Base technologies for vehicular networking applications: review and case studies

M. Rodelgo-Lacruz, F. J. Gil-Castiñeira, F. J. González-Castaño, J. M. Pousada-Carballo
Departamento de Ingeniería Telemática, Universidad de Vigo, Spain
javier@det.uvigo.es

J. Contreras
CTAG, Spain
jordi.contreras@ctag.com

A. Gómez
CESGA, Spain
agomez@cesga.es

M. V. Bueno-Delgado, E. Egea-López, J. Vales-Alonso, J. García-Haro
Departamento de Tecnologías de la Información y las Comunicaciones, Universidad Politécnica de Cartagena, Spain
javier.vales@upct.es

***Abstract*— In this paper, we review the state of the art of two key hardware technologies that support vehicular applications: on-board embedded systems and wireless sensor networks (WSN). We focus on pre-competitive or state-of-the-art hardware, and illustrate its use with two case studies: on-line navigation assistance and data collection in a mobile WSN. In the first case (based on a joint collaboration within project FUNCMOV PGIDIT05TIC00501CT, Xunta de Galicia, Spain), we describe our development experience with automotive embedded systems. In the second case, we analyze the feasibility of wake-up schema to gather data from highly dispersed sensor nodes. The goal of the paper is to offer a perspective on the current possibilities of these hardware systems.**

I. INTRODUCTION

This paper offers a perspective on the current possibilities of some recent hardware systems to implement advanced vehicular applications: embedded systems for automotive applications and WSNs. We illustrate their possibilities with two case studies: on-line navigation (for the former) and mobile data collection (for the latter).

The paper is organized as follows: section II describes modern car electronics. Section III introduces WSNs. In section IV we present our first case study: the development of an embedded on-line navigation system with a multimedia/telematics electronic control unit (ECU). In section V we describe WSN wakeup schema in vehicular applications. Section VI concludes the paper.

II. ELECTRONICS FOR AUTOMOTIVE APPLICATIONS

A. Automotive electronic components

Current automotive systems comprise many electronic subsystems with different communication needs. These subsystems can be classified in functional domains [15] as follows:

- *Chassis*: suspension, steering and braking control.
- *Powertrain*: engine and transmission control.
- *Safety* (both active and passive).
- *Human Machine Interface* (HMI).
- *Body* (mostly comfort issues).
- *Multimedia/telematics*.

All of them require multiple electronic components, but in this section we focus on the core devices.

a) Powertrain: powertrain systems are computationally demanding, since they must control the engine with response times of few milliseconds. The MPC500 32-bit MCU family [1], [2] has been used in powertrain control applications like the *Valvetronic System* of BMW.

b) Chassis: this domain includes VDC-ESP (Vehicle Dynamics Control - Electronic Stability Program), ABS (Anti-lock Brake System), ASC (Automatic Stability Control) and 4WD (4-Wheel Drive). These systems are critical from the point of view of safety, which imposes strong communication requirements on 32-bit microcontrolers (like the MPC500 [1]). Furthermore, x-by-wire technologies are being introduced to assist braking or steering (a robust network like FlexRay [3] is essential).

c) Safety: safety systems are time-critical. They include airbags, impact and rollover sensors, adaptive cruise control, distance control systems, etc. In this case the amount of data to process is low, and it is possible to employ devices with lower performance like the S12X 16-bit family [4].

d) HMI: HMI seeks to provide friendly safe interfaces.

e) Body: these systems control familiar devices such as mirrors, dashboard, doors, windows, seats, wipers and lights. Any automotive certified microcontroller (S12X or the M68HC08 family of 8-bit MCUs [5]) can control these non critical elements.

f) Multimedia and telematics: they require a large bandwidth for the exchange of data within the car or with external devices. This domain is becoming very important due to the relevance of telematics in modern life: hands-free telephony, car navigation systems, CD, DVD, rear-seat entertainment, remote diagnostics, etc). The MPC5200B [6] System On Chip (SoC) is a complete core to power a telematics system [7].

There exist different types of networks to interconnect the ECUs of the functional domains. The J1939 standards of the Society for Automotive Engineers (SAE) define three classes of networks or field buses [8]:

- Class A networks manage low data rates (less than 10

1-4244-0755-9/07/$20.00 ©2007 IEEE

Kbps). They rely on inexpensive devices and transmit simple data like body (door locks, internal lights, rain sensor, ...), entertainment, trip computer data, etc. An example of class A network is LIN [9].
- Class B networks work at higher bitrates (10 to 125 Kbps) and support ECU communications. Low-speed CAN [10] is an example. They transmit general information like instrument cluster data, vehicle speed, emissions data, etc.
- Class C networks: the main class C protocol is CAN 2.0 [11]. Class C protocols work at high speeds (125 Kbps to 1 Mbps). The powertrain and chassis domains employ them.
- Class D networks: new applications like multimedia systems and x-by-wire require even higher bandwidths. MOST [12] is a multimedia protocol, and FlexRay [3] supports prediction and fault tolerance.

B. On-board diagnostics

OBD (On-Board Diagnostics) [13] was developed in the US in the seventies as a response to car pollution. OBD systems sense engine performance and adjust it to minimize pollution. They also assist in early diagnostics. There are diverse OBD protocols, but we focus on ISO 15765 CAN [14].

Since 1996, all the cars in the US market must comply with the OBD-II specification. Since 2001, all gas engines in the EU market must have an EOBD (a variant of OBD-II) interface. It is possible to query on-board computers through that interface to obtain sensor data.

III. Wireless sensor networks

WSNs have been designed to be embedded in the environment. They sense and collect data of interest in their surroundings. WSN nodes -or *motes*- support a wide range of applications: hands-free access control, vehicular control of gates and barriers, personnel location control, asset management, access for handicapped persons, assistance in emergency evacuations, etc. These tasks require the measurement of many different physical parameters. Commercial motes monitor temperature, light, vibration, sound, radiation, acceleration, magnetic fields, and position. Indeed, it is easy to extend motes to interface with highly specialized sensing hardware. In "conventional" WSN deployment, the selection of a particular type of motes usually depends on radiocommunication and networking requirements such as network topology, transmission range and throughput.

The *high mobility* of both data acquisition nodes and data collecting nodes characterize the applications of vehicular networks. Mobile nodes must transmit collected data to acquisition points -maybe mobile as well- in real time. The amount of data can be considerable, since information exchanges may be sporadic. Thus, mobile nodes should have a high transmission capacity, a long transmission range and enough storage capacity with low access time to ensure a reliable data transmission within a short timeframe (typically a few seconds).

The emerging sensor hardware industry and the research projects in WSNs have produced a wide range of devices. In the following section we describe the most relevant hardware platforms available. Tables I and II summarize their main features.

A. Commercial sensor nodes

The University of Berkeley and the Smart Dust Project [16] are developing the most popular sensor nodes in the market. They developed the WeC mote in 1999, followed by the Rene and MICA motes. MICA2, MICAZ [17] and Telos [18] are the latest versions of these families. They satisfy most requirements in typical applications: low power consumption (over one year with AA batteries), multi-channel transceiver with medium range (up to 200 meters, depending on the frequency), unicast/multicast and broadcast transmission to base stations or other acquisition nodes, and an open-source operating system: TinyOS [19]. Moreover, MICAZ and Telos employ Zigbee, the IEEE 802.15.4 standard for Wireless Personal Area Networks (WPAN).

ETH Zurich has designed a sensor family called BTnode [20], which uses the same low cost microcontroller and radio transceiver as MICA2. However, BTnode also includes a Bluetooth radio interface. Both interfaces can operate simultaneously. It is possible to disable them independently to reduce power consumption. When using both radio interfaces, BTnode has a shorter life than MICA2/Z.

The Embedded Sensor Board (ESB) [21] by Berlin University is more complex than the previous systems. It has the ability to communicate via GSM/GPRS modules. ESB has motion sensing capability (by means of infrared sensors) to disable the mote hardware. In certain applications, life time reaches 5 years.

Medusa MK-2 is among the most powerful sensor nodes [22]. UCLA designed it for the Smart Kindergarten project. It consists of two microcontrollers and it has the same radio transceiver as MICA2. It admits wired and wireless communications. Like BTnode, it has the ability to disable idle modules.

Finally, iBadge [23] is another UCLA sensor node that, unlike Medusa, includes a Bluetooth chip and a location unit to estimate the relative position of the node. This complex hardware results in a high cost per unit. iBadge can act as a gateway between wired and wireless networks.

B. Research projects

Currently, there are a number of research projects pursuing the development of WSN nodes. Among them we can cite:

The Eyes European Project has designed the EYES sensor node [25] to test and check energy-efficient network algorithms under the proprietary PEEROS operating system (Preemptive EYES Real Time Operating System).

Imote [26] is another sensor node by Intel that employs highly sophisticate modules, such as the Intel StrongArm/Xscale 32-bit processor.

Other research initiatives are μAMPS [24] (energy-efficient constraints, self-configuration and flexibility), Pushpin [27]

Model	Transceiver	Bandwidth (Kbps)	Range	Operating frequency	Transmitter power	Sensitivity	CRC	FEC
MICA	TR1000	40 Kbps	$\leq$100 m	433/916 MHz	0 dBm	-97 dBm	no	no
MICA2	CC1000	38.4/19.2 Kbps	$\leq$100 m	433/916 MHz	0 dBm	-110 dBm	yes	yes
MICAZ	CC2420	250 Kbps	$\leq$125 m	2.4 GHz ISM band	-24 dBm to 0 dBm	-94 dBm	yes	yes (ARQ)
TELOS	CC2420	250 Kbps	$\leq$125 m	2.4 GHz ISM band	-24 dBm to 0 dBm	-94 dBm	yes	yes (ARQ)
BTnode	CC1000	$\leq$76.8 Kbps	$\leq$100 m	ISM band/Bluetooth	-20 to 10 dBm	-110 dBm	yes	yes
ESB	TR1001	115.2 Kbps	$\leq$300 m	ISM band	1.5 dBm	-106 dBm	no	no
MEDUSA MK-2	TR1000	40 Kbps	$\leq$20 m	433/916 MHz	0 dBm	-97 dBm	no	no
iBadge	TR1000 / ROK10107	40 Kbps	$\leq$20 m	433/916 MHz	0 dBm	-97 dBm	no	no
EYES	TR1001	115.2 Kbps	$\leq$300 m	ISM band	1.5 dBm	-106 dBm	no	no
iMote	Bluetooth	$\leq$500 Kbps	$\leq$30 m	2.4 GHz	4 dBm	n/a	n/a	n/a

TABLE I

RADIO COMPARISON

Model	Microcontroller	Memory	Operating System	battery life/type
MICA	Atmel ATMega 128	4 KB SRAM/128 KB flash	TinyOS	$<$1 year/AA
MICA2	Atmel ATMega 128	4 KB SRAM/128 KB flash	TinyOS	$<$1 year/AA
MICAZ	Atmel ATMega 128	4 KB SRAM/128 KB flash	TinyOS	$<$1 year/AA
TELOS (Moteiv)	MSP430F149	2 KB RAM/69 KB + 256 B flash	TinyOS	$<$1 year/AA
BTnode	Atmel ATMega 128	64+180 KB SRAM/128 KB flash	BTnode System Software and TinyOS	$<$1 year/AA
ESB	TI MSP 430	2 KB/60 KB	TinyOS/Contiki	$<$5 years/AAA
MEDUSA MK-2	STMega128L/ATMEL ARM THUMB	136 KB RAM/1 MB flash	PALOS and μCOS-II	Rechargeable Lithium Ion
iBadge	Atmel ATMega 128	4 KB SRAM/128 KB flash	PALOS and μCOS-II	Rechargeable Lithium Ion
EYES	MSP430	2 KB RAM	PEEROS	$>$1 year
iMote	ARM7TDMI core	64 KB SRAM/512 KB flash	TinyOS	$<$1 year/AA

TABLE II

MOTE HARDWARE COMPARISON

and Tribble [28] (all by MIT), Pin&Play [29] (by Lancaster University) and the WINS platform, which has developed the Rockwellis WINS nodes [30].

IV. CASE STUDY 1: ON-LINE NAVIGATION ASSISTANCE

In project FUNCMOV (PGIDIT05TIC00501CT, Xunta de Galicia, Spain) we programmed an on-line navigation system on a multimedia/telematics embedded ECU. The system is equipped with a GPS antenna and a touchscreen. However, instead of calculating routes locally, the system sends the coordinates via a wireless link (GPRS in our case) to a central server, which calculates the route and returns it. Obviously, the main disadvantages are dependence on third parties and communications cost, but the system takes advantage of real-time variable information like weather, blocked roads, gas prices and the like. Besides, software and map updates are transparent to the user.

A. Hardware

We selected the PC/104 architecture [31] for the multimedia/telematics ECU in the project. Figure 1 shows the prototype (the touchscreen was integrated in the vehicle dashboard). Its main characteristics are:

- MOPSlcd6 main PC/104 board:
  - Pentium MMX 166 MHz.
  - 256 MB RAM.
  - Integrated graphics controller.
  - 1 GB flash HD.
- Siemens TC65 GSM/GPRS module (serial).
- Laipac GPS module (serial).
- Xenarc 700TS 7" TFT LCD Touchscreen (USB).

Fig. 1. System prototype

- HE104+DX PC/104 power supply.

B. Software

The client employs remote procedure calls (RPC) to access server cartography and route functions. For this purpose we chose Web Services (WS) [32]. Web services use SOAP (a XML messaging protocol). Messaging protocols are less efficient than binary ones, but they are specially adequate for integrating many services and acquiring real time information from heterogeneous sources.

We selected Linux for the ECU due to its efficiency in embedded systems and the many libraries and documentation available, which simplify the development cycle. Specifically,

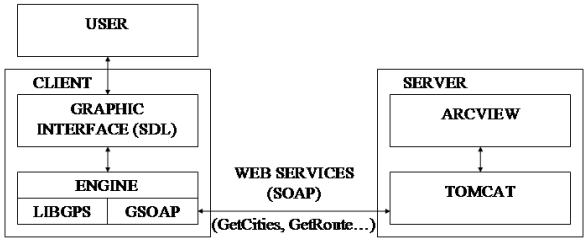

Fig. 2. Software design diagram

since the ECU memory is quite limited, we selected the Debian-based Damn-Small Linux distribution [33] with a custom 2.4.31 kernel. The *libgps* and SDL libraries allowed us to access GPS data and control the graphic interface, respectively. The WS client was developed with the gSOAP toolkit.

For the sake of clarity, the client design has two parts: the engine and the graphical interface. The engine runs as a daemon. It captures positioning data from the GPS daemon, interacts with the navigation assistance server, performs map-matching calculations, and returns the route information to the graphic interface through a data socket. The graphic interface passes destination data (cities, streets and home numbers) to the engine and displays the corresponding route information on the touchscreen.

The server is a Windows program, since the ArcView GIS [34] we used is only available for that platform. Apache Tomcat [35] with the AXIS WS platform implementation [36] ran on the server machine.

Figure 2 shows the software design diagram.

C. Operation

The user enters the first letters of the destination city name with a virtual keyboard on the touchscreen. The client invokes a server procedure and displays the list of cities it retrieves (Figure 3). The user selects the city by tapping on it and chooses between navigation and introduction of a street name. In case of the latter, the streets are displayed. Once again, the user can navigate or introduce a house number. In the second case, the client calls a procedure to get the coordinates of that address. When the user taps on the *navigate* command, the client sends the current GPS coordinates and the city/street/house destination coordinates to the server, which calculates the route and returns its set of coordinates and the corresponding indications. Initially, the client only requests the starting part of the route to quickly initiate navigation and reduce latency. Once that part is available, the client transparently requests the entire route.

When the client has the information, the route is visually indicated by pictograms (Figure 4). The display shows information like distance to the next turn, next indication, distance left, current speed, current street, next street and final destination. If the user takes a wrong turn, a new route is transparently and automatically requested to adjust the indications to the new context.

Fig. 3. Cities found

Fig. 4. Navigation assistance operation

V. Case Study 2: WSN Wake-up Schema in Vehicular Applications

There is a wealth of research on WSN energy saving. Many studies focus on wake-up schema to reduce energy consumption. However, there is a tradeoff between energy savings and wake-up delay (elapsed time since a packet arrives to an active node until that node passes the packet to the MAC layer) [37],[38].

The interesting wake-up scheme in [39] improves WSN energy savings without increasing end-to-end delay. This scheme is adequate for vehicular applications, since wake-up time must be as low as possible to ensure correct data acquisition from mobile collecting nodes.

Figure 5 depicts the operation of the wake-up scheme. It needs a tone channel and the regular data channel (thus the nodes need a multi-channel transceiver). When an active node has pending packets, it sends a wake-up tone through the

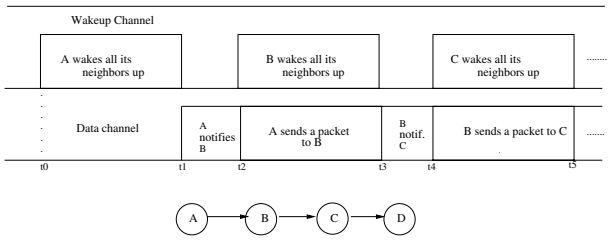

Fig. 5. Pipelined Tone Wakeup or PTW

special tone channel. The nodes that receive this signal become active and start listening to the data channel. The Pipelined Tone Wakeup (PTW) scheme in [39] reduces wake-up delay: while active nodes wait for incoming packets, they send the wake-up tone to their neighbors. In figure 5, node A wakes all its neighbors up. After this, the neighbors wait for a data packet in the data channel. Node A sends a short notification packet through the data channel to indicate that a single active node (node B) is the receiver. Then, node B acknowledges the notification to A, and it transmits the wake-up tone to its neighbors. Meanwhile, A sends the data packet to B via the data channel. In the PTW scheme only the wake-up delay of the first hop contributes to end-to-end packet delay. This initial delay includes queueing, channel access, transmission, and propagation delays.

Note that PTW can be seamlessly applied in a mobility context, because the notification stage can be used for neighbor discovery. All nodes nearby are unknown *a priori* in a vehicular network. Therefore, it is mandatory to analyze and select a suitable set of nodes for packet delivery. PTW notification packets may be useful for this task. The notification delivery packet must include a set of labels identifying the desired capabilities of the receptor (e.g. node type, physical address, free memory required, etc). Then, the receptors that qualify acknowledge notification and start waking neighbors up for a possible transmission in the next hop. If at least one acknowledgement is received, the sender broadcasts its data to the receivers, which process those data or retransmit them. This scheme allows a fast forward packet delivery, since it guarantees that the neighbors in the next hop are active for reception without further delay.

Regarding practical implementations, the authors of PTW recommend TR1000 radio transceivers (MICA, iBadge and EYES nodes would be adequate). Let us assume that the relative speed between a sender and a receiver is 50 km/h (180 m/s). Then, from Table I, the amount of data that can be transferred is: 347 Kb (MICAZ), 8.9 Kb (iBadge) or 384 Kb (EYES). Therefore, both MICAZ and EYES motes achieve a high transfer rate, and they could support vehicular network applications with a transmission procedure based on PTW.

VI. Conclusions

In this paper we have briefly described two base technologies for advanced vehicular networks: car electronics and WSNs. We have classified them, identified some key players and exemplified their usage with two case studies: multimedia/telematics ECUs for on-line navigation assistance and wake-up schema in mobile WSNs.

Acknowledgement: this research has been partly funded by Xunta de Galicia grant PGIDIT05TIC00501CT and *Ministerio de Educación y Ciencia* project m:Ciudad (FIT-330503-2006-2), Spain.

References

[1] *The MPC500 Family of 32-bit Embedded Controllers from Motorola*. Rudan Bettelheim, 2001.
[2] *The MPC500 Family of 32-Bit Embedded Controllers*. Freescale Semiconductor, 2004.
[3] *FlexRay Protocol Specification V2.1 Rev. A*. FlexRay Consortium, 2005.
[4] *MC9S12XE-Family Data Sheet, Rev. 1.02*. Freescale Semiconductor, October 2006.
[5] *MC68HC908QC16, MC68HC908QC8, MC68HC908QC4 Data Sheet*. Freescale Semiconductor, 2006.
[6] *MPC5200B Data Sheet*. Freescale Semiconductor, March 2006.
[7] *MPC5200B Data Sheet*. Freescale Semiconductor, January 2006.
[8] SAE J1939 Standards Collection on the Web. http://www.sae.org/standardsdev/groundvehicle/j1939.htm.
[9] *LIN 2.0 specification package*. Lin Consortium, September 2003.
[10] ISO 11519-3: Road vehicles – Low-speed serial data communication – part 3: Vehicle Area Network (VAN), 1994.
[11] ISO 11898-2: Road vehicles – Controller Area Network (CAN) – part 2: high-speed medium access unit, 2003.
[12] *MOST Specification Rev 2.4*. MOST Cooperation, May 2005.
[13] *On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual*. SAE International, July 2006.
[14] ISO 15765: Road Vehicles Diagnostics on Controller Area Networks (CAN). International Organization for Standardization, 2004.
[15] N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, "Trends in automotive communication systems," Proceedings of the IEEE 93/6 (2005).
[16] Smartdust project. http://robotics.eecs.berkeley.edu/ pister/SmartDust/
[17] MICA MOTES. http://www.xbow.com.
[18] J. Polastre, R. Szewczyk and D. Culler, "Telos: Enabling ultra-low power research," Proc. 4th International Conference on Information Processing in Sensor Networks, Los Angeles, USA, 2005.
[19] TinyOS. http://www.tinyos.net.
[20] J. Beutel, O. Kasten, F. Matter, K. Roemer, F. Siegemund and L. Thiele, "Prototyping sensor network applications with BTnodes," Proc. IEEE European Workshop on Wireless Sensor Networks (EWSN'04), Berlin, Germany, January 2004.
[21] Freie Universitat Berlin, Computer System Telematics, Scatter Web Project. http://www.inf-fu-berlin.de/inst/ag-tech/scatterweb-net.
[22] J. Polastre, J. Hill and D. Culler, "Versatile Low Power Medium Access for Wireless Sensor Networks," Proc. ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), 2004, pp. 95–107.
[23] S. Park, I. Locher and M. Srivastava, "Design of wearable sensor badge for smart kindergarten," Proc. 6th International Symposium on Wearable Computers (ISWC2002), Seattle, USA, October 2002, pp.13.1-13.22.
[24] μAMPS project. http://www.mtl.mit.edu/research/ictsystems/uamps/projects.
[25] Eyes Project. http://www.eyes.eu.org/.
[26] L.F.W. van Hoesel, S.O. Dulman, P.J.M. Havinga and H.J. Kip, "Design of a low-power testbed for Wireless Sensor Networks and verification," Tech. Rep. R-CTIT-03-45, University of Twente, September 2003.
[27] PushPin. http://www.media.mit.edu/resenv/Pushpin.
[28] J.A. Paradiso, J. Lifton and M. Broxton, "Sensate media. Multimodal electronic skins as dense sensor networks," BT Technology Journal 22 (2004), pp. 32-44.
[29] K.V. Laerhoven, N. Villar and H-W. Gellersen, "Pin&Mix: When Pins Become Interaction Components," Proc. Workshop on Real World User Interfaces- Mobile HCI Conference, Udine, Italy, September 2003.
[30] Wireless Integrated Sensor Networks. http://www.janet.ucla.edu/wins.
[31] PC 104. http://www.pc104.org.
[32] Web Services. http://www.w3.org/2002/ws/.
[33] Damn Small Linux. http://www.damnsmalllinux.org/.
[34] ArcView. http://www.esri.com/.
[35] Tomcat. http://tomcat.apache.org/.

[36] AXIS. http://ws.apache.org/axis/.
[37] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, "Topology Management for Sensor Networks: Exploiting Latency and Density, " Proc. ACM Mobicom 2002.
[38] W. Ye, J. Heidemann and D. Estrin, "An Energy Efficient MAC Protocol for Wireless Sensor Networks," Proc. IEEE Infocom 2002.
[39] X. Yang and N.H. Vaidya, "A Wakeup Scheme for Sensor Networks: Achieving Balance between Energy Saving and End to End Delay," Proc. 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04).