. Combined with a physical bus interface it allows for measuring and recording the bus signals. For this at first a circuit has been developed, which transforms the voltage and current-modulated M-Bus signals to a voltage signal that can be read by a standard ADC and processed by an MCU. The bus signals and packets are displayed using a web server, which analyzes and classifies the frame fragments. As an additional feature an oscilloscope functionality is included in order to visualize the physical signal on the bus. This paper describes the development of the read-out circuit for the Wired M-Bus and the data recovery.**

***Index Terms*—Bus analyzer, signal recording, ADC, M-Bus.**

I. Introduction

The Metering Bus, also known as M-Bus, is a communication protocol used for various meters and data collectors, e.g. for metering of gas, heat, or electricity. It is defined in the European standard EN13757 [1]. The M-Bus is widely used, mainly for apartment buildings, but it is also deployed in the industrial sector. The standard has some issues because of unclear or missing definitions, which might result in inconsistencies during the setup of an M-Bus network. To reduce the problems, companies founded the working group 4 (WG4) [2] in the Open Metering Systems (OMS) group to revise the current standard. The Laboratory "Embedded Systems and Communication Electronics" (ESK) from Offenburg University of Applied Sciences is an active contributor to this group. To overcome existing problems, the authors developed an M-Bus bus analyzer which is shown in figure 1. The M-Bus bus analyzer is a tool that can help to create a properly-working M-Bus network. Therefore, the analyzer (also called sniffer) records the complete signal flow of an M-Bus network and monitors and analyzes the data flow by use of an integrated web server.

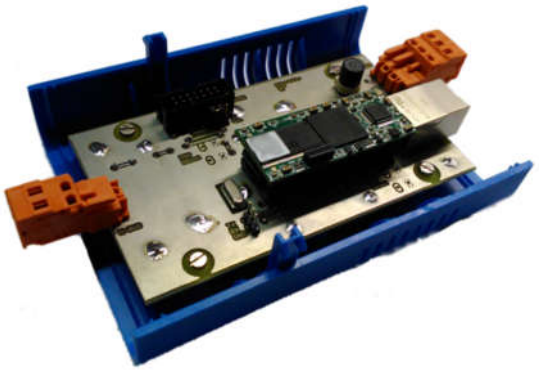

Figure 1: M-Bus protocol analyzer prototype.

The main platform for the analyzer is the Wireless M-Bus analyzer capt2web [3] which uses an ARM9 controller[4], running a Linux distribution and an embedded web server to illustrate the bus data. Since the ARM controller was originally developed as a head unit for a Wireless M-Bus RF module, one of the main tasks in this project was to replace the existing RF module by a read-out circuit for Wired M-Bus and to adapt the head unit's frontend to accommodate to the wired M-Bus protocol. Furthermore, the read-out circuit had to be developed, since no other solution had been available.

This paper is organized as follows: Chapter II explains the Physical (PHY) and Data Link Layer (DLL) of the M-Bus protocol. After this, chapter III discusses the possibilities to tap the signal with minimum influence on the measured system. Then, chapter IV describes the hardware architecture of the analyzer. The software functionality will be presented in chapter V. Finally, chapter VI describes the head unit, and chapter VII provides a summary and an outlook.

Thomas Matt, thomas.matt@hs-offenburg.de, Manuel Schappacher, manuel.schappacher@hs-offenburg.de, Axel Sikora, axel.sikora@hs-offenburg.de, are with Hochschule Offenburg, Badstraße 24, 77652 Offenburg.

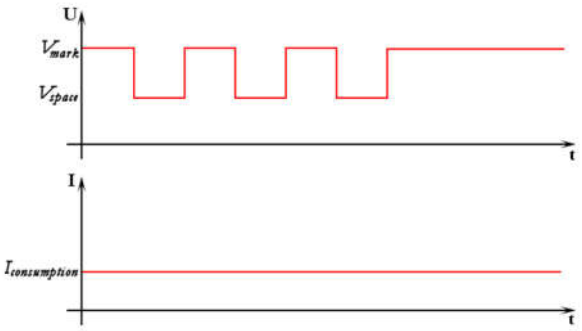

Figure 2: M-Bus physical levels during a master to slave transmission.

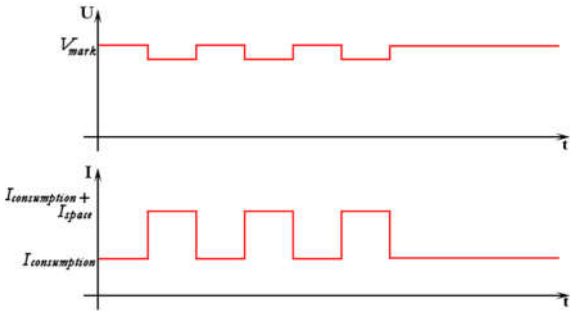

Figure 3: M-Bus physical levels during a Slave to Master transmission.

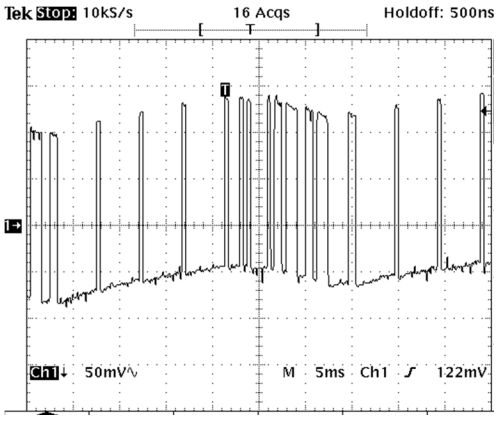

Figure 4: Real slave signals in a screenshot from an oscilloscope.

II. Metering Bus

A. General

The Metering Bus is based on a common UART interface that uses a fixed parameter set to transmit a byte. Therefore, the overhead is composed of a single start bit, a parity bit, and a stop bit. For the transmission speed, different baud rates in the range from 300 to 38,400 Baud can be used. However, baud rates above 9,600 Baud are neither recommended, nor widely used due to the characteristics of the physical layer.

The M-Bus does not provide a separate wire as bus clock, but uses self-clocking. To reduce the number of wires and to avoid collisions on the common bus, the M-Bus is based on a single master to control the data flow. Both send and receive operations can be done over a single pair of wires. In addition to the bus arbitration, the master provides the power supply for each slave, resulting in a constant voltage offset on the bus. Since this offset is not clearly specified in the standard, the voltage can be in the range of several volts, while typical values range is from 24 V to 50 V. The M-Bus provides both the downlink communication from master to slave (M2S) and the uplink communication from slave to master (S2M).

B. Master to Slave Communication (M2S)

For the communication, a logical one is represented by a fixed offset voltage V_{mark} and a logical zero by an offset voltage V_{space} of -12 V [1]. Therefore, the communication is defined more or less by voltage drops of 12 V [1]. The current $I_{consumption}$ is constant during the transmission, as shown in figure 2.

C. Slave to Master Communication (S2M)

For the communication, the slave modulates its own current consumption. In this case, the current will not be constant anymore and the master that supplies the power to the slaves can decode the communicated data through the total current consumption on the bus.

In its idle state, a slave consumes a constant current of 1.5 mA [1], which is also known as unitload or I_{mark}. Depending on the current consumption, a device can consume a single or integer multiples of a unitload. For example, if a device needs 5 mA to run, it will burden the bus with four unitloads (6 mA). In the S2M direction, I_{mark} represents a logical one sent by a slave. The maximal possible unitload within an M-Bus network depends on the master providing the power supply. Typically, a master can provide up to 250 unit loads. The sum of all unitloads is called $I_{consumption}$.

The logical zero or I_{space} is represented by a current consumption in the range between 11 mA and 20 mA [1]. The modulation of the current consumption also has an effect on the offset voltage of the M-Bus master. As shown in figure 3, a slave's logical zero will always cause a voltage drop on the bus. In order to verify that, a part of this project was to record the signals on the transmission lines. The oscillogram in figure 4 shows a record of an S2M communication. It can be seen that the current transmission implies a small voltage swing on the bus according to theory. This swing depends on the internal resistance of the master and on the resistance of the transmission line. Since the internal resistance of a master is not unambiguously specified in the standard, it cannot be predicted. Typical values are between 1 Ω and 68 Ω. These values result in a voltage

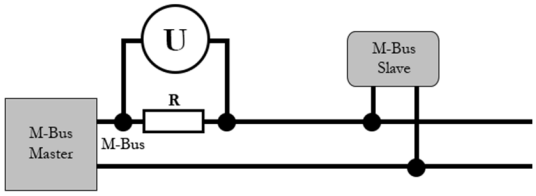

Figure 5: Signal tapping over a shunt.

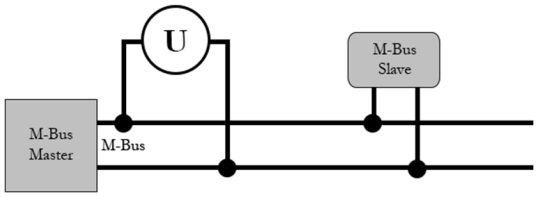

Figure 6: Signal tapping parallel to a slave.

swing at the master clamp between 7 mV_{pp} and 1.1 V_{pp}. Figure 3 shows an example of some slave signals taken from a test application given as reference. The voltage swing in this case is $V_{pp} \approx 200\,mV$. For these measurements, the oscilloscope must be set to AC mode, because the offset voltage is about 200 times higher than the signal.

III. Signal Tapping and Acquisition

A bus or protocol analyzer needs to interpret the electrical signals of the bus. In the case of the M-Bus, this becomes even more difficult because of the two different communication methods. Existing M-Bus sniffers only interpret parts of the physical signals, since they are designed as master devices without the abstraction of the application and data link layer. Furthermore, they only consider S2M direction, but not the M2S, since this communication is always originated by the master itself. This makes the approach unsuitable for a real analyzer.

For the M2S direction, various compatible driver ICs are available as commercial-off-the-shelf (COTS) devices, e.g. TSS721a [5] from Texas Instruments. However, it is not possible to use such a transceiver for the planned analyzer, as it would increase the current consumption of the bus and therefore would affect the bus itself. Since this is an unwanted behavior for an analyzer, the usage of such components should be avoided. In addition, there is also a circuit needed for the S2M direction, because the previous described solution cannot be used. The project goal was to create an analyzer and not a bus master. To be able to analyze both directions different methods of the signal acquisition have been investigated. However, since there were no suitable methods or existing sniffers available, an own approach had to be developed.

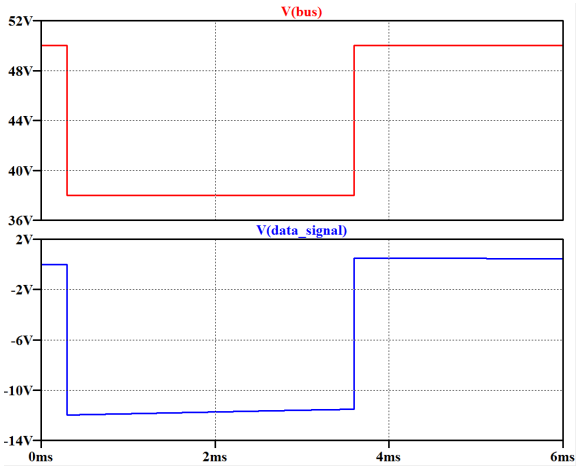

Figure 7: DC cancelation using a passive 1st order high pass filter with $f_c = 1$ Hz at 300 Baud.

A. Current Methods

The S2M direction uses the current modulation for data transmission, allowing these signals to be grabbed with a current clamp or a shunt. The shunt method is shown in figure 5. Both alternatives can only be used for the S2M communication, because in M2S direction, the bus current is constant. Also a problem of this method is that the total S2M communication can only be regained at the master's clamp, because of Kirchhoff's current law. Each slave creates a new junction in the network, and so the sum of all currents can only be measured at the master's clamp.

B. Voltage Methods

The simplest way to retrieve M2S and S2M signals with a single circuit is to use the bus voltage as shown in figure 6, since both directions have effects on the bus voltage. However, the offset voltage of the M-Bus remains an unknown parameter, as it depends on the implementation of the devices. Since also the offset itself does not have any informational content, the basic idea of this approach is to separate the data signals from the offset.

C. DC Cancelation

The UART interface uses rectangular pulses to transmit the data, meaning the data has a direct current (DC) part as well. However, if an offset blocker is used to split up the data signal from the bus signal, it will have some impact on the frequency characteristics of the data signal, i.e. it will lead to a distortion of the signal. This is because of the system function of the offset blocker. The data signal will be convolved with this function, but the influence of the blocker will be marginal if the cutoff frequency is much smaller than the transmission frequency. Figure 7 shows what a data signal will look like after a passive 1st order high pass. The

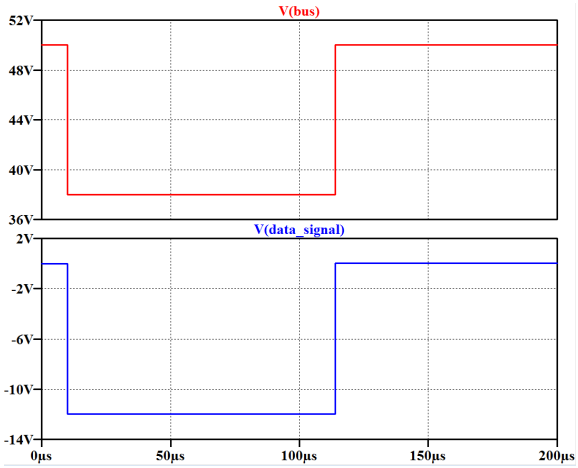

Figure 8: DC cancelation using a passive 1st order high pass filter with $f_c = 1$ Hz at 9,600 Baud.

cutoff frequency is $f_c = 1$ Hz, and the simulated baud rate is 300 Baud, which is also the lowest data rate for the sniffer. The upper graph shows the input signal, the lower the output signal of the high pass filter. As can be seen, even though the levels might drop slightly over time, the edges are clear enough to perform the required signal analysis. With higher baud rates, e.g. 9,600 Baud, these effects will no longer be recognizable as can be seen in figure 8. Higher baud rates do have frequencies around the baseband like lower baud rates, but the amplitudes are not this high. So the waveform of the higher baud rates will be less affected than the waveforms at lower baud rates. This method is also used in the AC mode in oscilloscopes.

To remove the voltage offset, the M-Bus analyzer uses such a 1st order high pass filter. After the offset is removed, the data signal will be amplified to get a larger voltage swing from the S2M direction. Furthermore, a specified offset will be added to achieve positive voltage levels. This signal will be captured by a microcontroller (MCU) using an ADC, and finally, the signal analysis will be performed.

The M-Bus does not specify which of the two wires of the cable is used as ground and which one carries the signal. This fact requires the sniffer to allow a voltage inversion at the bus input. In case there is no function or no component to invert the voltage, or if the device is not connected correctly, the phase of the data signal will be inverted, and the data cannot be analyzed correctly. A rectifier would represent the default solution here, but because of its transverse current it would also increase the influence on the bus. Therefore, the ADC input can simply be inverted to achieve the same effect. When the sniffer is plugged in with an incorrect phase, the configuration can be changed to fix this problem. This phase inversion is a very simple process where only the thresholds of the program and the sample of the ACD will be inverted. Thereafter, the whole program will run properly.

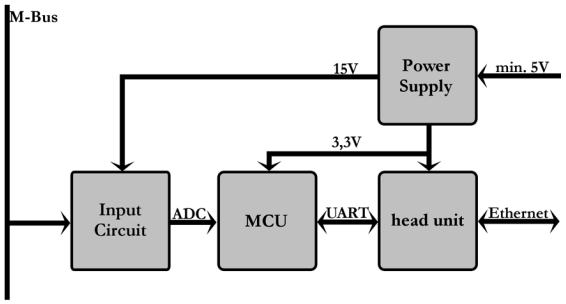

Figure 9: Architecture of the protocol analyzer.

Table 1: *CSPB* at 9,600 Baud.

	OSR 32	*OSR* 64	*OSR* 128
CLK	39	19	9
CLK/2	19	9	4
CLK/4	9	4	2
CLK/8	4	2	1

IV. Architecture

A. General Architecture

Figure 9 shows the protocol analyzer with its four functional blocks. The analyzer must be supplied with at least 5 V, with a maximum current of 500 mA. The power supply transforms the input voltage into two different output voltages.

First, the input circuit requires a voltage of 15 V to accommodate the M2S voltage swing. This swing is defined by a minimum peak to peak voltage of 12 V, but it is also possible to have a higher swing. To prevent the limits of the circuit to be reached, it works with 15 V. The rest of the circuit only needs a voltage of 3.3 V. The remaining blocks are mainly responsible for the actual signal capturing and analysis.

B. Analog to Digital Conversion

The selected MCU is a derivate of the Texas Instruments MSP430 family including a delta-sigma converter (DSC). An advantage is the high resolution, compared to an ADC with a successive approximation register (SAR) conversion method. The high resolution is needed to recognize the S2M signal without an additional amplifier. A disadvantage of a DSC is the long conversion time, compared to a SAR ADC. In theory, there is only one sample per bit needed to recognize the logical level. However, if the signals are too short, or if there is no separate synchronization signal available, single bits might be missing. Since S2M communication implies small signals on the M-Bus an asynchronous interface is used.

For a single bit conversion, there are at least two or more samples needed. To fulfil this requirement, the

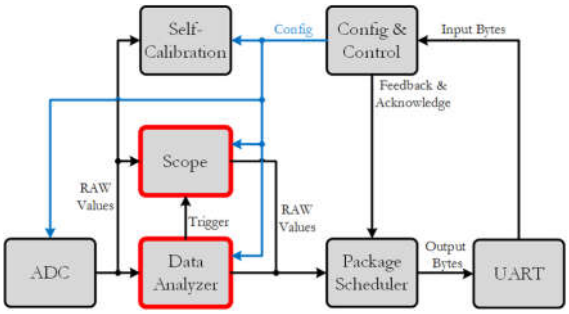

Figure 10: Software block diagram.

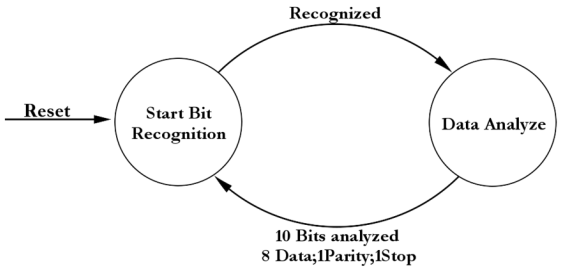

Figure 11: Finite state machine for data recognition.

conversion time of the ADC must match the configured baud rate. Therefore, a conversion table, as shown in table 1, can be used, which shows the count of samples per bit *CSPB* for a baud rate of 9,600. The conversion time depends on the selected clock frequency CLK_{max} and the oversampling rate *OSR*. Both parameters are relevant to calculate the *CSPB*, as shown in equation 1.

$$CSPB = \left\lfloor \frac{t_{bit} \cdot CLK}{OSR} \right\rfloor \quad (1)$$

The parameter t_{bit} describes the time, which is needed to transmit one single bit, e.g. t_{bit} = 104 µs for 9,600 Baud. The selected configuration for 9,600 Baud is *CLK* = 3 MHz and *OSR* = 64, thus *CSPB* = 4. The baud rates, which can be recorded, are between 300 and 9,600 Baud; otherwise, *CSPB* is too low to recognize a bit properly. This limitation at 9,600 Baud gives enough points to realize a scope function. The algorithms were designed with Matlab [6]. The microcontroller sends the ADC samples via its UART interface to a workstation. Thus, the algorithm could be written in Matlab and later be implemented in the controller source code.

V. Software

The overall architecture of the MCU software is shown in figure 10, including its two main functionalities, the data analysis and the scope function. Furthermore, the different software parts can be controlled and configured via the UART interface using a packet-based communication protocol.

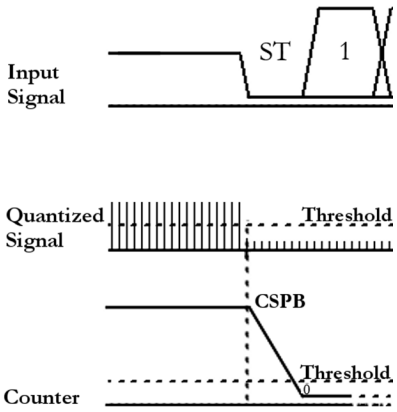

Figure 12: Recognition of the start bit.

In addition, there is also a mode for self-calibration. A self-calibration is required since the signal analysis uses level thresholds to regain the information. The values of the calibration are stored at the flash segment of the MCU, so a recalibration is only needed if there are significant errors within the conversion. The calibrated values are automatically reloaded during the initialization of the controller. The footprint of the software currently requires 5.6 kB for the code segment in flash and 860 Bytes of RAM memory.

A. Signal Analysis

According to the UART specification, an incoming byte always begins with a start bit. However, since the interface is asynchronous, the timing of the start bit is unknown. Therefore, the internal timing needs to be synchronized to the bus transmission timing. If the timing is not synchronized, the following data bits will not be analyzed correctly and bit errors might occur. To avoid this, the data recognition state machine, as shown in figure 11, consists of two different states to analyze the data. After a reset, the software assumes that the next incoming bit is a start bit. If start bit comes in, the state machine will switch to the data analysis state.

1) Start Bit Recognition

The recognition of the start bit uses the time discrete functionality of an ADC. Each ADC sample will be passed to a state machine that uses an averaging method and thresholds to detect the start bit. Once the start bit is detected, the main state machine switches to the data analyze function which is described in the following chapter.

Averaging multiple values is very simple from a mathematical point of view. All values need to be summed up and divided by the number of values. Some controllers, like the used MSP430, do not have special hardware units for divisions. This will cause a timing problem, since a division needs many more machine cycles in that case. To avoid this, the state machine uses

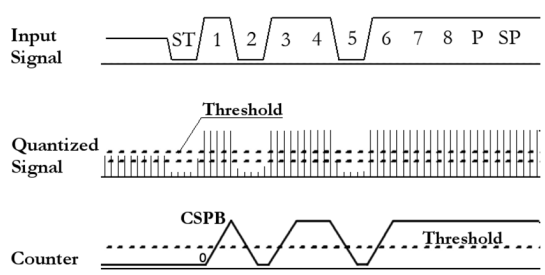

Figure 13: Signal analysis via ADC.

a counter mechanism. If the sample becomes lower than a predefined threshold, the counter will be decremented; if the detected value is higher, the counter will be incremented. The maximum value of the counter is equal to the CSPB. For example, in case the ADC uses eight samples per bit, the upper limit for the counter will be eight as well. The lower limit will always be zero. This method needs only a few machine cycles. Figure 12 shows the steps of how to recognize the start bit. The incoming signal is quantized and the counter's value is changed accordingly. In case the value of the counter runs below the threshold of the counter, the start bit is detected and the data analysis is triggered.

2) Data Analysis

In case of a successful start bit recognition, the following ADC values are understood as data bits. Like the start bit recognition, the data analysis also uses a state machine, which is driven by the ADC values. The main difference compared to the start bit recognition is that the state machine decides for the value of the recorded bit after each *CSPB* instead of after each sample.

The data analysis state machine uses three thresholds instead of two, if it is compared to the start bit recognition state machine. Two of the three thresholds will decide, how a counter value will changed, depending on the current ADC sample. The third threshold is to regain the logical information from the counter value. Figure 13 shows an example of the transmitted byte 0xED and how the data will be analyzed with this algorithm.

An additional functionality of the data analysis state machine is the parity filter, allowing the parity to be checked before the data is copied into the input buffer. If the parity calculation is not equal to the transmitted parity bit, the byte will not be discarded. Since parity errors are an important indicator for the bus analysis, this function can be enabled or disabled by the user.

B. Scope Function

The scope function is another important feature of the sniffer. Recording of the data by use of an ADC allows for sending the samples directly to the head unit. The head unit can then use these samples to draw a time and

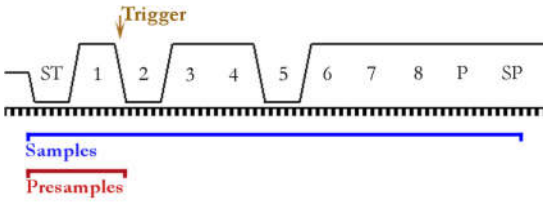

Figure 14: Example for scope record.

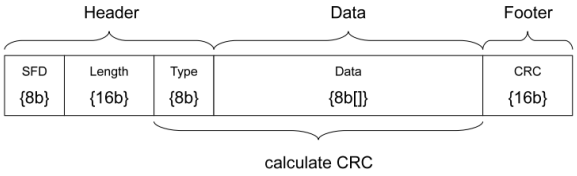

Figure 15: Packet design [7].

voltage discrete graph of signal edges and voltage levels on the bus. For the scope function common oscilloscope controls have been adopted. The scope has the ability to trigger at the end of each bit of the communication, as well as to operate in a free-running mode. A free running-mode prompts the data without waiting for any trigger. When using triggers, the scope offers two run modes. The first run mode is called single shot and will only record one interval and then freeze. The second mode is the continuous mode, where the record will be retriggered until the user stops the recording.

The scope function also supports presamples. For example, when using triggers, it would not be possible to see the start bit since the record can only be triggered after the bit was recognized. A memory or presample functionality helps in this situation. Figure 14 shows an example where the record is triggered by a first data bit. The presample functionality is implemented with a ring buffer using two pointers and a size of the ringbuffer equal to the maximum sample size as indicated in figure 14. The two pointers are used to get access to the buffer. One pointer is used to store the data and the other to pick the data. The position difference of both pointers is used for the presamples. If there are no presamples selected, both pointers are pointing to the same data segment, otherwise the pick pointer runs after the store pointer. The maximum possible number of presamples is the number of samples – 1.

The maximum possible sample frequency of the scope is limited by the communication interface between head unit and driver and the size of the ADC sample. Because of the 16 bit ADC resolution, one sample has the size of two bytes. The interface between the head unit and the driver is a UART communication with 115200 Baud. The maximum sample frequency can be estimated by use of equation 2.

$$f_{sample} < \frac{Baudrate}{n_{ADCBytes}} \qquad (2)$$

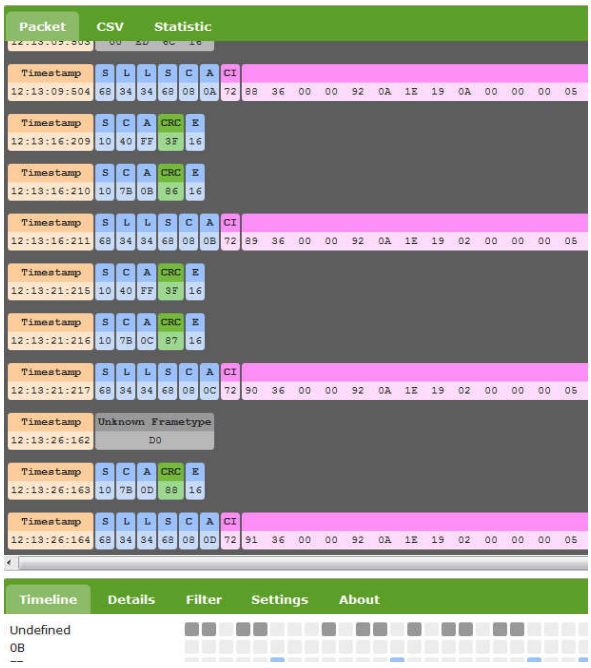

Figure 16: capt2web web interface.

Taking into account the data of table 1, the maximum sampling frequency is limited to 46 kHz.

C. Head Unit Interface

The main platform for the analyzer is the Wireless M-Bus analyzer capt2web [3] that uses a packet-based protocol to communicate with a driver module. This protocol had to be implemented to finally integrate the wired M-Bus driver into the existing sniffer backend. As mentioned, figure 15 shows the structure of such a packet. The packets begin with a start field with a constant value, which is called start frame delimiter (SFD). Each packet has a length field describing the size of the following data field. The content of the data field is described by the type field at the beginning of the data field. At the end of each package, a cyclic redundancy check (CRC) over the complete data field is appended. For this, the calculated CRC has to be updated immediately while the ADC continues its sampling to save time.

The MCU uses an array of packets to send the data. This is required because a packet cannot be filled and transmitted at the same time. If there is only one packet to fill and send, there would be some access problems if some new data comes in. The handling of these packages will be done by a scheduler. So the data analysis or the scope function will just pass the data to the scheduler, which handles the rest.

VI. Head Unit

The head unit uses a web server to process the recorded data, which can be displayed with any ordinary web browser like firefox or internet explorer. The usage of a web server as frontend increases the flexibility of the analyzer since it is possible to use any operation system with a web browser and there is no need to install additional tools. The languages which are used for the interface are Php and Javascript. The recorded data is stored in an SQL database.

The information is separated into the frame types of the M-Bus [1], which increases the comprehension of the data. The data is normally sent with one of the two M-Bus frames. These frames will be dissected into its fields. Also, the CRC will be checked and highlighted. Figure 16 shows a screenshot from the capt2web interface with some packages. Each information is displayed with a time stamp which is always at the beginning. The time stamp is generated by the head unit and shows the time when the packet with the information was received by the head unit. The parts of the data link layer are displayed in blue and the application layer in pink. If a data segment is not a part of a frame, the data will be in grey. The data can also be exported from the web interface; for this, there is a csv-export page available where the complete data displayed is available as semicolon separated textual fields. A future feature of the interface will be the implementation of the application layer. So, e.g., the values or mean values of meters will be shown.

VII. Summary and Outlook

This project presents a bus analyzer for the M-Bus, which is a big step forward for finding network problems and errors since the detection of errors has always been a big problem until now. The analyzer is a flexible, easy, and low-cost tool. Furthermore, it is designed with a low-cost input circuit, which allows the user to connect the analyzer to any point of the M-Bus. The information itself will be regained by the software and the analog output value of the input circuit. Through this ability, it will be easy to implement other UART-based protocols into the analyzer, e.g. RS232.

The advantage of the web-based interface is that it does not depend on a specific operation system or additional software tools, which increases the flexibility. Also the Ethernet interface of the head unit improves the flexibility, since the analyzer runs as a stand-alone unit with access over a local area network. All plugs and jacks of the sniffer are removable, so it is easy to swap the tool to another test circuit.

The future will bring some interesting features to the user. For example, the data will not only be presented as raw data, but the application layer data will be decomposed. This will allow the user to read everything in normal textual form without a second tool. Another future option is to unite the wireless and wired sniffer into one single unit to obtain a one-and-all solution for the M-Bus.

REFERENCES

[1] M-Bus Documentation Rev 4.8, http://www.m-bus.com/, 11.09.2015.
[2] Open Metering Systems work group 4, http://oms-group.org/en/oms-group/working-groups/, 11.09.2015.
[3] SSV-Teleservice Gateway, http://www.ssv-comm.de/produkte/mgw865.php, 11.09.2015.
[4] SSV-DNP/9265, http://www.dilnetpc.com/dnp0096.htm, 11.09.2015.
[5] M-Bus Slave Transceiver, Texas Instruments TSS721a, http://www.ti.com/lit/ds/symlink/tss721a.pdf, 11.09.2015.
[6] Matlab Homepage, http://de.mathworks.com/products/matlab/, 11.09.2015.
[7] Serial Reference Manual, Steinbeis Transfer Center, page 21, 2014.

Thomas Matt studied Electrical and Information Engineering (EI) at the University of Applied Science Offenburg and received his B.Eng. degree in February 2015. Since that time he has been working as an engineer for the lab “Embedded Systems and Communication Electronics” at University of Applied Sciences Offenburg.

Manuel Schappacher studied Computer Engineering at the University of Applied Sciences, Furtwangen and received his Dipl.-Inform. degree in April 2009. After that, he continued to work as project engineer at Steinbeis Innovation Center Embedded Design and Networking (sizedn) mainly in the field of embedded wireless and wired communication, including simulation of networking protocols. Since 2014 he is with the laboratory of Embedded Systems and Communication Electronics at University of Applied Sciences Offenburg.

Prof. Dr.-Ing. Axel Sikora holds a diploma of Electrical Engineering and a diploma of Business Ad-ministration, both from Aachen Technical University. He has done a Ph.D. in Electrical Engineering at the Fraunhofer Institute of Microelectronics Circuits and Systems, Duisburg, with a thesis on SOI-technologies. After various positions in the telecommunications and semiconductor industry, he became a professor at the Baden-Wuerttemberg Cooperative State University Loerrach in 1999. In 2011, he joined Offenburg University of Applied Sciences, where he holds the professorship of Embedded Systems and Communication Electronics. His major interest is in the field of efficient, energy-aware, autonomous, and value-added algorithms and protocols for wired and wireless embedded communication. Dr. Sikora is author, co-author, editor and co-editor of several textbooks and numerous papers in the field of embedded design and wireless and wired networking, and head and member of numerous steering and program committees of international scientific conferences.