Distributed Embedded Real-Time Systems for Education - TTcar

C. V. Madritsch
Real-Time Systems and Computer Science
Carinthia Tech Institute
Europastrasse 4, Villach, 9524, Austria
Phone: +43 4242 90500 2127 Fax: +43 4242 90500 2110 E-mail: cm@cti.ac.at

Distributed Embedded Real-Time Systems are becoming more important in the industrial sector. Applications in the field of automotive, aerospace and industrial control require a deep understanding of system design, implementation as well as validation and verification. These requirements need to be fulfilled during the education of future engineers. The Carinthia Tech Institute (CTI), department of Real-Time Systems (RTS), uses real-world applications like a model-sized car to teach the fundamental principles of Real-Time Systems design.

I. INTRODUCTION

Safety critical applications, like Flight Control Systems (FCS) or X-by-Wire Systems (e.g. brake-by-wire) require a new kind of design methodology. The main difference between these systems is that they need to be distributed in order to guarantee independence in the case of a fault [1]. Therefore reliable communication systems are needed to interconnect the distributed parts of the system. This design approach is totally different from the classic centralized systems design approach. Engineers need to develop a holistic view of the application and break it into its distributed parts. The planning of message delivery, task activation and redundancy management needs to be done before the actual system implementation begins.

CTI has developed a model-sized car (see Fig. 1.) to teach this design and implementation methodology to undergraduate students. The model-sized car (Time-Triggered Car – TTcar) consists of up to eleven independent hardware nodes which are interconnected using the fault-tolerant communication system TTP. The whole functionality is distributed among these nodes and all used components have been developed by students. The design flow, based upon tools like Matlab, Simulink and TTPtools, has also been setup by students and is exercised during the Real-Time Systems lecture.

The distribution of this paper is as follows: Chapter II explains the basic functionality of the TTcar. The main focus is on core components like drive and steering. Chapter III describes the mechanical and electrical construction as well as the used algorithms for drive and steering. Chapter IV focuses on the communication system TTP which interconnects the used hardware nodes. First it describes the basic principles of TTP. Finally it describes how the message schedule of the TTcar has been generated. Chapter V shows the design flow which needs to be executed by students. At the beginning is describes the overall system architecture and the partitioning into subsystems, tasks and hardware nodes. This chapter also gives a brief explanation of the used tools. Chapter VI finally concludes this paper.

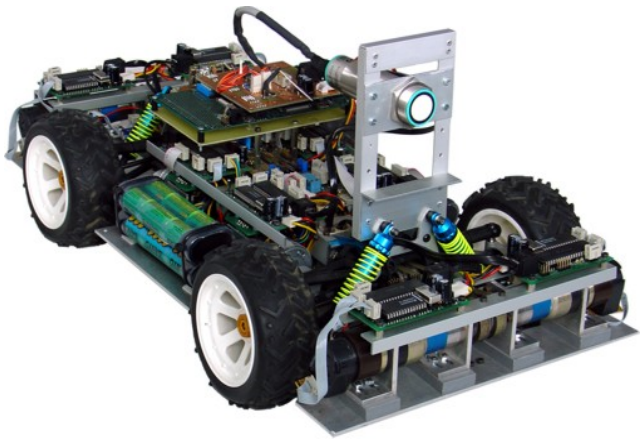

Fig. 1. Time Triggered Car - TTcar

II. ATTRIBUTES AND FUNCTIONS

The reason behind constructing the TTcar was that students in the area of Real-Time Systems are able to work on a real world functional model of a distributed real-time system. It not only gives an example how to plan and develop such systems but also shows the principles of X-by-Wire applications. Industrial partners like Magna Steyr (Graz, Austria) or TTTech (Vienna, Austria) are partly using the same algorithms, concepts and technology as we use in the TTcar.

The basic functionality of the TTcar can be divided into two parts: initialization and startup (1) and normal operation (2). During initialization and startup, all hardware nodes perform a self test. If this self test is successful the communication system starts. If necessary, initialization routines need to be executed in order to align the wheels or to reach predefined setup positions.

During normal operation the TTcar can be controlled either via a stand-alone remote control (RC) unit or a PC using a serial interface connection. The RC unit has also been developed by students and can be used to visualize the TTcar status messages. The PC can be used as a development, test and monitoring platform. The low level functionality in normal operation mode is that each wheel can be driven and steered independently. On a higher level of abstraction, attributes like four-wheel steering, parallel steering, tank steering, velocity dependent steering or geometrically correct drive are available. Those modes can be selected using the RC unit or the development PC and affect the behavior of the TTcar immediately.

III. MECHANICAL AND ELECTRICAL CONSTRUCTION

Some mechanical components (e.g. wheel suspension, wheels) have been available on the model-sized car market. Other parts (e.g. chassis) have been constructed and assembled by our students [2].

The electrical construction includes power supply, sensors and actuators and the hardware nodes. Furthermore a remote control system has been included. The power supply has been realized using two NiMH battery packs. One battery pack supplies the sensors, the RC systems and the hardware nodes. The second battery pack is responsible for the power supply of the actuators. Incremental encoders have been used to measure the steering angle and the RPM of the wheels. An analog potentiometer measures the zero alignment of the wheels and an ultrasonic sensor measures the distance to obstacles. Eight 15W brushless DC-motors (including motor drivers and gearbox unit) have been used to control the steering and drive functionality of the TTcar. The hardware nodes (see Fig. 2. TTcar Hardware Node) consist of a Microchip PIC16F877 microcontroller, an oscillator, the physical layer and interface to the TTP communication system and the interface to the sensors and actuators. Furthermore a PID-based digital motion control unit (LM629) has been used in order to perform position and velocity control.

Fig. 2. TTcar Hardware Node

The remote control system has been realized using standard components for transparent RS232 - RC - RS232 communication. One Hardware node is equipped with an additional RS232 interface. Fig. 3. gives an overview about the overall structure of the TTcar including all nodes, their purpose and the wiring of the TTP communication system.

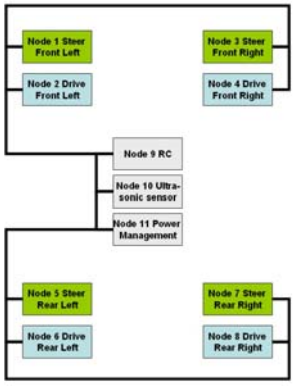

Fig. 3. TTcar Node Structure

The data received at the RC interface of the TTcar includes the steering and drive mode, the desired velocity and the desired steering angle of the car in total. This information is distributed to all nodes. Each node calculates its part (e.g. Node 1 Steer Front Left calculates the new desired value for the Front Left steering motor) and applies it to the corresponding actuator. Furthermore each node sends the actual value of its sensor (e.g. Node 1 Steer Front Left reads the incremental encoder value and converts it into steering angle units) to the RC node. The RC node transmits all sensor information to the PC or a stand-alone RC unit.

IV. TTP COMMUNICATION

A time-triggered system based on TTP provides both fault-tolerance for safety applications, such as brake-by-wire, and flexibility for on-line diagnosis. Furthermore erroneous components within a TTA system cannot affect other components [3]. All protocol tasks in TTP are executed according to an a priori known schedule. The TTP provides the required global base. Bus access is realized using Time Division Multiple Access (TDMA). Therefore nodes cannot interfere with each other. TTP can be implemented as a bus or as a multi-star on different physical layers (e.g. fiber optics or wire). Data transmission rates are not limited by the protocol. TTP has a very high data efficiency of approximately 85%

The TTP message schedule used for the TTcar includes sensor and actuator information, control commands and general status information. One round within the TTcar TTP schedule has a period of 10msec. Fig. 4. is an example message schedule consisting of four nodes and two communication rounds.

Fig. 4. TTP Schedule

The TTP schedule is stored on each communicating node and by using the global time base, each node is able to recognize the send and receive frames on the communication media.

V. DESIGN FLOW

In a distributed system, the overall system functionality (e.g. brake-by-wire) is divided into several subsystems (e.g. pedal-sensing subsystem, brake-force calculation subsystem, and wheel-speed sensing subsystem). To feature composability and fault-tolerance, the interface between the subsystems needs to be specified in both the time- and value-domain.

At system level, a system integrator (e.g. an automotive company) defines the subsystem functions and specifies

the communication interfaces in the time- and value-domain precisely. At the subsystem level, the component supplier retains complete control over all hardware and software design decisions as long as they comply with these interfaces.

This Two-Level Design Approach is supported by a tool-chain (e.g. TTPtools) which allows the seamless development and integration of different subsystems into one distributed system.

In Fig. 5. the upper half reflects the role of the system integrator. The cluster-design is the process of partitioning a system into several independent subsystems and defining the interfaces among each other. The result of the cluster-design process is a cluster-design database. The cluster-design process can be done using the tool TTPplan.

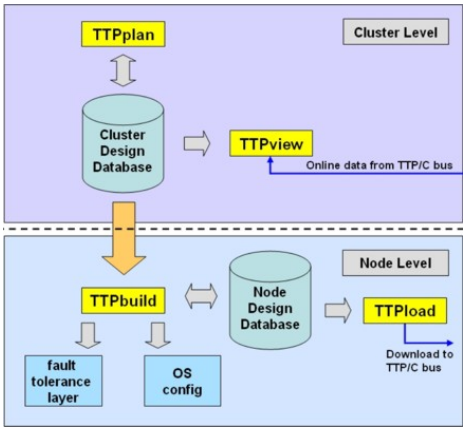

Fig. 5. TTcar Design Flow

The lower half of Fig. 5. represents the role of the component supplier. At the node-design, individual subsystems are partitioned into software-tasks and the corresponding messages between each other are defined. The results of the node-design are the configuration information for the node's real-time operating system (e.g. TTPos), the automatic generated source-code for the fault tolerance layer (e.g. OSEKtime compliant FTcom layer) and a node-design database. This node-design process can be done using the tool TTPbuild.

Due to changing system requirements and specifications, it is occasionally necessary to repeat the development process in order to achieve the system constraints. Also, if the system has to be timely optimized, an iterative approach, which basically repeats the cluster- and node-design, is applicable.

In some industries (e.g. automotive or aerospace) a model based design approach is used to design, develop, and implement embedded systems. Therefore it is necessary that the tools in use can be applied at a very early stage of the system design. Fig. 6. shows, how the TTPtools can be used in the case of a model based design.

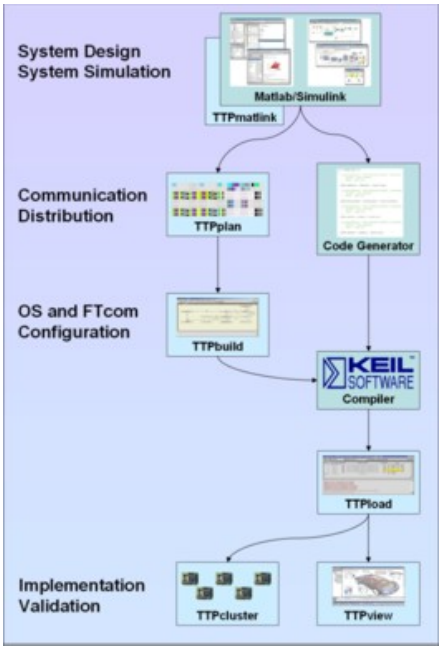

Fig. 6. Model Based Design Flow

TTPmatlink is a Matlab/Simulink blockset, which enables a behavioral simulation of the distributed system in the time- and value-domain. It partitions the system into subsystems and it partitions these subsystems into tasks. Furthermore, it defines the messages between the subsystems and the tasks. The TTcar consists of eight subsystems, 23 tasks, and 11 nodes.

IV. CONCLUSION

The TTcar was designed and constructed three years ago and has been used ever since as a demonstration and development environment for undergraduate students. The good feedback of companies, which employ our graduates, tells us, that the acquired knowledge and skills in the filed of RTS is state of the art and relevant for a variety of R&D engineering positions.

REFERENCES

[1] M. Ley, C. Madritsch, *Distributed Embedded Safety Critical Real-Time Systems, Design and Verification Aspects on the Example of the Time Triggered Architecture*, 39th International Conference on Microelektronics, Devices and Materials MIDEM03, Proceedings p51-62, ISBN 961-91023-1-2, Slovenia, October 2003

[2] C. Madritsch, *Projektorientierte Ausbildung im Bereich der Echtzeitsysteme an der Fachhochschule Technikum Kärnten*, Informationstagung Mikroelektronik (ME 2003), Wien, Österreich, Oktober 2003

[3] C. Madritsch, *Automated Fault-Injection for Distributed Real-Time Systems*, NI-WEEK 2004, Austin TX, USA, 2004