**CONTROLLER AREA NETWORK**

Mihai Suciu,

*Technical University of Cluj-Napoca*

[mihai.suciu1997@gmail.com](mailto:mihai.suciu1997@gmail.com)

**ABSTRACT**

The main objective of the project is to simulate the behavior of a Controller Area Network and its communication. I chose the Java programming language to solve this simulation. The method used to simulate this communication was the following: I created different objects for every car component: ESP, ABS, Airbag, etc. and call some methods to initialize the signals which will be stored and processed later.

**1. INTRODUCTION**

The main problem we address by realizing this project is establishing which programming language performs better for long and difficult simulations because it is a message-based protocol and we need speed for these signals. This is a big problem of the computer science domain. Of course each programming language has its own advantages and disadvantages but most of them should behave relatively well in making decisions in this type of simulations.

This problem has been addressed before and studies have been made with the same simulations. Development of the CAN bus started in 1983 at Robert Bosch GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers conference in Detroit. The first CAN controller, produced by Intel and Philips, came on the market in 1987. Released in 1991 the Mercedes-Benz W140 was the first production vehicle to feature a CAN-based multiplex wiring system.

The method we will approach is relatively simple. We will create some components, and to simulate them (on a pretty interface) is necessary to call some methods which displays some lights. Every component has a different light.

**2. PROCEDURES AND METHODS**

As we stated, we need to simulate some decisions. When the program will create some signals, we have to decide which signal will be solved first. To know if some signal is important or not, in every class I have an attribute named “isImportant” which is Boolean, and from this attribute I know if the current signal is or is not important. First of all, in the first simulation I have only two signals which appear at once, one is important and the other is not. As simulation, for each signal we have to wait 3 seconds to be solved. The main library used to simulate this time is **javafx.animation.PauseTransition**. In the following paragraphs we will describe algorithm implemented to repair one signal:



*Figure 1 Method to pause the view for 3 seconds.*

**3. RESULTS AND DISCUSSION**

If we have two signals of the same importance, the decision is taken randomly, but if we have the signals of different importance, the one with the greatest importance is taken.

First of all, we will take the ABS and Roof signals. After 3 seconds, the ABS is solved and then the next one is the roof signal.

 



*Figure 2 The result when 2 signals of different importance appear (ABS and Roof signals)*

Another example is when we have two important signals and we need to take a decision.



*Figure 3 There are 3 signals in the same time : ABS,ESP and Airbag*

In this case, we take randomly between ABS and ESP, solve them and after that we can solve the airbag.

 

*Figure 4 Here we have the proof, after the ESP is solved and then the ABS*

Another tested case, when we have one important signal (ESP) and 2 non-important. First we solved the important one, and then is random between seat belt and roof.

 

*Figure 5 We solve the ESP first of all, and then we will take a decision between seat belt and roof.*

Here, we have another decision between two non-important signals. It will be taked randomly.

 

*Figure 6 Here we can see the decision between 2 non important signals*

**4. CONCLUSIONS**

Can is ideally suited in applications requiring a large number of short messages with high reliability in rugged operating environments. Because CAN is message based and not addressed based, it is especially well suited when data is needed by more than one location and system-wide data consistency is mandatory

**5. REFERENCES**

**[1]** – CAN bus

<https://en.wikipedia.org/wiki/CAN_bus>

**[2]** – A simple intro

<https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en>

**[3]** – Automotive CAN transceivers

<https://www.infineon.com/cms/en/product/transceivers/automotive-transceiver/automotive-can-transceivers/>

**[4]** – Automotive Controller Area Network

<https://www.youtube.com/watch?v=YrJn2AyWVBc>

**[5] -** Steve Corrigan, “Introduction to the Controller Area Network, Texas Instruments”, Texas, 2016, pp.16