Indian Journal of Science and Technology, *Vol 8(15), 55680, July 2015*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Real-Time Task Scheduling for Distributed Embedded System using MATLAB Toolboxes

P. Sivakumar[1*], B. Vinod[2], R. S. Sandhya Devi[3] and E. R. Jayasakthi Rajkumar[1]

[1] Department of Electrical and Electronics Engineering, PSG College of Technology, Peelamedu, Coimbatore - 641004, Tamil Nadu, India; psk@eee.psgtech.ac.in, rajkumar.rece007@gmail.com
[2]Department of Robotics and Automation Engineering, PSG College of Technology, Peelamedu, Coimbatore - 641004, Tamil Nadu, India; bvin@rae.psgtech.ac.in
[3]Department of Electrical and Electronics Engineering, Kumaraguru College of Technology, Chinnavedampatti, Coimbatore - 641049, Tamil Nadu, India; sandhyasubburaj@gmail.com

Abstract

This paper is gives a clear idea about MATLAB based toolboxes for scheduling various real-time problems and to examine the performance of the system prior to its implementation. Two different MATLAB based toolboxes-TORSCHE (Time Optimization of Resources, SCHEduling) and True Time used offers a collection of data structures that allows the user to analyze various issues in resource sharing and scheduling. Further in real-time environment, these simulators are used to obtain an optimum performance by providing a platform to extend the design methodologies and to meet the requirements as demanded by the application with limited resources. Algorithms are implemented either as MATLAB function with fixed structure or as MATLAB function with middle level language constructs. Two modules are designed and simulated for each toolbox. Using True Time Toolbox, fixed priority scheduling algorithm module and CAN networking of four different kernels are simulated and results are analyzed. Using TORSCHE toolbox timing parameter for each task is analyzed for two different scenarios. Finally, a simulation model is built by using True Time toolbox and TORSCHE in MATLAB platform. The toolboxes are intended mainly for research purpose to handle and control scheduling issues in real-time systems. The result indicates that the simulators aids in improving scheduler accuracy and strengthen the real-time character of the system. Using this approach it is possible to evaluate and modify controller designs and hardware platforms to better understand the performance and timing effects.

Keywords: Deadline, Kernel, Real-Time, Task, True Time, TORSCHE

1. Introduction

1.1 Real-Time System

Real-time systems have rigid timing constraints that must be met. For predictable behavior, to manage the available resources, and to obtain offline guarantee of the application performance various scheduling algorithms are used. While designing a real-time system two tasks are jointly done by two different engineers. One engineer designs the model for the plant to be controlled and another engineer designs the control algorithm for this model. These control algorithms configure the real-time system by priorities and deadline. There are different scheduling algorithms to schedule the tasks in the real-time environment and the role of Real-Time Operating System is of paramount importance for meeting the deadlines, to manage the software and control the system[1]. There are different key terms used in this paper related with scheduling.

1.1.1 Priority

Priority describes the importance given to a task among all other tasks in the system[2].

**Author for correspondence*

1.1.2 Deadline

Deadline is the time before which the task must complete the operation.

1.1.3 Due-date

Due-date is similar to deadline where deadline is the key term used for periodic tasks and due-date is used for aperiodic tasks.

1.1.4 Arrival time

Time at which a task becomes ready for execution; it is also referred as request time or release time.

1.1.5 Computation time

Time necessary to the processor for executing the task without interruption.

1.2 TrueTime Toolbox

TrueTime tool box is used for designing real-time application. This tool is a plug - in for MATLAB and can be used as 1. A pure scheduling simulator. 2. As an experimental test bench for implementing new task-scheduling polices and network protocols and 3. For gathering various execution statistics and scheduling events[3]. Measurements can be logged on to a file and then analyzed in MATLAB. This simulator facilitates co-simulation of controller task execution in real-time kernels, network data communication in dynamic environment. Using this toolbox, performance of the tasks executed in a complex system can be measured such as when the user designs the system with different kernels, communication between the kernels becomes complex and it is difficult to measure the performance of the system. Therefore it is not easy to design this type of system analytically and also not easy to analyze the performance of the system prior to its implementation. In such cases MATLAB/Simulink environment with True Time simulator is extremely useful.

1.3 TORSCHE

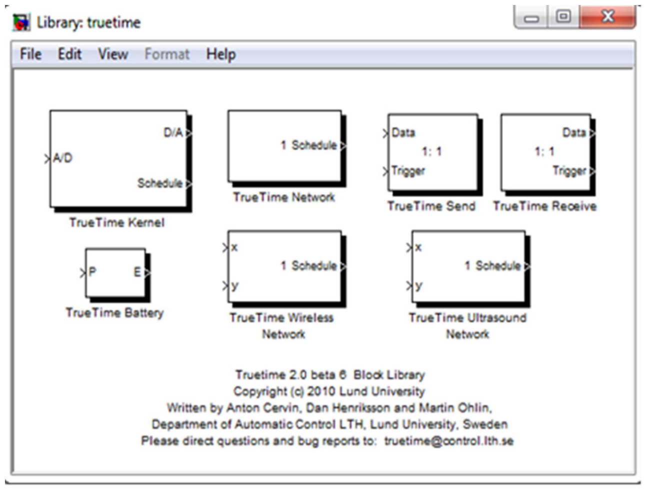
Scheduling is a very popular discipline and its importance is growing at a rapid rate in recent years. TORSCHE (Time Optimization of Resources SCHEduling) is a MATLAB based toolbox used to analyze the performance of different scheduling algorithms prior to its implementation on hardware. It supports complex scheduling algorithm design and verification both online and off-line. The final version of this toolbox covers the following areas of scheduling: scheduling on mono processor/dedicated processor/parallel processor, cyclic scheduling and real-time task scheduling. Graphs and graph based algorithms can be represented and this toolbox provides an interface to TrueTime Simulator. It is mainly used for research purpose.

2. TrueTime Architecture

2.1 TrueTime Toolbox

TrueTime toolbox has a set of files (more than hundreds of file) which are written in C++ and MATLAB. It supports code written in C++ or as M-files. In case of C++ file, a compiler is required to use TrueTime version of C++. No compilation is required for code written in MATLAB as pre-compiled files are available. Each file in the toolbox has a different name. Whenever user writes the code for kernel in C++, call the file which will do a set of operation else call the toolbox file written in C++ and the user compiles the file. Thus compiler is selected depending on the platform used such as Visual Studio C++ for Windows or GCC, G++ for Linux. Normally MEX command is used to compile the code for the software section. When user calls the ttmex command it calls the MEX command implicitly which automatically links the code with the compiler. The hardware module of this system is designed using Simulink. TrueTime library has the following blocks in the Simulink environment and shown in (Figure 1).

TrueTime kernel is the main block where the user writes the code. User saves the code with the same name as given in the init function of the block parameter[4]. Block parameter diagram is given in (Figure 2). Kernel and the



Truetime 2.0 beta 6 Block Library
Copyright (c) 2010 Lund University
Written by Anton Cervin, Dan Henriksson and Martin Ohlin,
Department of Automatic Control LTH, Lund University, Sweden
Please direct questions and bug reports to: truetime@control.lth.se

Figure 1. TrueTime toolbox library.

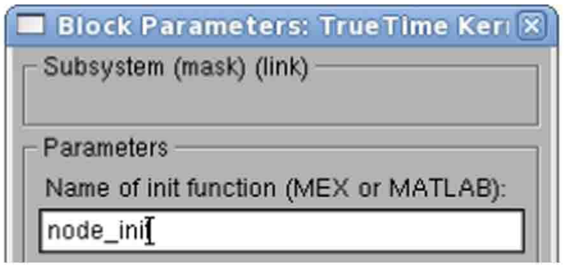

Block Parameters: TrueTime Kerı
Subsystem (mask) (link)
Parameters
Name of init function (MEX or MATLAB):
node_init

Figure 2. TrueTime kernel.

corresponding code are merged. After compilation it produces mexw64 file for 64 bit MATLAB, mexw32 for 32 bit MATLAB in TrueTime 2.0 version. While running the above mentioned files, kernel block gets integrated.

TrueTime kernel is event triggered. Input and output of the kernel are analog signals. Kernel automatically converts the analog signal into digital at the input side and converts digital signal into analog at the output side. Other blocks in the toolbox supports application design.

2.1.1 TrueTime Network Block

TrueTime Network block[5] is used to simulate medium access and packet transmission in a local area network. When a node (here node represents kernel) tries to transmit a message (using the primitive ttSendMsg), a triggering signal is sent through the network block to the corresponding input channel. When the transmission of the message is finished, the network block sends a new triggering signal on the output channel. When the signal is received, the receiving node buffers the message which contains header information of sending and the receiving node, arbitrary user data, length of the message, and real-time attributes such as a priority or a deadline. Six simple network models are supported: CSMA/CD (e.g. Ethernet), CSMA/AMP (e.g. CAN), Round Robin (e.g. Token Bus), FDMA, TDMA (e.g. TTP), and Switched Ethernet.

2.1.2 TrueTime Wireless Network

True Time wireless network is just similar to the wired one. It has x and y input as the true location of the nodes. This wireless network block supports WLAN & ZigBee.

2.1.3 Other Blocks

TrueTime battery is used to give the supply for kernel when in need. TrueTime Ultra Sound Network is used for communicating data as an ultra sound signal. Other standalone network blocks are also available in the toolbox.

2.2 TORSCHE Toolbox

TORSCHE (Time Optimization of Resources, SCHEduling) is a MATLAB based toolbox that has more than hundreds of MATLAB files. Normally this toolbox is used as a scheduling simulator. So from this toolbox user can know the execution time of each task in the task set for different scheduling algorithm. Scheduling algorithm is selected based on the application. Here more than two scheduling algorithms are used 1) Fixed priority scheduling algorithm for periodic task set with an example. 2) Horn algorithm for aperiodic task with an example. Normally files are stored in the toolbox and it is called at the time of program or script execution to do a specific operation defined in the file. Commands are used to create and schedule the task. Some algorithm is compiled by C++ compiler where user needs Visual C++ for Windows platform and g++ for Linux platform. The four steps to solve the scheduling problem are 1. Define task sets, 2. Define the scheduling algorithm. 3. Run the task. 4. Plot the result. To create new task, command with the parameters for the corresponding task is given below,

t1 = task([Name,]ProcTime[,ReleaseTime[,Deadline
[,DueDate[,Weight[,Processor]]]]])[6]

where ProcTime is the computation time of the task. Weight is task priority; ReleaseTime is the arrival time of the task. This toolbox has a graphical interface used to plot the result.

Figure 3 shows the graphical representation of scheduling of the task set which has two different tasks. Release time is represented by the upside arrow, Dead line is

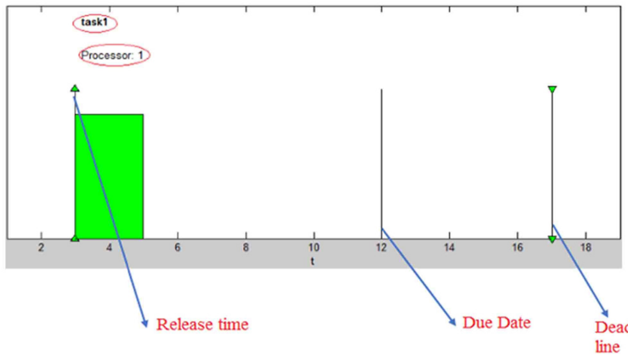


Figure 3. Graphical representation of a task.

indicated by the down side arrow and normal vertical line is used to represent the Due date. Each task's execution is differentiated by different color. First, task command is used to create different tasks; task set command is used to make a set of different tasks. Next command is related with problem, and this is used for aperiodic tasks. The command has the parameters α | β|γ , where α indicates machine environment (that means number of processor) β indicates task and resource characteristics (like preemptive, independent, synchronize) γ indicates optimality criterion to be followed in the scheduling. An example command for the above α | β| γ notation is prob = problem ('P|prec|Cmax'). For creating new periodic tasks ptask is the command used. Syntax of the ptask command is given below

```
pt = ptask([Name,]ProcTime,Period[,ReleaseTime
       [,Deadline[,Duedate[,Weight[,Processor]]]]])
```

For periodic task, problem statement is not necessary and fixed priority scheduling algorithm is used to schedule the tasks. Graphs are used to represent any precedence relationship that exists among the tasks by using this toolbox. Also cyclic tasks can be represented using graph objects. To create simple graphs using graph object the command used is g = graph (B). Similarly graphs can be constructed for various methods using this toolbox.

3. Experimental Evaluation

3.1 TrueTime Simulator

This paper discusses two different modules designed using TrueTime toolbox[7]. In first module, code is written for fixed priority scheduling algorithm which has only one task with a period of 0.5 time units. Thus task gets executed for every 0.5 time units and shown in (Figure 4).

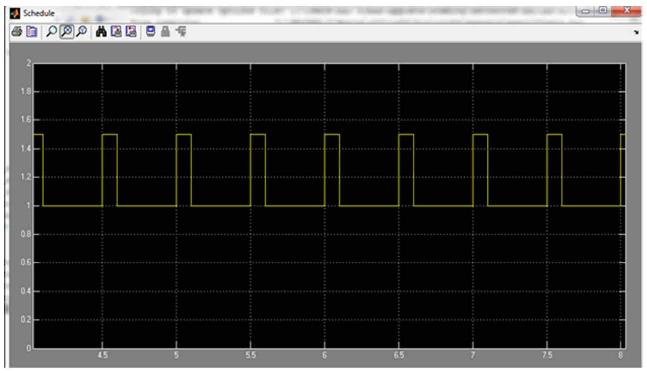
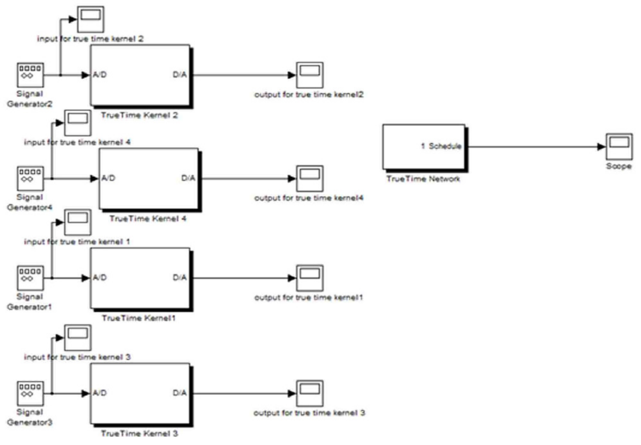


Figure 4. Fixed priority task.

In second module, network block is used for 4 different kernels, where each kernel is connected with independent signal source as shown in (Figure 5). Here, CAN network block is used to broadcasts data to all the kernels[8]. This model is based on CSMA/AMP - Carrier Sense Multiple Access with Arbitration based on Message Priority. When two or more nodes transmit message at a time, collision may happen. Using CSMA/AMP protocol in distributed network, collision is avoided as message is transmitted based on priority.

In this model, two tasks trans() and recv() are initially created. Here tran()s is a periodic task which gets input from the source using the command ttAnalogIn() and sends the message to the network block using the command ttSendMsg (receiver, data, length, priority) where receiver is the ID of the node which receives the



input for true time kernel 2
Signal Generator2
A/D
D/A
TrueTime Kernel 2
output for true time kernel2
input for true time kernel 4
Signal Generator4
TrueTime Kernel 4
output for true time kernel4
input for true time kernel 1
Signal Generator1
TrueTime Kernel1
output for true time kernel1
input for true time kernel 3
Signal Generator3
TrueTime Kernel 3
output for true time kernel 3
1 Schedule
TrueTime Network
Scope

Figure 5. Network with four different kernel.

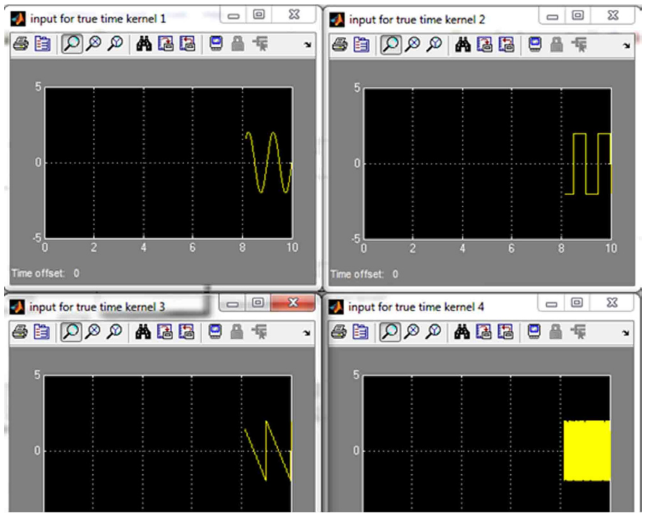


Figure 6. Input for network with four different kernel.

data[9]. Another task recv() is activated by the network handler when there is an incoming message.

The input given to each trans() node (kernel) is given below and shown in (Figure 6).

- Sine signal for kernel 1.
- Square signal for kernel 2.
- Saw tooth signal for kernel 3.
- Random signal for kernel 4.

Each node gives input to the CAN network block and transmits the message to another node using *ttSendmsg()*

Output taken from each *recv()* node (kernel) is shown in (Figure 7):

- Saw tooth wave form at kernel 1.
- Random signal at kernel 2.
- Square wave form at kernel 3.
- Sine wave form at kernel 4.

Scheduling of the tasks for different nodes (kernel) for CAN network is shown in (Figure 8).

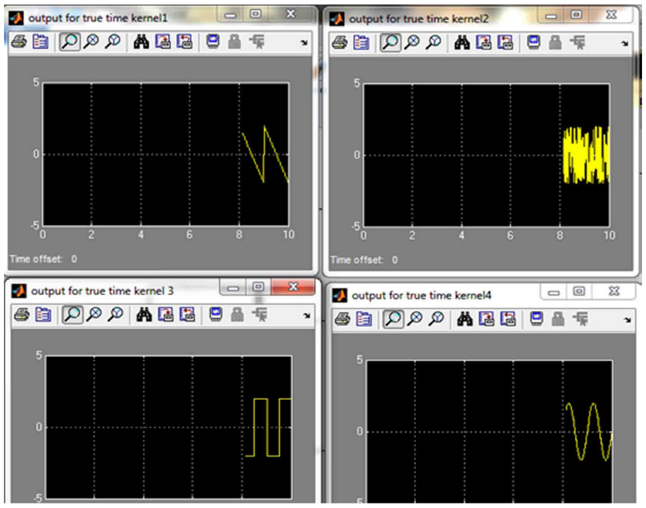

output for true time kernel1
output for true time kernel2
output for true time kernel 3
output for true time kernel4
Time offset: 0

Figure 7. Network output for four differernt kernels.

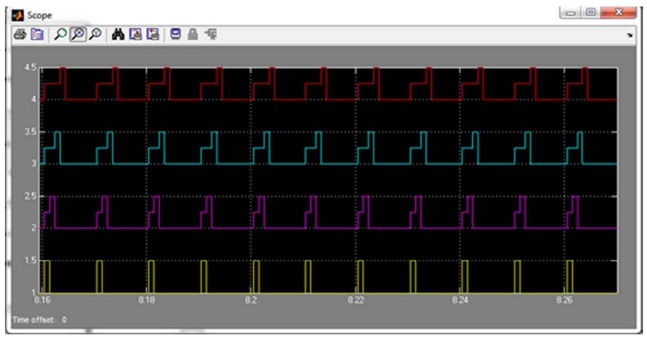


Figure 8. Scheduling task for network.

3.2 TORSCHE Simulator

This paper presents scheduling of tasks effectively for two different cases using TORSCHE toolbox.

3.2.1 Case 1

In a mobile phone manufacturing company, two processors are used to perform a set of task. Time consumed by each task is listed below:

T1: Panel Manufacturing – 1 hour.
T2: Hardware Design – 2 hours.
T3: Software Design – 2 hours.
T4: Porting software onto the hardware – 1 hour; T4 is executed after the completion of tasks T2 & T3.
T5: Hardware assembly with battery – 1 hour; T5 is executed after the completion of tasks T1 to T4.

The user must compute the minimum time required to finish all the tasks. Thus an algorithm is selected based on the optimality criterion required to minimize the maximum computation time with precedence constraints.

The graph shown in (Figure 9.) shows the precedence relationship among the tasks.

To minimize the maximum computation time, tasks are scheduled using List Scheduling (LS) algorithm[10]. It is a heuristic algorithm in which tasks are taken from a pre-specified list. This algorithm is implemented in scheduling toolbox as a function. Our own strategy can also be defined for LS algorithm. When the processor is idle, the first task on the queue utilizes the processor time and it is removed from the list immediately. The availability of a processor means that the task has been released and the processor is idle. Also, precedence/priority constraints are checked and processed smoothly.

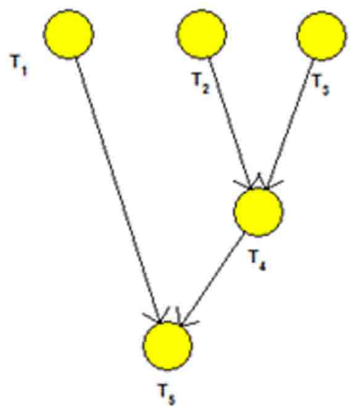


Figure 9. Precedence relationship between tasks.

An optimal scheduling for the above mentioned task set is done in five hours using TORSCHE toolbox and shown in (Figure 10).

3.2.2 Case 2

In automotive application, a system consists of five processors, five different sensors, amplifiers and filters. Each sensor is considered as a task and the time taken by each sensor to process the signal is listed here: Accelerometer - 5 µs, Impact sensor - 4 µs, Wheel Tachometer - 4 µs, Gyroscope - 8 µs, Brake pressure sensor - 7 µs. Output of all the tasks (sensors) is given to the Decision Task and shown in (Figure 11).
When crash occurs, decision task takes 3µs to process the sensor outputs, based on which it initiates the Air Bag (AB) module. The AB module takes 5 µs to release the gas inside the balloon. To complete the entire task set in minimum duration, List Scheduling (LS) algorithm is used and the same is implemented using TORSCHE toolbox.

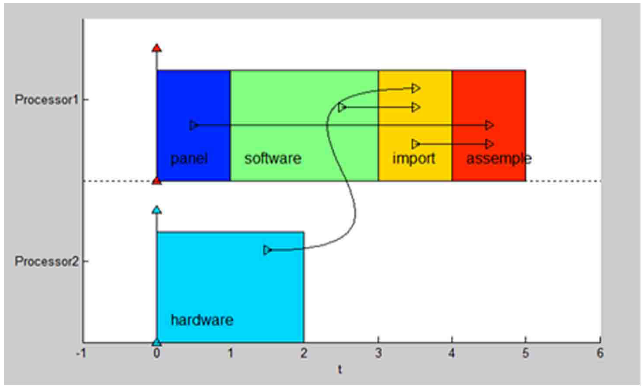


Figure 10. Output from TORSCHE toolbox.

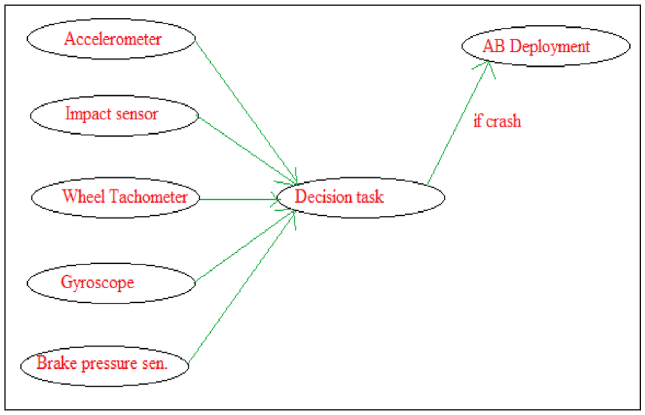


Figure 11. Task set.

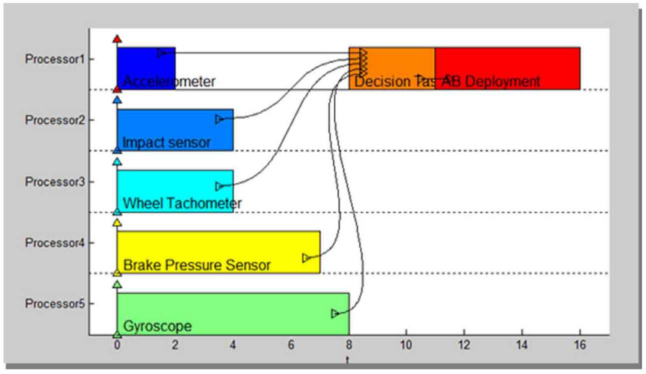


Figure 12. Output from TORSCHE toolbox.

From the TORSCHE toolbox output shown in (Figure 12.) the minimum time required to complete the execution of task set is 16 µs. Thus if the Air Bag deployment is done within 16 µs, the scheduling of tasks in the task set is appropriate. If not, effective scheduling is required for secure driving.

4. Conclusion

This paper presents the implementation of real-time task scheduling using TrueTime, TORSCHE Toolbox for MATLAB. Our approach using these toolboxes for example applications has reduced the computation time which guarantees the execution time of the system. Our work gives a brief idea to researchers about the toolboxes which are compatible for priority processing in real-time systems. Our future work will focus on extending the toolbox support for implementing EDF algorithm.

5. References

1. Liu JWS. Real-Time Systems. Upper Saddle River, NJ, USA: Prentice Hall; 2000.
2. Buttazzo GC. Hard Real-Time Computing Systems. 3rd ed. Kluwer Academic Publishers; 2011.
3. Nicolesu G, Mosterman PJ. Model-based design for embedded systems. Taylor and Francis Group; 2009.
4. Hemingway G, Porter J, Kottenstette N, Nine H, van Buskirk C, Karsai G, Sztipanovits J. Automated synthesis of time-triggered architecture-based truetime models for platform effects simulation and analysis. 21st IEEE International Symposium on Rapid System Prototyping. 2010 Jun 1–7.
5. Yan B, Yehua W. A novel shared-clock hybrid scheduling algorithm based on controller area network. Journal of Computers. Academy Publisher; 2014 Feb; 9(2):373–9.

6. Sucha P. TORSCHE scheduling toolbox for Matlab. Proceeding of IEEE International Conference on Control Applications; 2006 Oct. p. 1181–6.
7. da Penha DO, Weiss G. Integrated timing analysis in the model-driven design of automotive systems. Proceedings of NiM-ALP; 2013.
8. Eker J, Cervin A. A Matlab toolbox for real-time and control systems co-design. Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications; 1999.
9. Cervin A, Henriksson D, Ohlin M. TrueTime 2.0 Reference Manual. Sweden: Lund University; 2010. Available from: http://www.control.lth.se/truetime/
10. Kutil M, Sucha P, Capek R, Hanzalek Z. Optimization and scheduling toolbox, Matlab - modelling, programming and simulations. In: Leite EP, editor. InTech. 2010. Available from: http://www.intechopen.com/books/matlab-modelling-programming-and-simulations/optimization-and-scheduling-toolbox