Abstract - A novel algorithm to decode the Controller Area Network (CAN) frames in an automobile is presented. With didactic intentions and in the context of an industrial collaboration agreement, Artificial Neural Networks (ANN) are used to identify the relations among the diverse frames traveling in the CAN bus and the state of several electronic control modules connected to the network. Early results showed that the solution decodes all the CAN frames of the comfort fieldbus (i.e. dashboard, wipers, lights, doors, windows, seats, mirrors, climate control).

Keywords: Artificial Neural Networks, Controller Area Network.

1 Introduction

The great increase of the electronic systems in the automobile industry has led to the development of many automotive networks. Examples of such networks are Local Interconnect Network (LIN), Vehicle Area Network (VAN), J1850, FlexRay, Controller Area Network and others. CAN was developed by Bosch in the mid-1980s and first introduced in the market in the early 1990s. Currently, it is the most used network in the automobile industry [1]. Inside a car, there could be more than one network in which the different Electronic Control Modules (ECMs) of the vehicle communicate to each other the information regarding the automobile’s operation (i.e. state of the sensors, instructions and commands). The automobile’s manufacturers elaborate their own network codification. The only public information is the one related to the diagnosis of faults such as the standard OBDII.

As part of a collaboration agreement between the Autotronics Laboratory of Tecnológico de Monterrey1, campus Monterrey, and EXXOTEST2 Company, an algorithm to decode the different CAN frames inside a vehicle is proposed.

This paper is organized as follows: Section II presents the state of the art regarding the CAN protocol. Section III describes the experimental set up. Section IV formulates the problem. Section V describes the proposed solution. In section VI, the experimental results are shown. A discussion of the results is presented in Section VII. Finally, Section VIII concludes the paper and proposes future work.

2 State of the Art

The Controller Area Network (CAN) is an in – vehicle network created by Bosch for multiplexing communication between ECMs, reducing the wiring in the automobile as well as its weight. Its low cost, robustness (i.e. high tolerance to faults) and different mechanisms for error detection make CAN the preferred network by the automotive industry [2].

Regarding the CAN protocol, several research works of different topics have been made. In [3], the authors present a vehicle data management system (VDMS) for providing different information to vehicle manufacturers and suppliers about the different states of the variables of the vehicle. The system is implemented using a wireless link to CAN – TCP node in the vehicle. With the obtained data, remote diagnosis can be done as well as calculation of the lifetime of the components.

A description of a data reduction algorithm for CAN protocol is described in [4] in order to improve data exchange rates. On the other hand, [5] presents a general probabilistic schedulability analysis technique for CAN. The purpose of this algorithm is to calculate the effect on the response time of messages of the random network faults. The importance of this works strives in the need of accurate predictions of failure in safety - critical applications such as the Antilock Braking System (ABS).

Due to the characteristics of CAN, it is not used only in the automotive industry, but also in other fields, such as robotics and aviation. In [6], a work is presented about the use of CAN in the multi – processor system of robotic manipulators.

3 Experimental Setup

The Autotronics Laboratory at Tecnológico de Monterrey, campus Monterrey, is equipped with a multiplexed CAN X3 pedagogic scale model from EXXOTEST™, Figure 1. This work station is a training unit with real components of the Peugeot 807 and integrates three different network types: CAN, LIN and VAN.

---

1 www.mty.itesm.mx
2 www.exxotest.com
The CAN system is composed of twelve modules: Built – In System Interface (BSI), Built – In Supply Module (BSM), air conditioning, passenger door, driver door, front lights, back lights, dashboard, radio, AFIL (French Abbreviation for System Lane Departure Warning System), tow module and alarm. These modules are interconnected in three CAN networks: an intersystem network whose baud rate is 500 kbit/s (i.e. CAN high speed); a chassis network and a comfort one, both with a baud rate of 125 kbit/s (i.e. CAN low speed) [7].

In the intersystem network, the diagnosis module, the motor status module and the steering wheel sensor are connected. On the other hand, the comfort network is composed of the dashboard, the radio system, air conditioning, AFIL, driver and passenger door. Finally, the airbag, the alarm and the lights switchboard system compose the chassis network, Figure 2.

Communication with the network is done using the EXXOTEST® USB-MUX-4C2L module (which allows interfacing a PC to the CAN bus) and the MUXDLL dynamic link library, also provided by EXXOTEST®, Figure 3.

![Fig 1. EXXOTEST® scale model. Top picture shows the detailed modules; while bottom photo shows the experimental equipment.](image1)

![Fig 2. Block diagram of the multiplexed pedagogic scale model CAN X3 from EXXOTEST™. It is composed of three networks: one CAN high speed and two CAN low speed.](image2)

![Fig 3. Communication system.](image3)
4 Experimental Setup

The automobile’s manufacturer does not make public the translation of all the CAN frames’ traveling through the different networks of a specific car. Using a CAN frames analyzer, information traveling through the network can be seen; however, it is impossible to understand the meaning of the data, Figure 4.

An algorithm for interpreting the information contained in the possible greatest number of CAN frames inside an automobile is proposed. In other words, it is desired to know what does a specific frame refers to.

Fig. 4. Information provided by a CAN frames analyzer. As one can see, without the manufacturer’s database, the data shown are just meaningless data.

5 Problem Solution

When the system is in a steady state (i.e., parameters are not changing; radio remains on, door is closed, etc.), the data field of every CAN frame changes in a structured way. This means that there are some data fields whose values remain the same, but there are others whose bytes change following a certain pattern.

5.1 Generation of the database:

Based on the fact aforementioned, the procedure proposed to decode the CAN frames is depicted in Figure 5. First, the user tells the application what variable will be identified. After this, the user begins to change the state of the parameter and the CAN frames decoder starts to run. At the end, the application prints the identifier(s) and byte(s) of the data field that was found to be associated with the specified variable.

As an example, consider the case in which the user specifies the running lights as the parameter to be found. Then, he turns on and then off those lights (i.e., changes the state of the parameter). The result is the identifier(s) and byte(s) of the data field associated with that parameter.

Fig. 5. Procedure for decoding the CAN frames in a vehicle. In this process, the user gets the identifier(s) and corresponding byte(s) related to a specific parameter of an automobile.

5.2 CAN frames decoder:

The main part of the general procedure is the CAN frames decoder, Figure 6. The key idea is to predict the incoming frame, in order to compare it to the real value and providing an error of prediction as outcome. This result is compared with a threshold; based on that, the algorithm makes a decision.

Fig. 6. Block diagram of the CAN frames decoder.

When an incoming frame has a closed relation with a parameter whose state is being changed by the user, the

---

3 The information of a standard CAN frame is determined by an identifier of 11 bits and a variable size data field of 8 bytes long maximum.
error_m, as shown in Figure 6, will be large enough to detect that relation. The decision made at the final step states if the incoming frame refers or not to the variable specified by the user; if that so, it also marks which byte(s) of that frame is(are) associated with that parameter. The mentioned threshold in the block diagram is related to the CAN frames predictor as will be described below.

5.3 CAN frames predictor:

An important part of the proposed solution is to predict the data field of every CAN frame in a specific network. For this to be accomplished, we use, for every byte of the data field, an Artificial Neural Network (ANN) with a feedforward structure of 5-5-2-1, Figure 7. The five entries are the identifier of the CAN frame received and the four last samples of the specific byte to be predicted. The output is the estimated value of the byte received. Due to the fact that the data field of a CAN frame can be 8 bytes long at maximum, there are 8 ANN’s. Of course, there will be some frames where not all the outputs from the predictor will be valid (i.e. those frames with less than 8 bytes at the data field). In those cases, only the useful values are taken.

The bipolar sigmoid function is used as the activation function. Due to the fact that this function has a range of (-1, 1), it is necessary to make a normalization of the targets for training purposes, as well as a denormalization of the outputs. These two operations are made taking into account that a byte has a range from 0 to 255.

\[ f(x) = \frac{2}{1 + \exp(-x)} - 1 \]

Fig. 7. Structure of the ANN (5-5-2-1) used for predicting the data field of the CAN frames.

The training method used for each neural network is the backpropagation with incremental style; the weights and biases of the network are updated each time an input is presented to the ANN. The learning rate has a fix value of 0.05. Considering the nature of this kind of systems, the general concept of epoch for training does not exist. Thus, during the training procedure, the backpropagation learning is made online every time a CAN frame is sent over the bus of communication until a stop condition is reached. The stop condition could be manual or some kind of error criterion. We have used a moving average error. It is important to remark that the training has to be done when the system is in a steady state (i.e. every parameter is not changing; radio remains on, door is closed, etc.). This is because the solution is based on detecting the changes in the system at the moment the user modifies the state of a specific variable.

The error_m is the sum of all the error_mn, Figure 7:

\[ error_m = \sum_{n=0}^{k} error_{mn} \]  

where k refers to the data field size. This error_m is the one that is compared to a threshold whose value is determined by the experimentally obtained training error.

6 Experimental Results:

6.1 Application

The solution was implemented in the software LabWindows CVI 7.0 from National Instruments™. The developed application has different features, among them: it can monitor up to 3 CAN networks of different speeds (i.e. CAN frames analyzer). It also has the option of training the ANNs described in the previous section; an online training. The user can also monitor the error_m for every frame of a specific network for validation of training purposes. The CAN frames decoder also features database functionality, which enable users to associate states or values to every variable.

6.2 Experimentation

The experimentation was performed using the equipment described in section II. Due to the requirements of the algorithm (i.e. steady state condition, capability of the user to change manually the state of the parameters), the experiments were implemented for the low speed networks (i.e. chassis and comfort network, 125 kbit/s).

The first step was to train the ANNs for each one of those networks. Then, the identification of every variable was done according to the procedure described in the subsection 5.1. The obtained results showed a complete decodification of all the parameters whose state condition could be changed by the user. The validation method was using the feature of the developed software that shows the state of every variable contained in a specific database according to the incoming frames.
As an example, consider the case of the running lights. Table 1 shows the identifiers and corresponding bytes found by the algorithm as related with that variable.

**TABLE 1: IDENTIFIERS AND CORRESPONDING BYTES FOUND FOR RUNNING LIGHTS.**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Byte(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x36</td>
<td>4</td>
</tr>
<tr>
<td>0x46</td>
<td>1, 5 and 8</td>
</tr>
<tr>
<td>0x82</td>
<td>1</td>
</tr>
<tr>
<td>0x94</td>
<td>1</td>
</tr>
</tbody>
</table>

The followed procedure for the identification of the frames related to the running lights implies two steps: steady state condition at the start of the procedure and changing the state of the lights during the method. Taking into account these facts, the tracking of the $\text{error}_m$ of each of the identifiers listed in Table 1, is depicted in Figure 8-11. In these figures, the y-axis corresponds to the $n$-th sample of the $\text{error}_m$ obtained at the moment of the identification of the running lights. Figures 8, 9 and 11 have the same scale at the x-axis, which is not the same case for Figure 10. That is because the periodicity of the identifier 0x82 is lower than the one for the identifiers 0x36, 0x46 and 0x94.

The four graphs have the same shape: the pulses shown in each one correspond to the time when the user changes the state of the variable (i.e. lights on to lights off, and vice versa), whereas the other moments correspond to the steady state condition. This means that error of prediction is relatively low when the parameter is on steady state condition, and increases when it changes. It can be noticed that the $\text{error}_m$ for the identifier 0x46 is relatively higher at the transient moment compared to the others; the reason is because it involves three bytes whereas the others involve only one as is shown in Table 1.

It is interesting to notice that it does not matter if the training of the ANNs took place during the steady state condition when the running lights were off or on. In other words, the ANNs learned the structure of the system, not a specific state. That is the explanation about why the $\text{error}_m$ is relatively low at any steady state. The algorithm proposed to identify a specific variable of the vehicle is based on that behavior.

---

![Fig. 8](image1.png)  Error $\text{error}_m$ of identifier 0x36 during the identification procedure of the frames related to the running lights.

![Fig. 9](image2.png)  Error $\text{error}_m$ of identifier 0x46 during the identification procedure of the frames related to the running lights.

![Fig. 10](image3.png)  Error $\text{error}_m$ of identifier 0x82 during the identification procedure of the frames related to the running lights.
Fig. 11. Error in of identifier 0x94 during the identification procedure of the frames related to the running lights.

7 Discussion

The obtained results show a high reliance of the proposed algorithm for the parameters that are able to be changed by the user. This clearly restricts the method developed for only the CAN frames of the comfort fieldbus (i.e. dashboard, wipers, lights, doors, windows, seats, mirrors, climate control).

It is easy to get confused and think about a solution based only in comparing the actual incoming frame with the last samples; this solution would not work because the system is not static. Although some frames remain with the same value and only change according to their state, there are some frames that are continuously changing. This does not change the assumption of the steady state condition of the system discussed previously, because that condition implies that the frames change in a structured way.

Finally and due to the nature of our solution, the relation between the frames and the parameters can be identified easily using a CAN frames analyzer and looking carefully at the data fields that really “change” with the state of the parameter. The algorithm proposed makes this operation in an automatic way, faster and error free.

8 Conclusions and Future Work

An algorithm to identify the relation between the parameters of an automobile and CAN frames traveling in its network has been proposed. The proposal is restricted to independent parameters and those whose state can be changed directly by the user. This certainly indicates a future work for the parameters that do not fulfill those conditions. Also, interesting topics such as modeling CAN behavior for fault diagnosis and control are closely related to this research.

Acknowledgment

The authors gratefully acknowledge the EXXOTEST® Company for the laboratory facilities and support material. In particular, the authors would like to thank Mr. Stéphane Sorlin, CEO of the company just mentioned, and Mr. Pierre Clément for defining the relevance of this work.

References


