AN INTELLIGENT REMOTE SENSOR ANALYSER FOR ELECTRONIC AUTOMOTIVE SYSTEM

P. Mahalakshmi[1], R. Karthikeyan[2], N. GowriShankar[3]
Department of Electrical and Communication Engineering
Chennai, India.

Abstract: ***The technology innovation leads the automotive system to embed testing and diagnosis on-board. In the car automotive system, vehicle testing and diagnosis requires huge amounts of data to be gathered and analyzed. It is not possible to store all the data's due to the limited memory available in a tested vehicle. On-board preprocessing of data and decisions about which information has to be kept or omitted is thus vital for vehicle testing routines. This model introduces a method for flexible on-board processing of sensor data of a vehicle. A processing graph model for automotive applications is proposed, which consists of operator nodes and connecting streams. This method supplies both recording and processing functionality together. To account for dynamic changes of conditions within a vehicle most of the time only a small portion of the vehicle states are interesting for diagnosis both the model and actual software are built in a such a way that the whole system can be automatically be adapted at runtime whenever certain conditions are detect. This model can be adapted to any other automotive system such as electrical monitoring system, cyber-physical systems etc.***

Keywords: Data aggregation, embedded systems, on-board diagnosis, stream processing, and vehicle sensor data.

I. INTRODUCTION

The complexity of vehicles has increased in recent decades and will continue to increase significantly in the future. Until the late 1960s, cars were basically mechanical systems with a few electrical appliances, e.g., for engine spark and lighting. Modern vehicles are complex electro-mechanical systems with dozens of networked electronic control units (ECUs). ECUs enable or implement vehicle core functions such as power-train control, suspension control, safety, convenience functions, and infotainment. They are connected to a large number of sensors and actuators which they control. ECUs exchange information about their current sensor values over internal networks (for example, a CAN bus), so that multiple redundant sensors are avoided. The types of sensors used in the car are of a great diversity, ranging from pressure sensors over temperature to acceleration and contact sensors.

Data resident and stored on ECUs and data exchanged between ECUs describe, from the technical stand point, the state of the vehicle at any time. To capture, analyze, and interpret this data are important activities of engineering testing and quality control processes. In recent years, the availability of cost-effective in-vehicle and off-board computing and communications systems enabled the systematic acquisition and processing of data from many vehicles over long periods of time. This is particularly important for the late engineering and early production phases of a vehicle's life cycle. The data volume generated from hundreds of sensors, operating at a high frequency (see Figure. 1), is immense. Despite the improvement in computing infrastructure, it is still necessary to utilize filter and data aggregation mechanisms and only record detailed data for specific situations.

Another important application is the area of vehicle diagnosis. Current vehicle diagnosis relies on error codes [(Diagnostic Trouble Code (DTC)], that the corresponding ECU sets. These error codes may be retrieved via the on-board diagnostic socket (OBD-II), which became a mandatory standard for the United States in 1996. This diagnostic interface also allows to acquire the various sensor data from the ECUs of a vehicle in real-time.

Figure 1: Bus Systems

II. ARCHITECTURE AND RELATED WORK

This work aims at a flexible data aggregation system, that can dynamically adapt its behaviour at runtime, depending on the specific state of certain subsystems of the vehicle (e.g., the engine). As a reaction to critical events, our system is able to read and process sensor values at a higher rate and adapt recording. The adaptiveness can also be employed in order to store data selectively. This work aims at solving the industrial problem of vehicle diagnosis-keeping costs low while maintaining flexibility, maintainability and quality of recorded data-by employing and adapting research results from different fields.

A. Bus Architecture

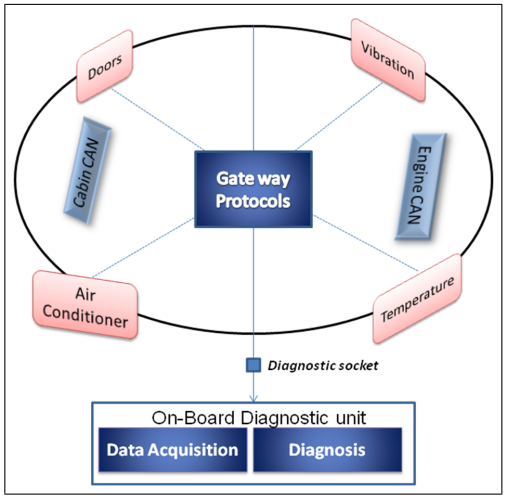

Figure 2: System Architecture: The On-Board Unit (OBU) is connected to the diagnostic CAN bus. It communicates with the ECUs through the Central Gateway.

Diagnostic tools are not directly connected to the vehicle's engine or cabin CAN bus system. Instead, they are connected to the OBD socket, which is connected to a separate diagnostic CAN bus [8]. Communication with the ECU is performed over a gateway within the vehicle. Because there is no direct access to the other vehicle buses from the diagnostic interface (i.e., only certain messages are routed), this architecture is considered to be more secure and safe, compared to directly hooking up to, for example, the engine CAN bus. Also, the OBD plug and the diagnostic protocols were standardized in 1996, so that diagnostic devices can be used for different vehicle types and even different manufacturers. While the primary targets for the OBD-II standardization were emission control systems, it also covers various other subsystems. The generalized bus architecture can be seen in Figure. 2. There are up to six separate CAN buses in today's vehicles.

B. Diagnostic Protocols

Figure. 3 shows the stack of protocols used for a diagnostic application. The Keyword Protocol (KWP) is the widely accepted standard for electronic diagnosis of vehicles [7]. When a diagnosis tool is attached to the OBD socket, it connects to the diagnostic CAN bus and uses the KWP as communication layer. The protocol usually runs on top of the CAN bus, although implementations also exist for other buses, such as LIN or MOST. Communication is performed in a request/response manner. In every request packet, there must be an ECU's identifier, which for a CAN bus is the address ID, and a Service Identifier (SID) (read, write, and others). An additional parameter tells the ECU about the precise set of data that is requested. For example the read and write SID require a field called LID (Local Identifier), which maps to memory or registers, whose

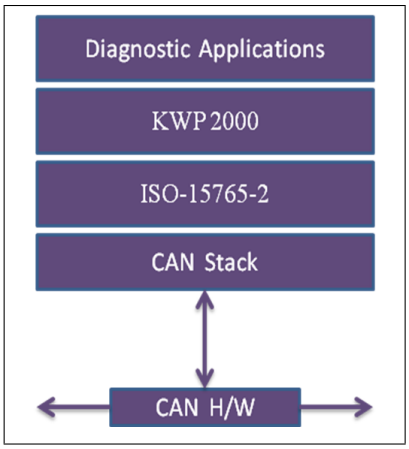

Figure 3: The diagnostic protocols of software architecture

values are sent back in a KWP response message. Various other services, such as uploading a new firmware to the ECU, are supported by KWP.

The ISO 15765-2 protocol is used for the segmentation of CAN frames [8]. As a normal CAN frame can only hold 8 bytes of payload and does not support the fragmentation of large data, the glue layer is introduced. An ISO 15765-2 packet can be up to 4 kilobytes big. It divides the data into separate CAN packets with 7 bytes of payload each.

C. Stream Processing Model for Automotive Sensor Data

Stream processing systems [10] define a computational pattern. Data are processed on-the-fly, so that the original data that may be of rather high volume can be discarded after the computation and only aggregated and filtered data need to be saved or evaluated further. It is related to data flow architectures in a way that data are processed immediately, if available at the inputs. The most promising work as in-vehicle system is probably VEDAS[11], which follows a distributed data-mining approach for automotive sensor data. Similar to our work, they are exploiting the existing diagnosis infrastructure of the vehicle. In fact, their system also collects aggregated, statistical data, but does not allow an on-the-fly modification of the data processing. Their goal is towards detecting misbehaviour of the vehicle, for example at rental car agencies. This is a rather different focus compared to needs of engineering testing.

The underlying model of stream processing system is a directed acyclic graph (DAG), which consists of different kinds of processing nodes. These processing nodes (operators) are connected by data paths as edges (streams). We call this graph a Stream Processing Graph (SPG). The graph can be adjusted to different situations to allow a varying scope of data processing and recording. Triggers for the adjustment of the graph are defined as part of the graph itself extension of the stream processing graph by event-action pairs allows an adaptation of the graph at runtime. The event triggers for the rearrangement and reconfiguration of the SPG are introduced. The two major

problems of previous recording systems in the vehicle are addressed: Customized aggregation and filtering of data is now possible and actions may be taken upon individually defined events.

III. IMPLEMENTATION

We have implemented a prototype system on an embedded Linux telematic device, which is equipped with a CAN transceiver. The TMS320DM6446 (also refer to as the DM6446) make full use of the TI DaVinci^{TM} technology, to meet the network application of encoding and decoding media processing requirements, a new generation of embedded devices. The diagnostic application will be processed by using the DM6446 with the help of Embedded Linux version 2.6. The main features of using DM6446 are High-Performance Digital Media SoC 513-, 594-MHz C64x^{TM} Clock Rate and 256.5-, 297-MHz ARM926EJ-SâĎć Clock Rate. ARM9 Memory Architecture has the following features of 16K-Byte Instruction Cache, 8K-Byte Data Cache, 16K-Byte RAM, 8K-Byte ROM. Each sensor readings will be sent to the ARM 7 controllers, after processed by each ARM 7 microcontroller, the readings will be sent through DSP Processor for Diagnosing purpose through CAN bus. The LPC 2129 is based on a 16/32 bit ARM7TDMI CPU with real time emulation and embedded trace support, together with 128/256 kilobytes (KB) of embedded high speed flash memory. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at maximum clock rate. With their compact 64 pin package, low power consumption, various 32-bit timers, 4-channel 10 bit ADC, 2 advanced CAN channels, PWM channels and 46 GPIO lines with up to 9 external interrupt pins these microcontrollers are particularly suitable for automotive and industrial control applications.

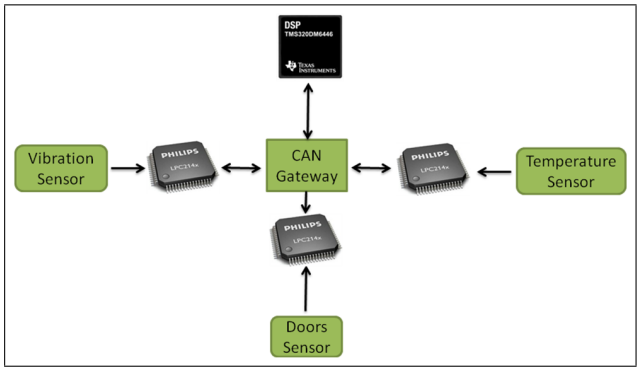

Figure 4: Implementation of On-board diagnostic Unit

IV. SIMULATION RESULTS

MATLAB SIMULINK : Simulink, developed by Math Works, is a commercial tool for modeling, simulating and analyzing multi domain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in control theory and digital signal processing for multi domain simulation.

Coupled with Simulink Coder, another product from Math Works, Simulink can automatically generate C source code for real-time implementation of systems. As the efficiency and flexibility of the code improves, this is becoming more widely adopted for production systems, in addition to being a popular tool for embedded system design work because of its flexibility and capacity for quick iteration. Embedded Coder creates code efficient enough for use in embedded systems.

With Simulink HDL Coder, also from Math Works, Simulink and State flow can automatically generate synthesizable VHDL and Verilog. Simulink Verification and Validation enables systematic verification and validation of models through modeling style checking, requirements traceability and model coverage analysis.

Simulink Design Verifier uses formal methods to identify design errors like integer overflow, division by zero, dead logic and assertion violation, to generate test vectors and for model checking.

The systematic test tool TPT is used for the formal test process to stimulate Simulink models but also during the development phase where the developer generates inputs to test the system. By the substitution of the Constant and Signal generator blocks of Simulink the stimulation becomes reproducible.

Sim Events adds a library of graphical building blocks for modeling queuing systems to the Simulink environment. It also adds an event-based simulation engine to the time-based simulation engine in Simulink.

The simulation is obtained from various sensors information such as vibration, temperature and humidity are gathered and interfaced through CAN bus and the information has been sent to the host system. The simulation results of vibration sensors and temperature sensors and humidity sensors are shown below :

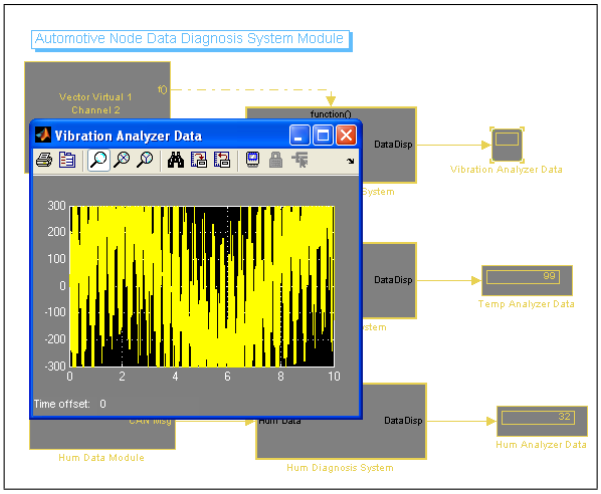

Figure 5: Simulation Result

V. CONCLUSION

In the current implementation of an intelligent remote sensor analyzer the Simulink model captures three sensors information with three sensor nodes using CAN interface. The hardware model will be developed with the help of simulation results by using advanced Micro controllers such as ARM or PIC according to the requirement of the system.

The current auto mobiles electronic architecture is influenced by many different standards. There are ambitions within the automotive industry to standardize the diagnostic software interface in a way that abstract data sources such as vehicle speed are mapped to concrete sensors on the fly.

The big advantage of using vehicle diagnostic infrastructure is that there is virtually no need to modify the car. Therefore, the system is easy to install and remove. Our system employs these benefits and combines them with advanced recording strategies. The given interfaces OBD-II and diagnostic protocols can be used to achieve a high level goal by employing low level tools and data. This adaptable data recorder can be used in various vehicle related contexts and for different purposes, by using one standardized interface and a small amount of processing power and memory. Our situation dependent recording demonstrates a great improvement over existing systems and allows a selective reduction of data quantity while maintaining the adequate data quality.

In the future, stream processing in the vehicle can be enhanced with more advanced operators, which can employ data mining and machine learning techniques on multiple datastreams and will be able to show correlations in case of errors. Use of DSP processor for the diagnosis process can be extended for the application of Obstacle detection and Vibration analysis in the automation field.

References

[1] Hendrick Schweppe, Armin Zimmermann - Member IEEE, and Daniel Grill. Flexible on-board stream processing for automotive sensor data.

[2] H. Schweppe, A. Zimmermann, and D. Grill. Flexible in-vehicle stream processing with distributed automotive control units for engineering and diagnosis. *In Proc. International Symposium on Industrial Embedded Systems, SIES*, pages 74–81, 2012.

[3] Renjun Li, Chu Liu and Feng Luo. A design of automotive CAN bus monitoring system.

[4] Anyu Cheng, Yan Yao, Zhihui Duan, Anjian Zhou and Wei Hong. ECU Loader Design of in-vehicle CAN Network Based on ISO15765.

[5] S. Ilic, J. Katupitiya, and M. Tordon. In-vehicle data logging system for fatigue analysis of drive shaft. *In Proc. International Workshop on Robot Sensing, ROSE*, pages 30–34, 2004.

[6] V. Barabba, C. Huber, F. Cooke, N. Pudar, J. Smith, and M. Paich. A multimethod approach for creating new business models: The General Motors OnStar project. *Interfaces*, 32(1):20–34, 2002.

[7] S. Madden, R. Szewczyk, M. J. Franklin, and D. E. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. *In Proc. WMCSA*, pages 49–58, 2002.

[8] L. Golab and M. T.Ozsu. Issues in data stream management. *SIGMOD*, 32(2):5–14, 2003.

[9] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra,J. Dull, K. Sarkar, M. Klein, M. Vasa, D Handy, M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, Eds. Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring. *In Proc. SDM*, pages 300–311, 2004.

