FACHHOCHSCHULE
AMBERG - WEIDEN

Automotive Applications via Wide Area Access

Dipl.-Ing.(FH)Heike Lepke
Prof. Dr.-Ing. Josef Pösl

University of Applied Sciences Amberg-Weiden

Kaiser-Wilhelm-Ring 23
92224 Amberg

h.lepke@fh-amberg-weiden.de

January 11, 2008

Abstract

The University of applied sciences FH Amberg-Weiden developed a user interface, which can access electronic control units (ECU), integrated in vehicles via a common internet browser. The connection is set up e.g. through the area-wide communication-service GPRS. In use of a standard GPRS-modem, which is joined to the ECU, and an integrated TCP/IP-protocol (lwIP - Lightweight TCP/IP stack), we provide an internet connection to driving vehicles. The interface is realised by a embedded HTTP web server. Every usual PC, with internet access can be connected to the ECU for data exchange. The further research is focused on the practicability of a wide area connection for our applications. These are e.g. a flash loader through CAN. The flash process of a new software for the ECU is initialised through the web server. We transmit the data via a web interface to the ECU, where we verify it. If the vehicle is in safe mode (e.g. not driving), the software can be transmitted to the flash and the system can the be restarted. We regard problems like the answer times of the server with different speeds of the vehicle, connection loss and transfer rates. This project is integrated in the joint project AUTOSAFE. It aims at improving road safety through an integral safety system.

1 Motivation

In automotive electronic, an ECU is an embedded system that controls subsystems in a vehicle. Such a subsystem is for example the engine control unit, which is amongst other things responsible for the quantity of fuel injected into each cylinder. In modern cars there are up to 70 ECUs which are connected by several bus systems. A commonly used bus system is the asynchronous CAN bus (Controller Area Network) [1]. The CAN bus handles the communication between the ECUs. For the active avoidance of traffic accidents, we need many applications, which are e.g. able to detect obstacles on the road. A further application has to guarantee that the vehicle does not leave the road by a lane detection system. Therefore we use many different sensors like MRR (mid range radar), LRR (large range radars), LIDAR (laser imaging and ranging) or cameras, as shown in figure 1. These sensors are connected to several ECUs. The sensors supply an ECU with sensor data, which is processed on the ECU. If necessary, e.g. an obstacle is detected, the ECU will forward messages to other ECUs, which are responsible for the behaviour of the car, for instance an alarm is raised to inform the driver.

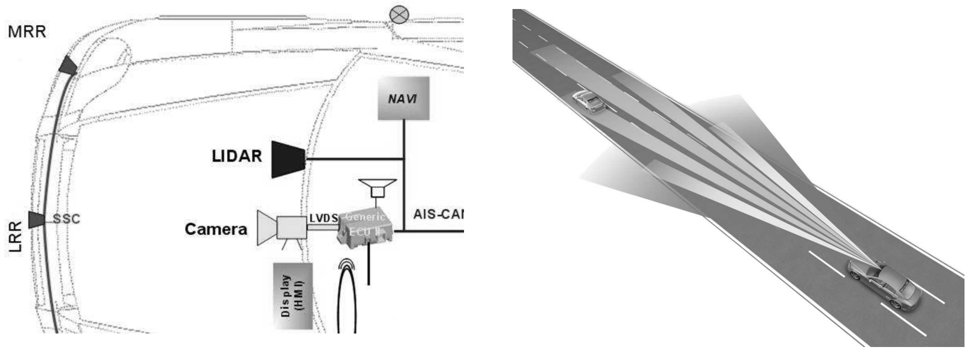

Figure 1: Sensors connected to an ECU. The sensors deliver data e.g. for obstacle detection

During the development of new safety application for a vehicle the accumulated data, delivered by the sensors during a test run, has to be verified. We often have to save parts of these data for an additional analysing. If the vehicle is on a longer test drive, the size of the required test data exceeds the amount of the buffer memory on the ECU, which leads to data loss. In addition we sometimes recognize during a test run that a sensor delivers insufficient data and requires new settings. We have to change these settings via CAN messages from the ECU to the sensor in order to continue the test. In many cases it is better to visualize test data immediately during a test run or to retrace the events produced by the application in specific moments. If the analysing is done after a test run, it may be impossible to reconstruct the reactions of the system, e.g. why a lane on the road was not detected. If the test data is not accurate evaluated by an application, the software of this application has to be changed. After an upload of the improved software to the ECU, the ECU has to be rebooted for a new test run.

Actually, to transfer data from a vehicle to a stationary PC, ECUs must be connected to a station and maybe dismantled. This additional expense disturbs processing or makes a research during the operation of a vehicle impossible. The solution to this problem is a wireless data transfer, where we have to account for great distances during a test run. We aim at a connection between a fixed station to the remote ECU in the vehicle. For this purposes we searched for a simple method for data transmission.

2 Wireless Access to Vehicles

In order to connect to several vehicles, every car has to be equipped with an access module, which enables the exchange of data from a backend control system (BCS) to the vehicle and vice versa. The access to the vehicle has to be independent of the vehicle location and status (e.g. standing or driving). Therefore the access module has to be wireless and has to provide a broad reach. Through such a wireless module, the BCS is able to access an integrated Web ECU and is able to activate several procedures on the Web ECU.

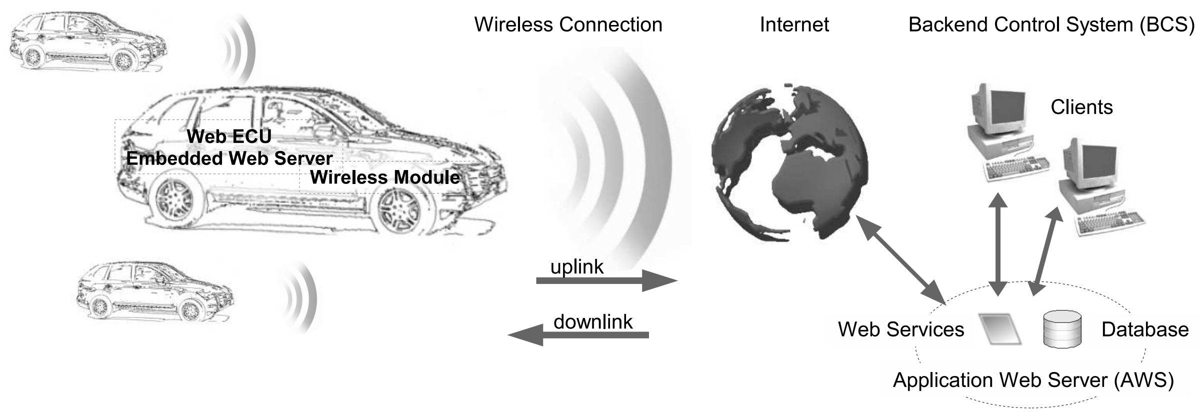

Figure 2: Wireless Access to Vehicles.

2.1 Backend Control System (BCS)

The BCS is made up of one or more clients (stationary PCs) and an application web server (AWS). The user interface to the vehicle can be accessed through a standard web browser (e.g. Internet Explorer, Firefox) on a client. A client can use this interface from every standard workstation, which has access to the internet. This has the advantage, that the system can be controlled by one or multiple PCs at the same time, without the need of a specific software installation and independent of the PC locations.

The AWS manages the connection to the vehicles. Every vehicle has an unique identifier, which is related to the IP address of the embedded wireless module. Therefore, the vehicle stores its current IP address in the database of the AWS. Furthermore, the AWS database is used to store vehicle specific data persistently. The AWS offers web services to process the stored data, e.g. to show the location of the vehicles on a map displayed in a web browser. Finally, the data can be downloaded to a client for further processing.

2.2 Wireless Connection

Because the connection between BCS and driving vehicle has to be independent of the location of the car, new connection technologies like WiMAX are not suitable. They still have a limited reach, because they have no large area coverage. The reach of WLAN or Bluetooth is not sufficient for our applications. For those reasons we regard the three well-known technologies GPRS, EDGE and UMTS, which provides us with full mobility.

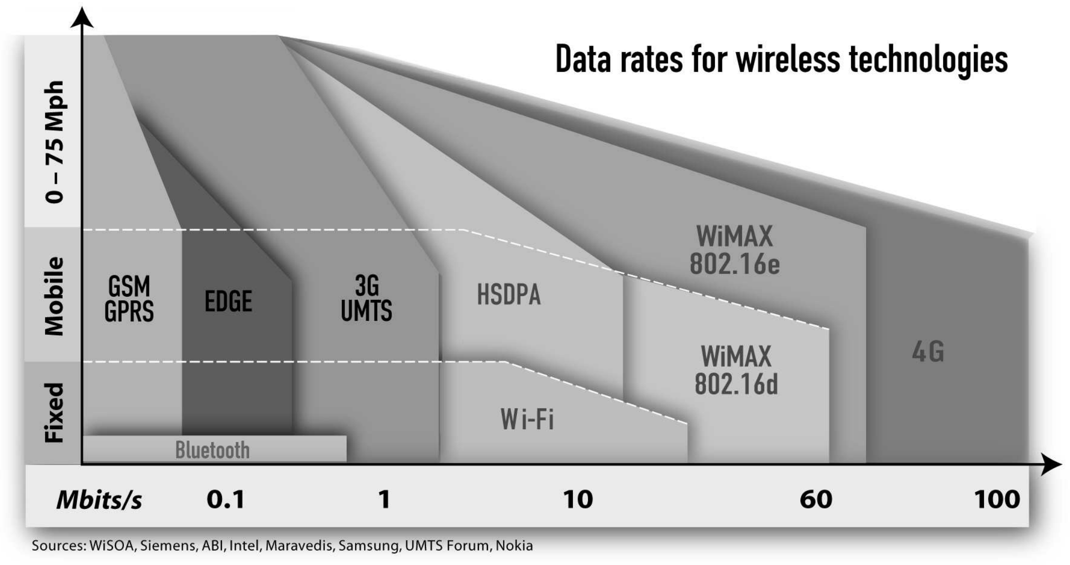

Figure 3: Data rates for wireless technologies [2]

Multislot Class	Downlink Slots	Uplink Slots	Active Slots
...	...	...	...
9	3	2	5
10	4	2	5
11	4	3	5
12	4	4	5
...	...	...	...

Table 1: GPRS Multislot Classes

Coding scheme	Speed (kbps/slot)
CS-1	9,05
CS-2	13,4
CS-3	15,6
CS-4	21,4

Table 2: GPRS Coding Schemes

2.2.1 GPRS (General Packet Radio Service)

GPRS has a coverage similar to mobile phones, that means a vehicle can be reached nearly everywhere in the world, in cities and in rural areas. On the other hand, GPRS has a low transfer rate in comparison with other technologies, but it is adequate for the most terms of our applications. The transfer rate depends on the chosen GPRS multislot classes [3] and coding schemes. The multislot classes are defined by the maximum capability of the mobile device, e.g. a standard GPRS modem has nowadays the multislot class 10 or 12. The GPRS speed is a direct function of the number of assigned time slots, divided in downlink and uplink slots, as shown in table 1. The coding scheme depends on the quality of the reception of the GPRS signals. The fastest coding scheme CS-4 is available near a base transceiver station and provides a speed of 20 kbps per time slot, as show in table 2. The coding scheme is adapted automatically depending on the mobile location.

The theoretical maximum value of the transfer rate is 171,2 kbps if 8 timeslots are clustered, but practically the transfer rate is limited by the provider. The transfer rate diversifies between 36,2 kbps and 85,6 kbps for a class 12 modem in down- and uplink.

2.2.2 EDGE (Enhanced Data Rates for GSM Evolution)

EDGE uses an improved modulation method, which allows a higher transfer rate for the GPRS network. Several timeslots can be combined for one connection resulting in the theoretical maximum of 473,6 kbps for eight timeslots. True values are a transmission speed from 150 kbps up to 200 kbps. By moving the receiver the transmission slows down noticeable and falls back at least to GPRS.

Coding and modulation scheme (MCS)	Speed (kbps/slot)
MCS-1	8.80
MCS-2	11.2
MCS-3	14.8
MCS-4	17.6
MCS-5	22.4
MCS-6	29.6
MCS-7	44.8
MCS-8	54.4
MCS-9	59.2

Table 3: EDGE Coding Schemes

2.2.3 UMTS (Universal Mobile Telecommunications System)

Because of the significant high transmission rates, the UMTS [4] standard is appropriated to transfer large amounts of data between the client and the vehicle. The new mobile protocol HSDPA (High Speed Downlink Packet Access), used for UMTS, supports theoretically a downlink speed up to 14,4 Mbit/s. However, mobile devices suitable for the connection to an embedded system in a vehicle reach 384 kbps for UMTS in up- and downlink. If the HSDPA-protocol can be used, the transmission rate increases to 3.6 Mbps in downlink and 384 kbps in uplink. However the connection falls back to GPRS respectively EDGE if the receiver moves to rural areas.

2.3 Web ECU

The Web ECU, shown in figure 4, is responsible for receiving and sending data via the internet. The received requests are processed on the Web ECU. It reacts to this requests, for example shows the selectable remote debug applications, available on the Web ECU, or starts an appropriate debug application.

The main task of the Web ECU is to assure the safety of cars, by safety applications as for e.g. obstacle or lane detection. The web functionality is integrated in an existing safety ECU. The ECU works with a 32 Bit processor MPC5200 [6] and 8 MB SDRAM. The processor is developed for applications of the automotive market, which need a fast data throughput like telematics, GPS (Global Positioning System) or image processing. It supports several interfaces i.a. Ethernet, Serial, SSC (Synchronous Serial Channel) and CAN. The operating system OSEK-OS [5] is developed for the requirements of automotive systems. A special attribute of OSEK-OS is the static memory management; no dynamic memory can be allocated during a process run. We have to consider that the size of memory an computing power in embedded systems is not available in such a large quantity as on a PC. In order to adapt the presented system to smaller platforms, the size of used memory for the system modules can be varied.

2.3.1 lwIP (lightweight TCP/IP stack)

The lwIP-module developed by Adam Dunkels [7] contains a TCP/IP stack and provides several different protocols. This protocols are necessary for the connection of a wireless module to the

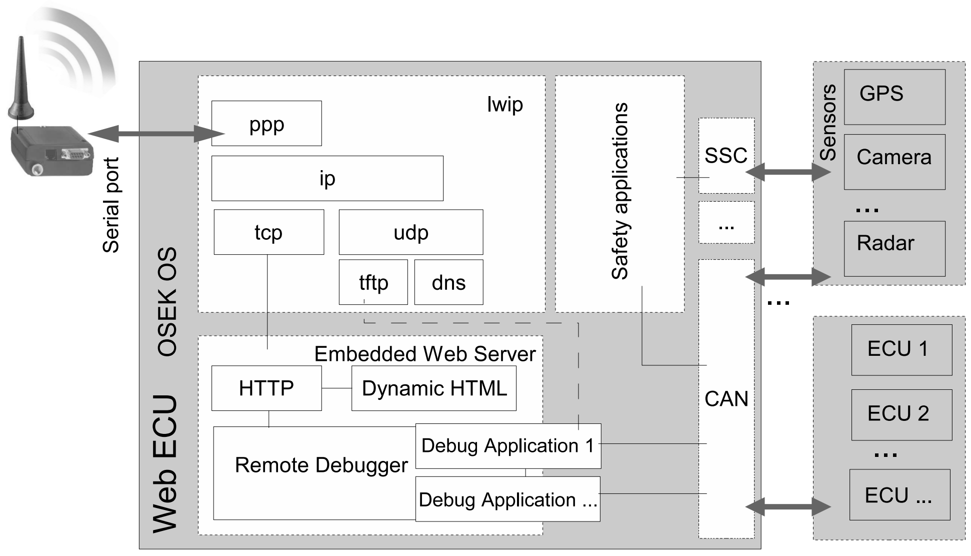

Figure 4: Embedded Web ECU

internet. The Point-to-Point-protocol (PPP) sets up a connection to the provider and receives the data through the wireless module via the serial port of the Web ECU. The data is passed through TCP/IP (Transmission Control Protocol/Internet Protocol) to the embedded web server. TCP [9] offers a robust internet technology through its control mechanism, which allows us to transmit data without loss.

Furthermore the lwIP provides data transmission via UDP [10] (User Datagram Protocol). UDP does not guarantee reliability or ordering of the data transmission like TCP does, but avoids a lot of overhead of checking whether every data packet actually arrived. This makes the transmission quite faster, and allows a quick transmission of files via the TFTP [11] (Trivial File Transfer Protocol).

LwIP is scalable for several platforms and optimized for embedded systems. The total TCP/IP memory consumption in the embedded system can be varied by changing the send window size. A typical TCP send window of 11000 bytes leads to a total RAM consumption of 16 KB [8].

2.3.2 Embedded Web Server

A standard browser like Firefox or Internet Explorer) sends a request for an HTML-site (Hypertext Markup Language) stored on the embedded web server. The embedded web server receives the requests in form of a HTTP-request (HyperText Transfer Protocol). The embedded web server interprets this requests and replies to the client application by sending a HTML-site. The basic HTML-frames are stored on the Web ECU and dynamically build, contingent on the state of the Web ECU and the remote debug applications. Depending on the client request and the state of the Web ECU, the embedded web server initialises applications of the remote debugger.

The remote debugger on the ECU is responsible for buffering and distribution of data, which is sent by the client or collected from ECUs or Sensors. It buffers e.g. the test data until it is fetched from the application web server, or forwards CAN messages and software to other ECUs.

3 Applications

We developed applications, which take advantage of the wireless communication with vehicles. This applications are advantageous especially during the development phase, by expediting the development process. A characteristic feature of the development process is a frequent change of software running on ECUs, or the need for verifying test data. Beyond this, further applications are supposable, if we focus the analysis of subsets of sensor data, e.g. localisation of vehicles using GPS.

3.1 Flash over Web

The entire application software is stored in the flash memory of the ECUs. In order to install a new software, we had to dismantle the ECU until now and then flash the software via a hardware-debugger. During the developing- and testing phase we often have a change of the software, e.g. new settings for sensors or new algorithms. Therefore the developing process is interrupted by the time-consuming removal of the ECUs. Furthermore the vehicle maybe has to return from a test-drive.

We developed the application “Flash over CAN” [12] to avoid the dismantling of the ECUs. “Flash over CAN” enables us to flash a ECU via the CAN-channel. Therefore a laptop is connected to the CAN-channel of a vehicle and a new application can be transmitted to a specified ECU. In order to transmit a new software during a test drive, the “Flash over CAN”-module was extended by “Flash over Web”. The new software can now be uploaded via the internet to the Web ECU, which acts also as a file buffer for the software. After a successful upload, the file is transmitted to the ECUs via the CAN-channel.

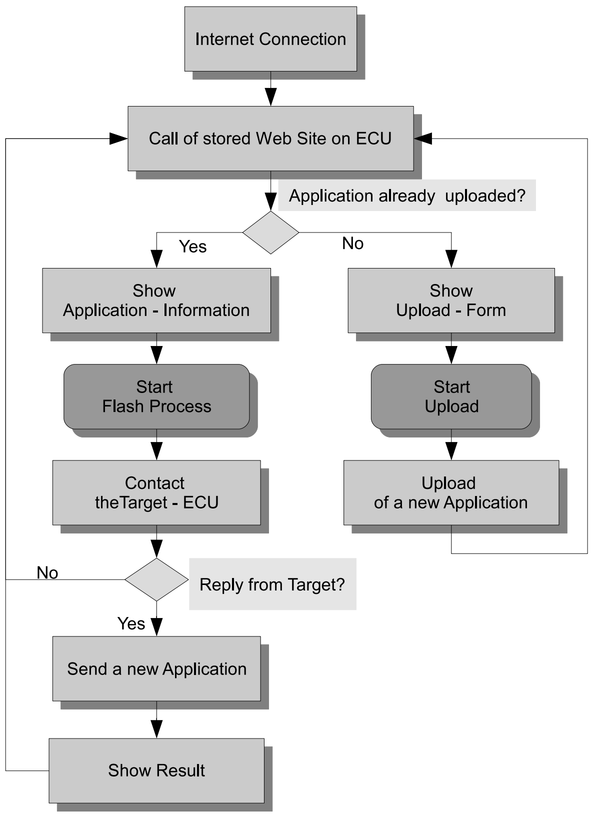

Figure 5: “Flash over Web” Flowchart

3.1.1 “Flash over Web”-Procedure

We have two main parts, necessary for the flash process. First task is the embedded web server itself, receiving the new application in a special file format (WUF, Web Upload File). The WUF-File contains amongst others, the ID of the target ECU, a checksum and the application as binary. The second task has to analyse the transmitted file and perform the flash process. After uploading the WUF-file, the client can view the file information in the browser and verify it. The client can change the WUF-file or start the flash process via the browser. After finishing the flash process the client is informed about the result. The general flow of “Flash over Web” is shown in figure 5. The flash process itself is performed by the “Flash over CAN”-module. This module assures the correctness of the included binary, e.g. checksum, file size and the correct write address for applications.

3.1.2 Upload WUF-Files

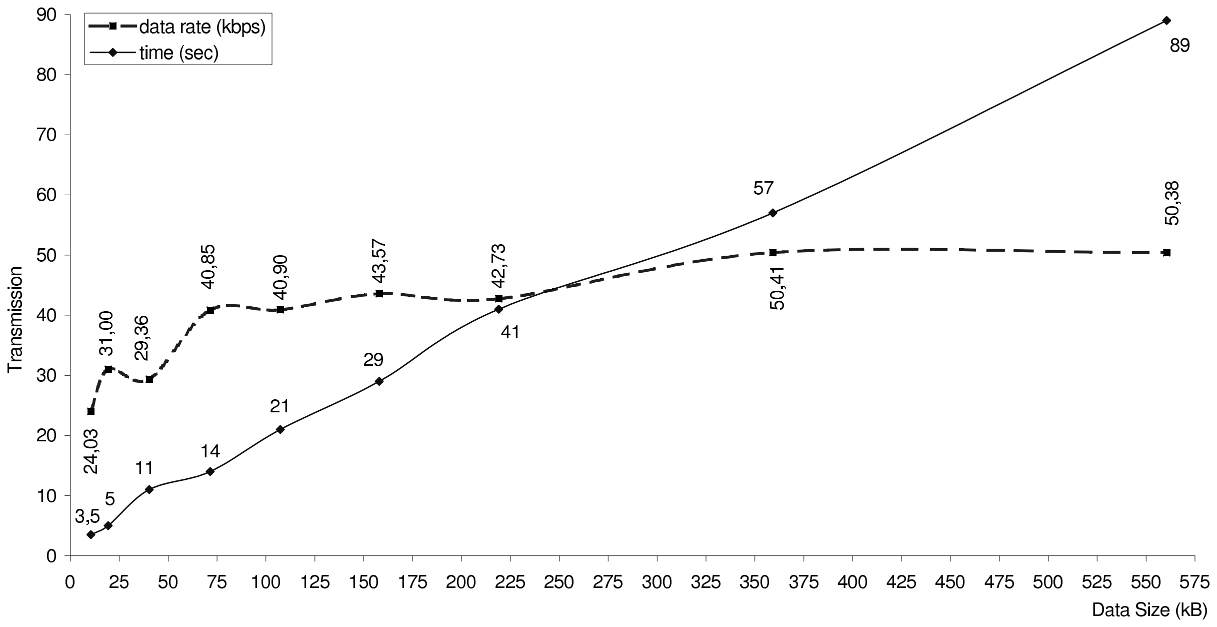

Figure 6: File Upload Transfer Rates using Class 10/12 GPRS Modems

As demonstrated in chapter 2.3 the data is transmitted via the TCP-protocol. The network characteristics of GPRS, like large Round Trip Times (RTTs), do not work well with current TCP/IP implementations. The RTT is the time between sending a packet and the receipt of an acknowledgement for it. In mobile networks RTT is generally substantially higher than in grid-bound nets. TCP was developed for a line-bound data communication and slows down the transfer rate, when a transfer error occurs. If, due to a slow transfer, the acknowledgement for a currently received packet is still on the way, the protocol assumes that a package did not arrive, and sends it again. As we can see in figure 6, we get a transfer rate of 40 - 50 kbps for uplink transfer, if we receive data on the server (downlink). Here a Class 10 GPRS-Modem is used. The WUF-files have a size of 500 - 700 kB, this leads to an upload duration of approximately 112 seconds for 700 kB file size. Using a EDGE-modem the duration reduces to approx. 40 seconds.

3.1.3 “Flash over Web”-Extension

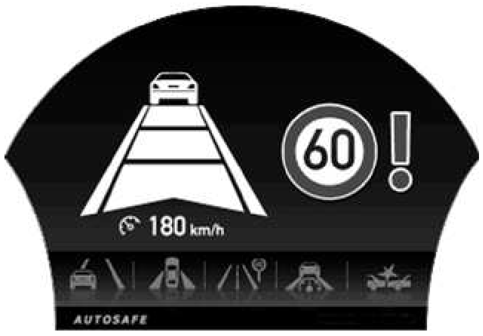

Figure 7: AUTOSAFE-Display integrated in Vehicle. The Images can be changed via Internet.

The “Flash over Web”-module is used now for the upload of applications for ECUs, but also can be extended by the upload of any file. Such files can be for example images shown on the AUTOSAFE-display ,see figure 7. Before uploading, the files have to be converted into the WUF-File. The write address is part of the WUF-file header,so the “Flash over CAN”-module needs a table of the layout of the flash memory to assure, that no other important parts of the flash are overwritten.

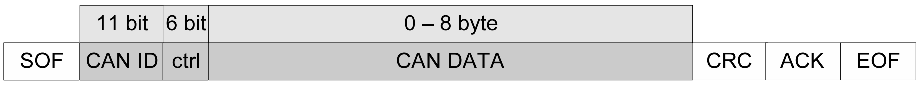

Figure 8: Example Standard CAN Message

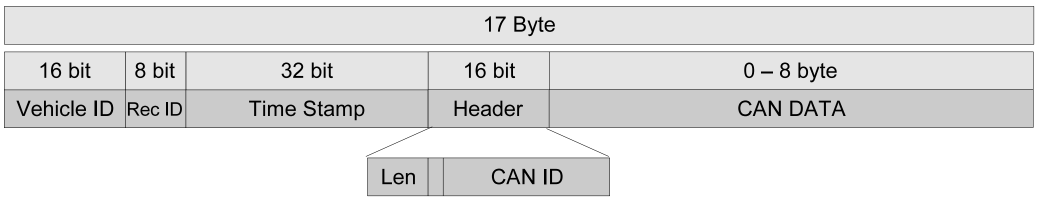

Figure 9: Remote CAN Message (RCM)

3.2 CAN Recorder

Via the CAN channel we have access to different vehicle information like the position, break pressure or the engine speed. To record specified data we can choose CAN IDs and send a record request for a specified time to the Web ECU. The messages are sent immediately to the application web server, part of the BCS, where they are stored in a data base, see figure 2.

The messages are transformed from the standard CAN message format (see figure 8) to a Remote CAN Message (RCM), as shown in figure 9. The RCM includes the vehicle ID, a unique identifier for every vehicle which is connected to the backend control system. The Record ID enables us to store several different records, recorded on a vehicle. The time stamp, header and the CAN data itself can be converted in data formats of CAN-message analysing tools, e.g. the CANalyzer [13], to visualize the records in time flow.

46 byte	Payload max. 1454 byte			
Network Protocol Headers	RCM	RCM	RCM	...

Figure 10: Remote CAN Messages, Packed in a TCP-Message.

Via TCP we can assure, that all messages receive the application web server, but the transfer rate limits the recorded CAN data. The standard MTU (Maximum transmission unit), defined by the PPP-protocol, is 1500 byte, reduced by the underlying network protocol headers to 1454 byte payload for a TCP message, as shown in figure 10. It is reasonable to store the messages until the payload of the TCP is filled up to avoid additional overhead, produced by the network protocol headers. Thereby the records have to be buffered until the TCP signalled a successful transmission. Using a class 12 GPRS-modem, we have the same transfer rates as shown in figure 6 and are able to transmit approximately 370 remote CAN messages per second.

4 Summary

Getting a wide area connection to vehicles brings a lot of advantages along. A quick reaction to new findings during a development phase is facilitated by applications, tailored to an internet connection. A lot of additional functions are possible in this field, just thinking about the localisation of a vehicle or the exchange of data between vehicles through the backend server.

5 Acknowledgement

This work was supported by the German Research Ministry (BMBF). The authors want to thank the Nanoelectronic unit of the BMBF for granting AUTOSAFE under funding no. 01M3076.

References

[1] Robert Bosch, *CAN Specification* , 1991

[2] WiSOA, *www. wisoa. net* , 2007

[3] R. Kalden, I. Meirick, M. Meyer, *Wireless Internet Access Based on GPRS*, IEEE Personal Communicatons, April 2000

[4] 3rd Generation Partnership Project (3GPP), `www.3gpp.org`

[5] OSEK/VDX, *OSEK OS*, `http://www.osek-vdx.org`, 2005

[6] Freescale Semiconductor, *MPC5200*, `www.freescale.com`, 2005

[7] A. Dunkels, *Design and Implementation of the lwIP TCP/IP Stack*, Swedish Institute of Computer Science, `www.sics.se/~adam/lwip/`, 2001

[8] A. Dunkels, *Full TCP/IP for 8-Bit Architectures*, Swedisch Institute of Cumputer Sience, 2000

[9] Dapra Internet Program, *Transmission Control Protocol*, RFC 768, USC/Information Sciences Institute, September 1981

[10] J. Postel, *User Datagram Protocol*, RFC 768, USC/Information Sciences Institute, August 1980

[11] K. Sollins, *The TFTP Protocol (Revision 2)*, RFC 783, MIT, July 1992

[12] A. Spitzkopf, *Entwicklung und Umsetzung eines ber CAN-Bus angesteuerten Flashloaders*, Diploma Thesis, University of Applied Sciences Amberg-Weiden, January 2008

[13] CANalyzer , *www. canalyzer. de* , Vector, 2007