IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 7, 2020, accepted June 26, 2020, date of publication July 13, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008872

Statistical Based Algorithm for Reducing Residual Error in Embedded Systems Implemented Using the Controller Area Network

REZA ALAEI[1], PAYMAN MOALLEM[1], AND ALI BOHLOOLI[2], (Member, IEEE)
[1]Department of Electrical Engineering, Faculty of Engineering, University of Isfahan, Isfahan 8174673441, Iran
[2]Faculty of Computer Engineering, University of Isfahan, Isfahan 8174673441, Iran

Corresponding author: Payman Moallem (p_moallem@eng.ui.ac.ir)

ABSTRACT In this paper, a new approach is introduced to reduce bit stuffing and consequently residual error probability in the controller area network (CAN). The proposed method is based on XOR masking. Unlike the XOR method, the proposed approach does not use a fixed mask for all IDs. Using statistical parameters of data, a proper mask for each CAN ID is generated and applied to the messages before transmitting. The performance of the method to reduce bit stuffing and residual error probability has been evaluated by considering a real data set. Results show that the method can significantly reduce bit stuffing and residual error probability. A comparison has been also conducted with previously reported methods. The results show the superiority of the SMC method in reducing residual error without payload and data transfer rate reduction.

INDEX TERMS Controller area network (CAN), stuff bit (SB), residual error.

I. INTRODUCTION

Controller Area Network (CAN) is a well-known serial communication protocol originally developed by Bosch in the mid-1980s for multiplexing communication between electronic control units (ECUs) in automobiles [1]. Although CAN was firstly introduced for automotive industries, it has received wide attention in industrial automation including military, aviation, electronics, factories, and other real-time control applications [2]. CAN is a suitable protocol for communicating a large number of short messages with high reliability between a controller and other devices without a host computer. However, CAN has also several weaknesses like the absence of security functions [3], [4]. Additionally, it can induce some problems in control systems because of its time-varying delay (like stimulating torsional oscillations in the integrated motor-transmission system) [5]–[7]. But its appealing features and low implementation costs cause wide opportunity in the development of networks and controller systems [8]–[15].

Real-time distributed systems increasingly use the controller area network for transmitting real-time information. Because of potentially high network noise in such systems, reliably detecting errors could become important to avoid the spreading of corrupted data. corrupted data in some systems can lead to disaster. For instance, signals in automotive communication networks often include safety-relevant information. However, electromagnetic interference can disturb transported data in-vehicle networks. In [16], it is stated that bit error rate in the order of magnitude of 10^{-3} is possible when a vehicle is close to a high-power radio frequency transmitter or a high-voltage power supply. The stated content indicates that having a powerful mechanism for error detection is vital in such systems. The cyclic redundancy check (CRC) is one of the most effective ways which can be applied easily to any kind of network for error detection. CAN specification states that the CRC in each message will detect all burst errors up to 15 bits, as well as all errors with 5 or fewer disturbed bits [1].

The minimum number of single-bit errors that are necessary to cause residual error is called CRC's Hamming Distance, which in theory is equal to or greater than six [17], [18]. Although using CRC guarantees a relatively highly reliable network but it is not perfect. Every error pattern that turns a valid frame into another valid frame may be undetected. This is particularly a problem in a safety-critical system. therefore, it is important to design systems so that the chance of corrupted data being mistaken for valid data is small.

The associate editor coordinating the review of this manuscript and approving it for publication was A. Taufiq Asyhari.

Bit stuffing that is used in CAN to maintain synchronization may decrease the efficiency of CRC error detection so that the CRC's Hamming Distance decreases from six to two. This error detection deterioration was first noted by Funk while investigating the HDLC protocol [19].

In [20], the corrupted messages that escape from detection are classified and the possibility of residual error in CAN networks is analyzed. In [21], assuming a two-state symmetric binary channel model for the physical transmission medium, the probability of errors that remained undetectable at receivers is analyzed and the contributions of different error mechanisms to the residual error probability are identified. The investigation of Both [20], [21] are purely analytical and not verified by simulation or a case study. In [22], it is stated that the interference of bit stuffing and error detection mechanism can be solved by applying methods such as software bit stuffing (SBS) [23], inversion bit stuffing mechanism (IBSM) [24], third-bit complement (TBC) [25], 8B9B encoding [26], eight-to-eleven modulation (EEM) [27], and Nolte C method [28]. These methods can prevent the occurrence of stuffed bits in the data field completely by using suitable encoding schemes and consequently decrease residual error probability. but these approaches cause message payload reduction and increase CPU processing time.

XOR method is another approach that can reduce the probability of bit stuffing without payload reduction and consequently reduce the probability of residual error. In this method transmitted frames are XOR-ed with the bit-pattern 101010. . . [28]. Applying 101010. . . mask can remove identical consecutive bits (111. . . or 000. . .) and prevent bit stuffing (XOR a bit with 1 flips the bit and XOR with 0 remains the bit unchanged). But in some cases, this method may lead to an increment of residual error. In fact, if adjacent bits have a lot of changes relative to each other, this method may cause stuffed bits, and consequently, the residual error probability will be increased. MXOR [29] is another method based on the XOR approach. In this method, each byte of data is compared with some specific patterns and one of the 0xAA or 0×55 masks is applied to each byte of data before transmitting. Similar to the XOR method, due to the use of fixed masks, MXOR may also increase bit stuffing probability.

In this study, a new method which is named as statistical mask calculation (SMC) is introduced. In this method, it has been attempted to perform statistical computations to identify parts of each message that adjacent bits have a lot of changes relative to each other and avoid applying 101010. . . mask to them. In other words, the decision to flip a bit or not is based on the probability of variations in the amount of neighboring bits. FIGURE 1 compares the performance of this method rather than using fixed 101010. . . mask in the reduction of stuffed bits. As shown in FIGURE 1(a), in the least significant bits in which adjacent bits have a lot of changes relative to each other, using a fixed 101010. . . mask creates stuffed bits, whereas, SMC mask reduces the number of stuffed bits down to zero.

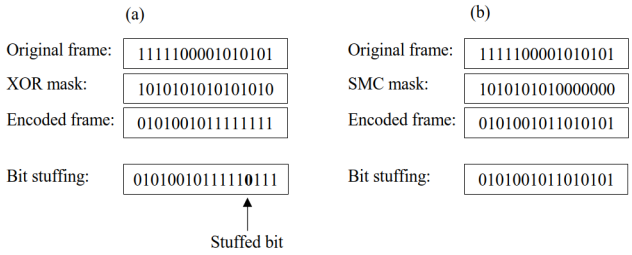

FIGURE 1. The encoding process for (a) XOR and (b) SMC masking.

It is worth to mention that the SMC mask is only applied to the data field. In order to omit bit stuffing in the header part of the CAN frame, the presented method in [30] is employed. By excluding the identifiers that lead to bit-stuffing, one can make sure that there will be no stuffed bits in the header part of the messages.

The paper is organized as follows: Section 2 introduces the CAN message format, section 3 describes different kinds of errors. The analyses of errors and related equations are presented in section 4. The proposed method is introduced in section 5. In section 6, in order to validate the proposed method, a small case study is presented. The results of applying the method on the case study are shown in section 7. This paper is concluded with section 8.

II. CAN MESSAGE FORMAT

The frame format of a CAN message is classified into the standard 2.0A with an 11-bit identifier and the extended 2.0B with a 29-bit identifier. Different fields of CAN frame are shown in FIGURE 2. As shown in the FIGURE 2 data frame consists of start-of-frame (SOF), arbitration field, control field, data field, acknowledgment (ACK) field, and end-of-frame (EOF). A SOF bit marks the beginning of a data frame. It is represented by one dominant bit (value $= 0$). The identifier indicates the priority of the message. Remote transmission request (RTR) serves to differentiate a remote frame (a frame requesting the transmission of a specific identifier) from a data frame (a frame containing data for transmission). A dominant RTR bit indicates a data frame while a recessive RTR bit indicates a remote frame. The distinction between standard 2.0 A and extended 2.0 B frames is made by using identifier extension bit (IDE), which is transmitted as a dominant bit in standard 2.A frame, and as a recessive bit in extended 2.B. The last four bits of the control

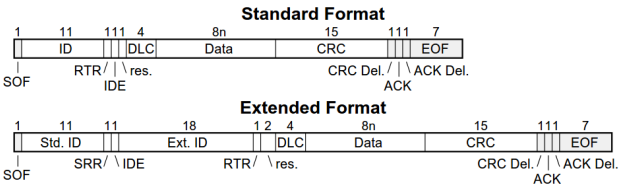

FIGURE 2. Standard and extended format of CAN message.

field is called the data length code (DLC). Its value represents the length of the data field. Data field contains up to 8 bytes of data to be transmitted. The CRC field consists of a 15-bit CRC code for error detection and a recessively transmitted delimiter bit. The acknowledge field is used to acknowledge the receipt of a valid CAN frame. Each node that receives the frame without finding an error, transmits a dominant level in the ACK field. The ACK field has one delimiter bit. The EOF consists of a sequence of 7 recessive bits [1].

III. ERROR IN THE CONTROLLER AREA NETWORK

Errors in the controller area network can be categorized into four main groups:

- **Don't care error**: Errors in the reserved bit r0 of the control field and the last bit of the end of the frame are considered as don't care and are not included in the calculation of residual error. In fact, in the latter case, there is no time left for signaling the error.
- **CRC errors**: Errors that only affect the CRC field are detectable and must be neglected.
- **Errors that influence the frame length**: If RTR, IDE, or DLC fields are affected by bit errors, the message length will be changed and the difference is more than one byte, which will lead to reliable detection of errors. As shown in FIGURE 3 (a), an error may also affect a bit and transform a subsequent bit into a stuffed bit. Alternatively, an error can transform a stuffed bit into a data bit (FIGURE 3 (b)). These kinds of errors can change frame length. Thus, they can be detected by CRC or frame format. Therefore, this group of errors is not included in the analysis of residual errors.
- **Errors that do not affect the message length**: The corruption of ID and data fields may escape detection and must be considered in the calculation of residual errors. This state can be realized in two ways. As shown in FIGURE 3 (c), an error may affect data bits without creation/deletion of a stuffed bit or any other side effects. There is also a chance that a pair of bit errors delete a stuffed bit and create another. In this case, the original message length is maintained but error multiplication occurs. Depending on the data bits between the stuffed bit deleting and creating, the number of errors in the de-stuffed message may be much greater than the number of bit errors on the transmission channel. In fact, errors in stuffed messages may cause effective shifting of data patterns. This shifting leads to multiple bit errors being fed to the CRC. This effect can be seen in FIGURE 3 (d). In this example, a bit error changes the value of one of the five ones in the sequence. Therefore, the receiver assumes that no bit stuffing has occurred and reads the bit at zero (that is added as stuffing) as a normal data bit. If the second bit- error creates a sequence of five zeroes, the receiver incorrectly interprets the following bit at 1 as a stuffed bit. Thus, the message length is maintained. These two errors lead to bit sequence shifting and consequently a large number of errors. The number of errors may be too large to be detected by the CRC code. As demonstrated in FIGURE 3 (d), two errors in the transmitted message are transformed into more than six ones. This causes false acceptance of a corrupted message or in other words residual error.

In the above descriptions, it is clarified that bit stuffing may degrade the error detection capability of the CRC. In another word, bit stuffing may reduce the CRC's Hamming Distance from 6 to 2.

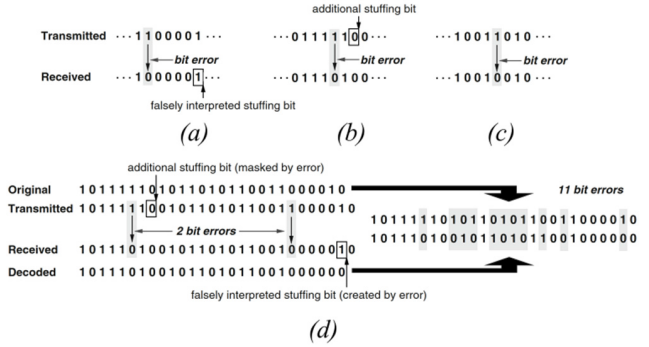

FIGURE 3. (a) Bit error converts data bit into a stuffed bit (b) Bit error converts stuffed bit to data bit. (c) Bit error doesn't affect stuffed bits (d) Two errors lead to bit sequence shifting and consequently a large number of errors.

IV. RESIDUAL ERROR ANALYSIS

This section concentrates on the analysis of errors that affect ID and Data. In the following analysis, cases with at least one corrupted bit in the ID field is called Masquerade error. In this case, there may be other corrupted bits in the data field, too. Cases with at least one corrupted bit in the data field and no corrupted bit in the ID field are known as Corruption error.

In all analyses, the following assumptions have been made: bit errors are independent of each other, single bit-error in a frame will always be detected, and each device is in error active mode [19]–[21].

Total residual error probability can be written as Eq. (1):

$$P_{Total} = \sum_{k=a}^{b} \left(\frac{N_{ID} + N_{Data} + N_{CRC}}{k} \right) p^k \cdot (1-p)^{N_{ID}+N_{Data}+N_{CRC}-k} \quad (1)$$

N_{ID}, N_{Data}, and N_{CRC} denote the length of ID, data, and CRC field respectively. Bit error probability is shown by p, while a and b define the range of bit-error number, where $b \geq a$ and $a \geq$ *Hamming Distance*. Probability of Masquerade, Corruption, and CRC error can be written as Eq. (2) to Eq. (3):

$$P_{MASQ} = \sum_{k=a}^{b} \left(\frac{N_{ID} + N_{Data} + N_{CRC}}{k} \right) .p^k \cdot (1-p)^{N_{ID}+N_{Data}+N_{CRC}-k} - (1-p)^{N_{ID}}$$

$$\cdot \left[\sum_{k=a}^{b} \left(\frac{N_{Data}+N_{CRC}}{k} \right) .p^k . (1-p)^{N_{Data}+N_{CRC}-k} \right] \quad (2)$$

$$P_{CORR} = (1-p)^{N_{ID}} \cdot \left[\sum_{k=a}^{b} \left(\frac{N_{Data}+N_{CRC}}{k} \right) .p^k . (1-p)^{N_{Data}+N_{CRC}-k} \right] - (1-p)^{N_{ID}+N_{Data}} \cdot \left[\sum_{k=a}^{b} \left(\frac{N_{CRC}}{k} \right) .p^k . (1-p)^{N_{CRC}-k} \right] \quad (3)$$

$$P_{CRC} = (1-p)^{N_{ID}+N_{Data}} \cdot \left[\sum_{k=a}^{b} \left(\frac{N_{CRC}}{k} \right) .p^k . (1-p)^{N_{CRC}-k} \right] \quad (4)$$

The relative proportion of Masquerade and Corruption errors are calculated by Eq. (5) and Eq. (6).

$$r_{MASQ} = {P_{MASQ}}/{P_{MASQ} + P_{CORR} + P_{CRC}} \quad (5)$$

$$r_{CORR} = {P_{CORR}}/{P_{MASQ} + P_{CORR} + P_{CRC}} \quad (6)$$

Probability of errors per hour R_{re} can be computed by Eq. (7) [31]:

$$R_{re} = \frac{3600.s}{h} \cdot \left(\frac{1}{T} M_{InUse} \right) \cdot \left(r_{CORR}.P_{CORR} + P_{MasEff} \right) .100 \quad (7)$$

In which h and s are hour and second, respectively. T is transmission interval and it is assumed to be equal to the common value of 10 ms. M_{inUse} is the number of different messages. P_{RES}, and P_{MasEff} are calculated using Eq. (8), and Eq.(9), respectively.

$$P_{RES} = P_{ME} \cdot 4.7 \cdot 10^{-11} \quad (8)$$

$$P_{MasEff} = \left(P_{RES}.r_{MASQ} \right) .{(M_{InUse}-1)}/{\left(2^{N_{ID}}-1\right)} \quad (9)$$

P_{ME} is message error probability and it is calculated by Eq. (10).

$$P_{ME} = 1 - (1-p)^{N_{BitsInMsg}} \quad (10)$$

In Eq. (10) $N_{BitsInMsg}$ is the number of bits in a message and p is the bit error probability.

V. METHOD

As mentioned before, bit stuffing increases the probability of residual error. On the other hand, using methods such as 8B9B in order to prevent bit stuffing leads to message payload reduction. Another approach to decrease bit stuffing without reducing the message payload is to apply the 101010. . . mask to the message before transmitting it. The weakness of this method is that it always uses a fixed 101010. . . mask. This leads to an increment of bit stuffing probability in parts of the message that the amount of adjacent bits have a lot of changes relative to each other. In this paper, the probability of changes of the amount of adjacent bits is calculated using statistical calculations. Then parts of the message where the amount of adjacent bits have a lot of changes relative to each other are detected and it is prevented to apply the 101010. . . mask to these parts. In fact, the decision to apply the 101010. . . mask to message segments is based on statistical calculations on messages. In the following, the SMC algorithm is described based on the presented block diagram in FIGURE 4.

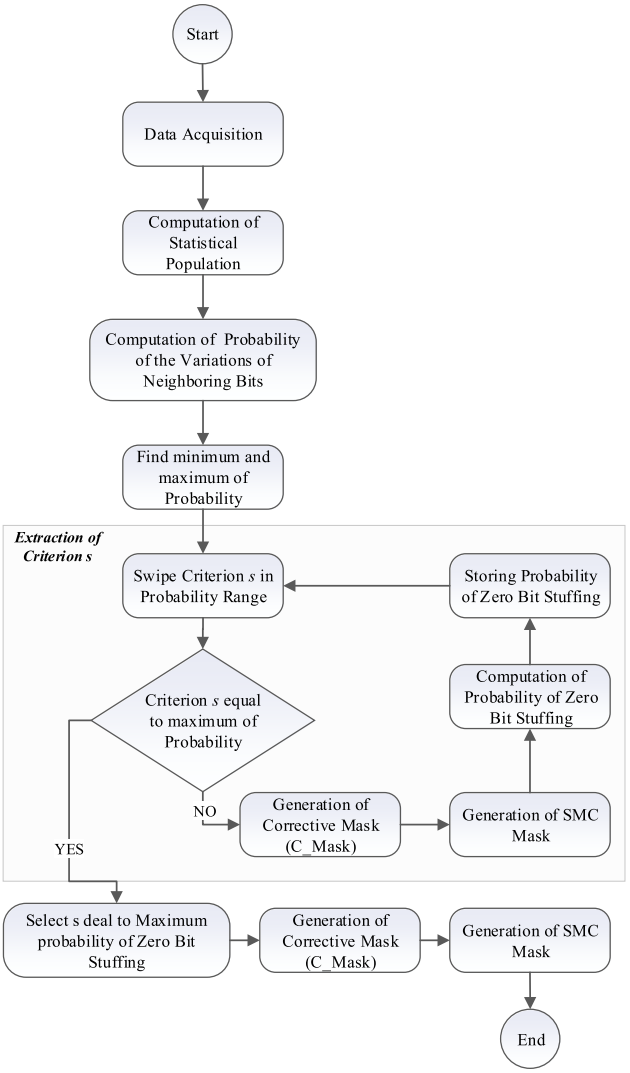

FIGURE 4. SMC masking encoding process block diagram.

Given that the proposed SMC algorithm operates based on statistical computation on data, the data field of transmitted messages on CAN bus for each ID must be firstly collected and saved as a statistical population. In other words, in each application firstly it is necessary to run a representative experiment. For instance, considering a drone, for which the detailed specifications are described in section 6, as the system, all subsystems of the drone (Gyro, Accelerometer, etc.) should be tested in all possible states and their data should be stored. For instance, according to the allowed voltage range of battery (30 to 24V), data of the utilized battery should be

stored while the drone is operating. In another experiment, the altitude of the drone must be varied between its specified minimum and maximum values, and the pressure data must be stored. For a case study with six different IDs and 25000 messages corresponding to each ID, there are six statistical populations with 25000 members. In order to calculate the probability of neighboring bit changes for each statistical population, each message data field should be scanned bit by bit. The frequency of changes in the neighboring bit content is considered as f(i). The probability of neighboring bit changes is P(i) which can be calculated by dividing f(i) by the number of bits. For a case study with six different IDs and 25000 messages, the variation of bit(i) rather than bit(i + 1) for all 25000 messages of each statistical population should be counted and divided by the number of data bits. P(i) can be calculated by Algorithm 1. To find a criterion, named parameter *s*, in order to measure changes in the bit content, firstly the minimum and maximum of P(i) for each ID should be found. Then parameter *s* is swept from minimum to maximum value of P(i) and the probability of zero-bit stuffing is calculated for each s. The parameter *s* related to the maximum probability of zero-bit stuffing is selected as the criterion of bit content variation. In the next step, a corrective mask should be generated to prevent applying 101010... mask to parts of the message that the amount of adjacent bits have notable alternations relative to each other.

Algorithm 1 P(i) Calculation Procedure

Inputs : N, M
M: *Number of packets*
N: *Length of data field of packet in bits*
Outputs: $P(i)$
$P(i)$: *Probability of neighboring bit changes* (*i is bit number of data field)*
f(i): *Frequency of neighboring bit changes* (*i is bit number of data field) initial value set to zero.*
Foreach *j in Number of packets* **do**
  For *i in Length of data field of packet in bits* **do**
    If (bit(i) **xor** bit($i+1$)) *equal to* 1 **then**
      $f(i) = f(i) + 1$
    End
  End
End
For *i in Length of data field of packet in bits* **do**
  $P(i) = f(i) / (M * N)$
End

Considering the parameter *s*, the data field is divided into regions with a high rate of changes (P(i) > s) and sections with a low rate of changes (P(i) < s). If P(i) is larger than *s*, C-mask(i) will be equal to 0, whereas if P(i) is smaller than *s*, C-mask(i) will be equal to 1. By applying bitwise AND operation on corrective mask and 101010... mask the final upgraded mask (SMC mask) will be generated. Finally, bitwise XOR must be applied to the SMC mask and the data field of messages. The SMC mask calculation procedure is shown in Algorithm 2.

Algorithm 2 SMC Mask Calculation Procedure

Inputs : $P(i), N$
$P(i)$: *Probability of neighboring bit changes* (*i is bit number)*
N: *Length of data field of packet in bits*
Outputs: *SMC mask and C-mask*
MinP: *Minimum value of* $P(i)$
MaxP: *Maximum value of* $P(i)$
Foreach *s in MinP to MaxP, 100 steps* **do**
  Foreach *i in Length of data field of packet in bits* **do**
    If $P(i)$ *less or equal to s* **then**
      $C\text{-}mask(i) = 1$
    Else
      $C\text{-}mask(i) = 0$
    End
  End
  $SMC = C\text{-}mask$ **and** (1010101...)
  Encoded data = SMC **xor** data field
  Find probability of zero bit stuffing in Encoded data
End
The s that leads to maximum of probability of zero bit stuffing is the best choice to calculate SMC mask

VI. CASE STUDY

In order to validate the proposed method in real applications, a case study is investigated. Samples are taken from a set of messages which are transmitted through CAN network between six different subsystems of a prototype drone. The six subsystems are battery ('Battery'), accelerometer ('Acc'), gyroscope ('Gyro'), magnetometer ('Mag'), pressure and temperature sensors ('Pressure & Temp'), and tilt angle sensor ('Angle') of the drone. All messages are periodic. FIGURE 5 shows this architecture and Error! Reference source not found. specifics the requirements of the messages to be scheduled.

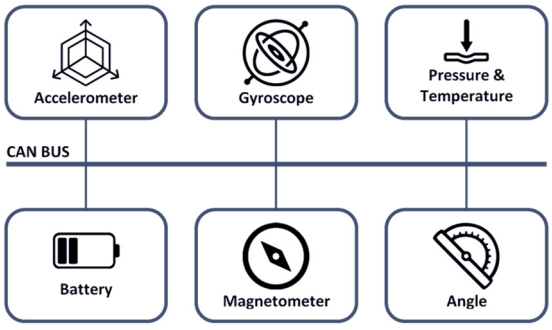

FIGURE 5. Subsystems of the prototype drone.

The test trajectory of the drone is shown in FIGURE 6. The drone begins to move from the starting point, it goes to point

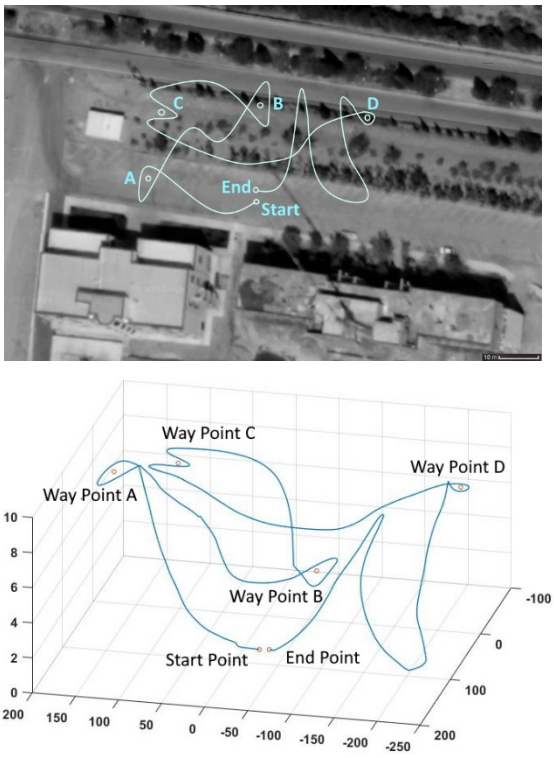

FIGURE 6. 2D and 3D trajectory of the drone.

A at the altitude of 10 meters, and then reaches point B by lowering its altitude to 2 meters. After that, it passes through the point C at the height of 7 meters and, with a specified maneuver, reaches the point D at the same height. Finally, it reaches the end of the path through a relatively complex route. For a case study with six different IDs and 25000 messages corresponding to each ID, there are six statistical populations with 25000 members.

The probability of neighboring bit changes, P(i), versus bit number for each ID is shown in FIGURE 7. The length of the data field of each ID is in accordance with TABLE 1. The most important remarks related to this figure will be described based on accelerator and battery data:

TABLE 1. ID and budget of messages.

Packet ID	Description	Size (Byte)	Period (ms)
276	Accelerometer	6(2Bytes for each direction. z, y, and x respectively)	10
260	Gyroscope	6(2Bytes for each direction. z, y, and x respectively)	10
86	Magnetometer	6(2Bytes for each direction. z, y, and x respectively)	10
133	Angel	6(2Bytes for each direction. z, y, and x respectively)	10
258	Battery	2	1000
36	Pressure & Temp	4(2Bytes for each one)	100

- **Accelerator data investigation**:
  The P(i) diagram of the accelerator data in Z direction has a significant maximum in bit number 9 compared

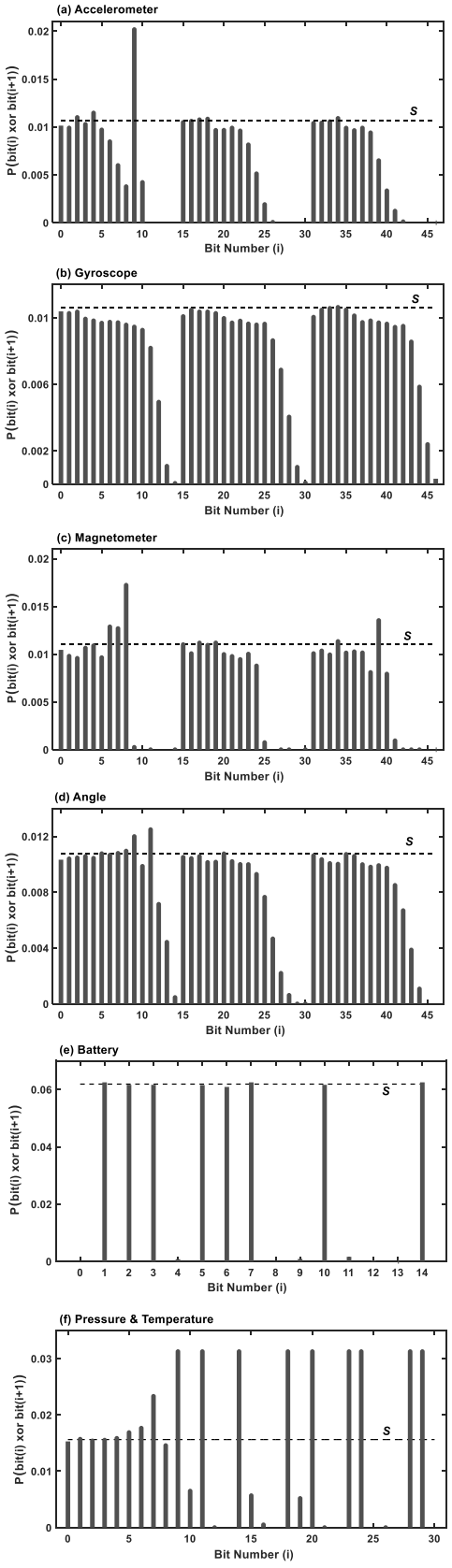

FIGURE 7. Probability of neighboring bit changes (P(bit(i) xor bit(i + 1))) versus the bit number (i).

to P(i) of the accelerator data of X and Y directions. The reason for this observation can be explored by calculation of relative frequency of accelerator amount in X, Y, and Z direction. As shown in FIGURE 8 the maximum frequency of accelerator amount in X and Y directions is related to zero acceleration while it is related to 1000 mg in the Z direction. In other words, the mode parameter related to accelerator data in X, Y, and Z directions is 0, 0, and 1000 mg respectively. On the other hand, the range of changes around the mode in all directions is 600mg. Thus, based on the binary format of 600 (0000001001011000) in the data related to X and Y directions, the changes of neighboring bits relative to each other happen almost up to bit 9. While in the case of data related to Z direction considering the binary format of the mode, and the range of changes (1000: 0000001111101000, 600: 0000001001011000 respectively) in most cases the amount of bit 9 and bit 10 is 1 and 0, respectively. Therefore, the maximum of P(i) is in bit number 9.

- **Investigation on the voltage of the battery**:
  The battery voltage is almost constant and equal to 30900 mv with the binary format of 1111000101101000. Therefore, changes in bit-content almost are in bit 1 to 2, 2 to 3, 3 to 4, 5 to 6, 6 to 7, 10 to 11, and 14 to 15. As shown in FIGURE 7 (e) the maximums of P(i) are matched with the mentioned bits.

As described in the previous section the optimum value of parameter s is selected by sweeping parameter s between minimum and maximum of P(i) and finding the maximum of zero-bit stuffing probability. The optimum value of the corrective mask, parameter s, and SMC mask related to each ID is calculated and shown in TABLE 2.

TABLE 2. Parameter s, C-Mask, and SMC Mask related to each ID.

ID	S	C-Mask	SMC-Mask
Acc	0.0107	FFFBFFF9FDEB	AAAAAAA8A8AA
Ang	0.0107	FFFFFFEFF45F	AAAAAAAAA00A
Batt	0.062	BF7D	AA28
Gyro	0.0106	FFFBFFFFFFFF	AAAAAAAAAAAA
Mag	0.011	FF7BFFF5FE3F	AA2AAAA0AA2A
Pressure Temperature	0.016	CE6BB50D	8A2AA008

VII. RESULTS

In this section, at first, the effect of applying the SMC algorithm on bit stuffing reduction is investigated. Then the performance of the method on the reduction of residual error probability is evaluated.

A. THE EFFECT OF THE SMC ALGORITHM ON THE BIT STUFFING PROBABILITY

After calculating the SMC mask associated with each ID and applying each mask to its related ID, the drone is operated in the same way similar to FIGURE 6, and the ability of the proposed method to reduce bit stuffing probability is evaluated. The message of each ID is transformed using a bitwise XOR operation on the data with its own SMC mask.

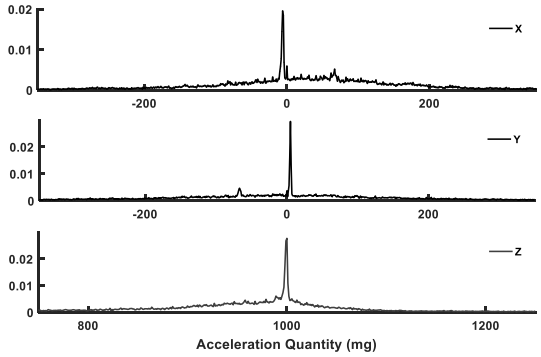

FIGURE 8. Relative frequency of acceleration in X, Y, and Z directions.

As shown in FIGURE 9, in the data related to accelerometer, gyroscope, magnetometer, and tilt angle sensor, the probability of bit stuffing is significantly reduced. Regarding battery and pressure & temperature data, no significant reduction is expected. It is due to the low probability of bit stuffing in battery and pressure & temperature raw data (see TABLE 3).

FIGURE 9 also shows the improvement of the SMC method compared to XOR and MXOR methods. As seen, the SMC has better performance. It is due to its ability to select the mask for each ID intelligently according to the statistical parameters of data. While the XOR method always uses the fix 1010... mask. MXOR method also selects one of the two 0×55 or 0xAA masks for each byte of data by comparing each byte with fixed patterns.

As previously noted, if there is more than one stuffed bit in the data field, the possibility of residual error increases. Therefore, the merit of the proposed method can be majorly evaluated by the reduction of the probability of more than one stuffed bit in the data field after applying the SMC mask to the raw data. TABLE 3 shows that the probability of more than one stuffed bit is averagely reduced by 54% after applying the SMC mask to raw data. TABLE 3 also compares the SMC method with XOR and MXOR methods. The obtained results clearly show better performance of the SMC method compared to the original XOR and MXOR methods. The 100% improvement is regarded to the case in which the bit stuffing probability after applying XOR or MXOR approaches is very small while after applying the SMC method decreases to zero.

B. THE EFFECT OF THE SMC ALGORITHM ON RESIDUAL ERROR PROBABILITY

Residual error probability per hour versus bit error probability for each ID is demonstrated in FIGURE 10. All calculations are based on section 4 and TABLE 3. As expected for the data regarded to battery and pressure & temperature the residual error probability after applying the SMC mask is not changed while it is dropped significantly in other cases. The efficiency

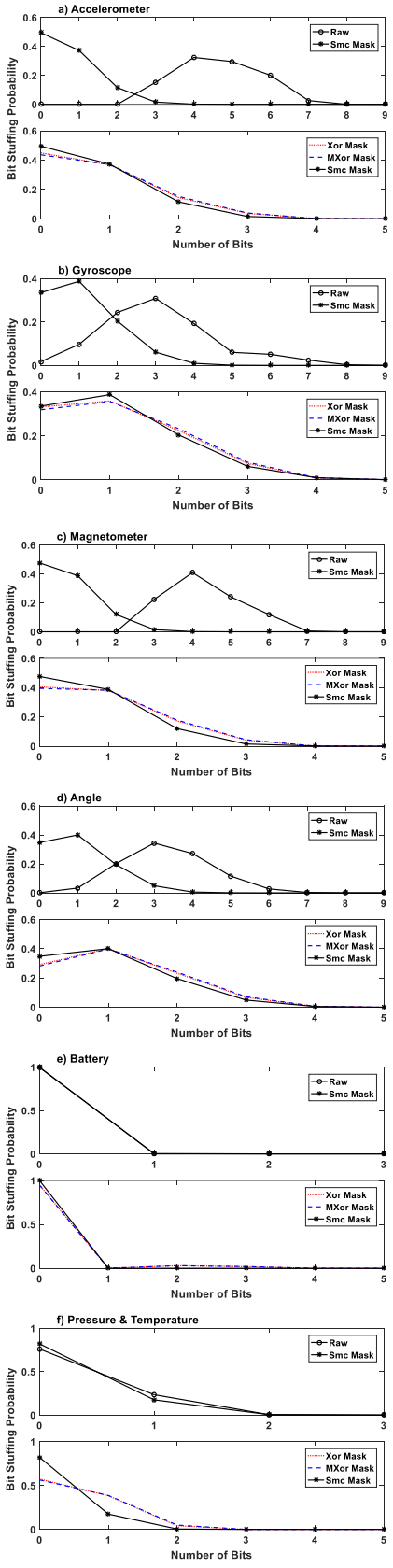

FIGURE 9. The probability that a certain number of bits is stuffed.

TABLE 3. Reduction of the probability of more than one stuffed bit after applying the SMC mask related to each ID in comparison to previous approaches.

ID	MXOR	XOR	RAW
Acc	27.75	31.71	88.5
Ang	18.26	21.40	79
Mag	36.26	38.84	87
Gyro	10.45	14.75	70
Batt	100	100	0
Pressure Temperature	89.48	90.97	0

of the SMC method to reduce the residual error in comparison to XOR and MXOR procedures is shown in FIGURE 10. As shown in FIGURE 10, SMC can reduce residual error probability more than XOR and MXOR. Although the reduction is small, it can be very effective. In fact, error probability, although small, becomes an issue when magnified by massive scale of products. Similarly, reduction of error probability, even in small amounts, is valuable. It is worth mentioning that according to the battery and pressure & temperature, applying XOR and MXOR method increase residual error while the SMC method leaves it unchanged. This result can be a strong emphasis on the efficiency of the SMC method in comparison to XOR and MXOR.

It is clear that methods such as 8B9B, etc., perform better in reducing the residual error compared to the SMC method due to complete elimination of bit stuffing. But they reduce the payload (bandwidth) and increase the CPU processing time for encoding and decoding. TABLE 4 shows the maximum payload of each method. As seen, these methods reduce payload. CPU processing time for encoding and decoding of the SMC method and also some previous methods is measured and presented in TABLE 5. For CPU processing time measurement, a hardware platform based on STM32f407 (ARM cortex M4) with a 168 MHz system clock and ARM-GCC compiler is used. As seen in TABLE 5, the CPU time is decreased significantly in the SMC method rather than other methods.

TABLE 4. Maximum payload of some previous methods.

Method	Maximum payload
Proposed SMC method	8
XOR	8
MXOR	8
Nolte C	6
TBC	6
8B9B	7
EEM	5

This is because of the fact that only online processing in the SMC method is a simple XOR operation. In contrast, other methods have more online processing. For instance, the decoding process of Nolte C involves checking each byte

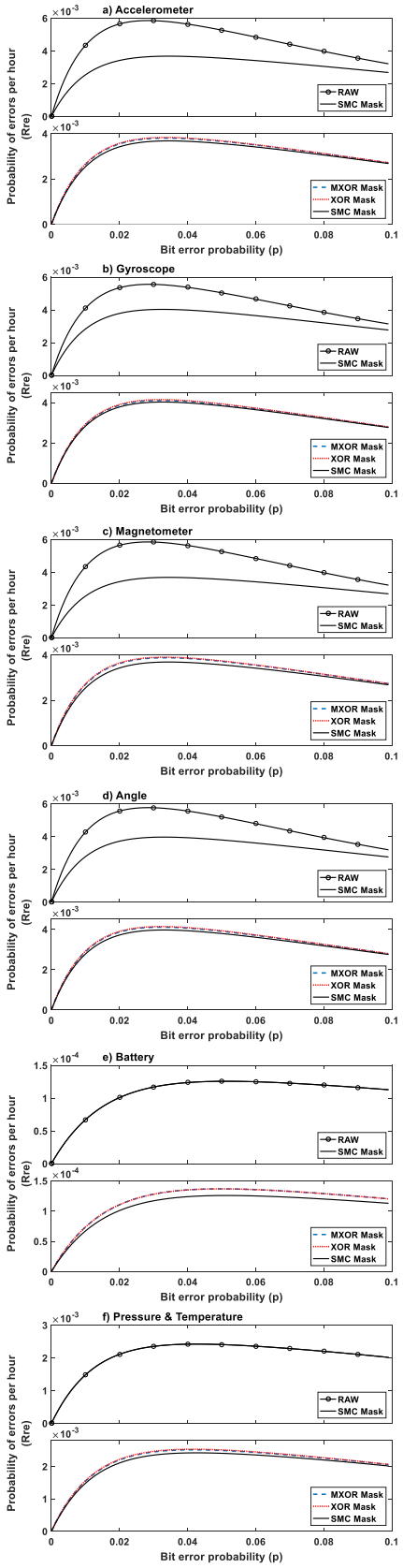

FIGURE 10. Residual error probability per hour (Rre) versus bit error probability (p).

TABLE 5. Encoding and Decoding overhead time.

Method	Encoding time (µs)	Decoding time(µs)
Proposed SMC method	0.11	0.11
XOR	0.11	0.11
MXOR	3.5	3.5
Nolte C	17.4	1.48
TBC	13.08	11.84
8B9B	1.99	2.10
EEM	1.86	1.98

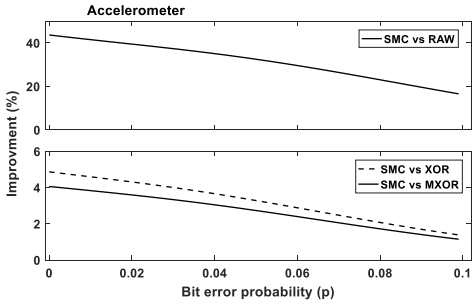

FIGURE 11. The efficiency of the SMC method in the reduction of residual error probability versus bit error probability for accelerometer data.

(bit by bit) to find hardware bit stuffing which takes a lot of time.

FIGURE 11 shows the efficiency of the SMC method in the reduction of residual error probability versus bit error probability for accelerometer data. The data of other subsystems have the same trend. As shown in FIGURE 11, the proposed algorithm has a better performance at lower bit error probability value. According to FIGURE 10, decreasing the efficiency of the proposed method by increasing the bit error probability compared to other methods is expected. As can be seen in FIGURE 10, in all methods, the value of the residual error probability approaches to a constant value as the bit error probability increases. Therefore, by increasing the bit error probability, the efficiency of all methods is reduced.

VIII. CONCLUSION

In this paper, a new method is proposed to reduce bit stuffing and consequently residual error probability in the controller area network (CAN). In this method, using statistical parameters of data, a suitable mask is generated for each CAN ID and applied to the messages before transmitting. The performance of the proposed method on the number of stuffed bits and the residual error probability is evaluated by considering a real data set. The results show a significant reduction of bit stuffing, and residual error probability using the proposed approach. Comparing the proposed algorithm with previous methods shows that the algorithm is superior in reducing bit stuffing and residual error probability without payload reduction and increment of encoding/ decoding processing time.

REFERENCES

[1] R. Bosch, *CAN Specification Version 2.0*. Gerlingen, Germany: Rober Bousch GmbH Postfach, 1991, Art. no. 300240.
[2] W. L. Ng, "Review of researches in controller area networks evolution and applications," *Proc. Asia–Pacific Adv. Netw.*, vol. 30, 2010, pp. 14–21.
[3] H. Sun, S. Y. Lee, K. Joo, H. Jin, and D. H. Lee, "Catch ID if you CAN: Dynamic ID virtualization mechanism for the controller area network," *IEEE Access*, vol. 7, pp. 158237–158249, 2019.
[4] R. Buttigieg, M. Farrugia, and C. Meli, "Security issues in controller area networks in automobiles," in *Proc. 18th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. (STA)*, Dec. 2017, pp. 92–98.
[5] W. Li, W. Zhu, X. Zhu, and J. Guo, "Robust oscillation suppression control of electrified powertrain system considering mechanical-electric-network effects," *IEEE Access*, vol. 8, pp. 56441–56451, 2020.
[6] W. Li, W. Zhu, X. Zhu, Y. Xu, J. Yang, and Z. Li, "Torsional oscillations control of integrated motor-transmission system over controller area network," *IEEE Access*, vol. 8, pp. 4397–4407, 2020.
[7] X. Zhu, H. Zhang, D. Cao, and Z. Fang, "Robust control of integrated motor-transmission powertrain system over controller area network for automotive applications," *Mech. Syst. Signal Process.*, vols. 58–59, pp. 15–28, Jun. 2015.
[8] N. Çenesiz and M. Esin, "Controller area network (CAN) for computer integrated manufacturing systems," *J. Intell. Manuf.*, vol. 15, no. 4, pp. 481–489, 2004.
[9] C. A. J. G. Ferreira, P. E. Cruvinel, E. A. G. Penaloza, V. A. Oliveira, and H. V. Mercaldi, "A hydraulic-pump speed controller in agricultural sprayers based on the automation and use of the control area network (CAN) bus," in *Proc. IEEE 12th Int. Conf. Semantic Comput. (ICSC)*, Jan. 2018, pp. 358–362.
[10] L.-B. Fredriksson, "Controller area networks and the protocol can for machine control systems," *Mechatronics*, vol. 4, no. 2, pp. 159–172, Mar. 1994.
[11] Y. Liu, X. Zhu, H. Zhang, and M. Basin, "Improved robust speed tracking controller design for an integrated motor-transmission powertrain system over controller area network," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 3, pp. 1404–1414, Jun. 2018.
[12] D. Mannisto and M. Dawson, "An overview of controller area network (CAN) technology," in *Raport Instytutowy,Raport Instytutowy*. Evanston, IL, USA: Machine Bus Corporation, Nov. 2003.
[13] S. Misbahuddin and N. Al-Holou, "Efficient data communication techniques for controller area network (CAN) protocol," in *Proc. ACS/IEEE Int. Conf. Comput. Syst. Appl., Book Abstr.*, Jul. 2003, p. 22.
[14] J. Luis Sevillano, A. Pascual, G. Jiménez, and A. Civit-Balcells, "Analysis of channel utilization for controller area networks," *Comput. Commun.*, vol. 21, no. 16, pp. 1446–1451, Oct. 1998.
[15] F. G. Tinetti, F. L. Romero, and A. D. Pérez, "CAN bus experiments of real-time communications," in *Proc. Argentine Congr. Comput. Sci. (CACIC)*, vol. 790. Springer, 2017.
[16] B. Gaujal and N. Navet, "Fault confinement mechanisms on CAN: Analysis and improvements," *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 1103–1113, May 2005.
[17] U. Kiencke, "Error handling strategies for automotive networks," *SAE Trans.*, vol. 97, pp. 638–647, Jan. 1988.
[18] J. Unruh, *Error Detection Capabilities of the CAN Protocol*. Stuttgart, Germany: Robert Bosch GmbH, 1989.
[19] G. Funk, "Message error detecting properties of HDLC protocols," *IEEE Trans. Commun.*, vol. COM-30, no. 1, pp. 252–257, Jan. 1982.
[20] J. Unruh, H. J. Mathony, and K. H. Kaiser, "Error detection analysis of automotive communication protocols," *SAE Trans.*, pp. 976–985, Feb. 1990.
[21] J. Charzinski, "Performance of the error detection mechanisms in CAN," in *Proc. 1st Int. CAN Conf.*, Sep. 1994, pp. 1–10.
[22] G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "Effect of jitter-reducing encoders on CAN error detection mechanisms," in *Proc. 10th IEEE Workshop Factory Commun. Syst. (WFCS)*, May 2014, pp. 1–10.
[23] M. Nahas and M. J. M. Pont Short, "Reducing message-length variations in resource-constrained embedded systems implemented using the controller area network (CAN) protocol," *J. Syst. Archit.*, vol. 55, no. 5, pp. 344–354, 2009.
[24] M. M. Hassan, "Bit stuffing techniques analysis and a novel bit stuffing algorithm for controller area network (CAN)," *Int. J. Comput. Syst.*, vol. 2, pp. 80–87, Mar. 2015.
[25] M. M. Hassan, "Third bit complement (TBC) mechanism to reduce bit stuffing jitter in controller area network (CAN)," *Int. J. Adv. Res. Elect., Electron. Instrum. Eng.*, vol. 4, no. 5, pp. 1–7, May 2015.
[26] G. Cena, I. Cibrario Bertolotti, and A. Valenzano, "An efficient fixed-length encoding scheme for CAN," in *Proc. 9th IEEE Int. Workshop Factory Commun. Syst.*, May 2012, pp. 265–274.
[27] T. R. Jena, A. K. Swain, and K. Mahapatra, "A novel bit stuffing technique for controller area network (CAN) protocol," in *Proc. Int. Conf. Adv. Energy Convers. Technol. (ICAECT)*, Jan. 2014, pp. 113–117.
[28] M. Nahas and M. J. Pont, "Using XOR operations to reduce variations in the transmission time of CAN messages: A pilot study," in *Proc. 2nd UK Embedded Forum*, Oct. 2005, pp. 4–17.
[29] S. K. Kabilesh and B. V. Kumar, "Design and simulation of modified selective XOR algorithm for payload attrition in CAN," in *Proc. 3rd Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2016, pp. 1–6.
[30] T. Nolte, H. Hansson, and C. Norstrom, "Minimizing CAN response-time jitter by message manipulation," in *Proc. 8th IEEE Real-Time Embedded Technol. Appl. Symp.*, Sep. 2002, pp. 197–206.
[31] J. Alanen and M. T. H. Malm, *Safety of Digital Communications in Machines*. Espoo, Finland: VTT, 2004.

REZA ALAEI was born in Shiraz, Iran. He received the B.Eng. degree in electrical and electronics engineering from the Shiraz University of Technology, in 2008, and the M.Eng. degree in electrical and electronics engineering from Shiraz University, Iran, in 2011. He is currently pursuing the Ph.D. degree with Isfahan University, Iran. His research interests include embedded systems hardware and software architectures.

PAYMAN MOALLEM was born in Tehran, Iran, in 1970. He received the B.Sc. degree in electronics engineering from the Isfahan University of Technology, Isfahan, Iran, in 1992, and the M.Sc. degree in electronics engineering and the Ph.D. degree in electrical engineering from the Amirkabir University of Technology, Tehran, in 1996 and 2003, respectively. From 1994 to 2002, he conducted research for the Iranian Research Organization Science and Technology on the topics, such as parallel processing, robot stereo vision, and DSP boards development. In 2003, he joined the University of Isfahan, Isfahan, as an Assistant Professor, where he was promoted to an Associate Professor and a Full Professor, in 2010 and 2015, respectively. He has authored more than 300 papers published in peer-reviewed journals and conference proceedings, and five books. His research interests include remote sensing, image processing and analysis, computer vision, neural networks, and pattern recognition.

ALI BOHLOOLI (Member, IEEE) received the B.Sc. and M.Sc. degrees (Hons.) in computer engineering from the Department of Electrical and Computer Engineering, Isfahan University of Technology, Iran, in 2001 and 2003, respectively, and the Ph.D. degree in computer engineering from the University of Isfahan, Iran, in 2011. He is currently an Assistant Professor with the Faculty of Computer Engineering, University of Isfahan. His research interests include wireless networks, network modeling, and cyber-physical systems.