Case Study

A collision-avoidance system for an electric vehicle: a drive-by-wire technology initiative

Ikenna Chinazaekpere Ijeh[1,2]

Received: 9 August 2019 / Accepted: 2 March 2020 / Published online: 21 March 2020

Abstract
The proposed collision-avoidance system has a network of near real-time embedded subsystems, designed to exploit three operational conditions to control the warning and speed of an electric vehicle at an imminent collision. Further recommended is a multi-level redundancy strategy, using a majority-voting pattern and a fault-tolerant bus, to ensure the efficient operation of the collision-avoidance system in case of a fault in any of its networked subsystems. Analytically studied is the performance evaluation of the fault-tolerant system and its effect on the data processing time. Implementation results demonstrate the proposed fault-tolerant strategy, which takes into account the reliability of both subsystems and data, to be more efficient than a hardware or software sole-based fault-tolerant system.

Keywords Drive-by-wire technology · Collision avoidance system · Optimal navigation · Fault-tolerant · Automated-guided vehicle

1 Introduction

Since the emergence of the drive-by-wire technology (DBW), there has been a continuous attempt to steer automobiles away from there mechanical features [1, 2]. The most significant potential of this feat is to deploy driverless vehicles but with the concern to avoid collision with obstacles. Collision avoidance is a crucial issue not only for the transport and logistics industry but also for policymakers.

Consequently, the European Union regulation No. 661/2009 made it compulsory for the attachment of collision avoidance (CA) systems such as advanced emergency braking systems (AEBS) and lane departure warning systems (LDWS) to certain vehicle categories based on their usage [3]. The study of the United Nations Economic Commission for Europe (UNECE) on the implementation of an AEBS in heavy vehicles designed from 2013 will reduce collisions up to 27% and save close to 8000 lives each year [4].

Presently CA systems have significantly improved due to an exponential advancement in some vehicle's braking and sensing systems (brake-by-wire technology) hence easing the addition of functions such as electronic brake force distribution, traction control and brake assist to vehicles [5].

Furthermore, [6] proposed the implementation of an electro-hydraulic braking (EHB) system, involving the use of pumps and valves with the aid of a control computer to stop the host vehicle. The EHB system is a typical brake-by-wire (BBW) control system first implemented in the fifth generation (2001–2002) of Mercedes Benz Sport Lightweight Series by Daimler Benz [7]. However, deployment concerns rose mostly due to complications in the electrical and mechanical components, decelerating efficiency, accumulator safety and 2-wheel versus 4-wheel backup modes [8].

Another non-commercialized BBW system, just like the EHB, is the electro-mechanical brakes. Hence, there

✉ Ikenna Chinazaekpere Ijeh, ikenna.ijeh@funai.edu.ng | [1]Department of Control and Instrumentation, University of Derby, Derby, UK. [2]Department of Electrical/Electronic and Computer Engineering, Alex Ekwueme Federal University Ndufu-Alike Ikwo, Ikwo, Nigeria.

has been little success in introducing autonomous driverless vehicles on public roads because of safety concerns, especially in relation to a collision.

Exploited in this paper are some of the wide range of efficient operational flexibility the drive-by-wire technology offers in developing a robust self-guided electric vehicle, which would contribute to its deployment on public roads.

A major challenge in designing an efficient CA system is that even in normal driving conditions, there is still an unnegotiable need for efficient sensing of the dynamic traffic situation ranging from obstacles to traffic rules. In most cases, there is a possibility of false alarms or subsystem faults, which would compromise the reliability of the entire system.

For the aforementioned reasons, most works have proposed several fault tolerant strategy often involving subsystem monitoring, malfunction detection and isolation with a possible backup mechanism. In most DBW designs, exchange of data happens between a supervising controller and other subsystems involved with; yaw stability control, anti-lock brake or traction control [9].

This work proposes a multi-level redundancy strategy coupled with a majority-voting scheme, and a fault-tolerant (FT) communication protocol integrated into the collision- avoidance system to improve the overall system reliability and efficiency. In addition, this work assumes vehicles in traffic as the primary obstacle, hence the exception of the steer-by-wire concept in the case of an obstacle as in [10] and [11].

The subsequent discussions of this paper are as follows. Section 2 introduces the framework of the system with respect to collision detection, collision avoidance, and fault tolerance. Details of algorithms for the development of the networked embedded subsystems are in Sect. 3. Section 4 presents the assumptions made in this work, the numerical data and the performance analysis for the real-time deployment of the proposed system using a commercial off-the-shelf automated guided vehicle (AGV). Finally, Sect. 5 is the conclusion of this work.

2 Theoretical framework of the system

The operational principles of most related works in collision-avoidance systems and fault-tolerant systems have inadequate exploitation of the flexibility of the drive-by-wire technology. The theoretical foundation of this work attempts to utilize the available features of a purely electronic architecture to design the operations of the underlined system.

2.1 Obstacle detection system

In [12], the vehicles were equipped with an infrared transceiver, and then a system programmed to maintain a minimum spacing between them or sound an alarm when approaching an unequipped vehicle. Exempted in this work is the idea of a light or radio wave based system as an absolute sensor for receiving and transmitting signals due to its high power consumption, inability to function properly in poor weather conditions, and expensive circuitry. However, associated features of an ultrasonic ranging system (audio waves) are low power consumption, specific distance measurement, a cost and all-weather efficient system [13], making it ideal for obstacle detection.

The obstacle detection system in this work consists of two subsystems, a front and a back, obstacle detection subsystems, FOD and BOD subsystems respectively.

Each of the obstacle detection (OD) subsystem is made-up of a microcontroller and an array of three 'SRF05' ultrasonic sensors (SRF) situated at the bumper, with a maximum ranging of approximately 400 centimetres (Fig. 1).

The SRF05 pin connections in this work are compatible with that of SRF04. For the connection mode considered, the trigger (input) and echo (output) pins are connected separately.

The detection zone of the ultrasonic sensor has a leaf or conical pattern, as in Fig. 2, depicting several echo pulses for measuring the proximity of obstacles [14, 15]. The SRF05 beam pattern has several wave fronts during ranging, so for a reliable and consistent measurement across each of the obstacle detection subsystems, the first detected wave fronts are considered.

In Fig. 2, taking the above beam pattern into account, the left (SRF_1) and right sensors (SRF_3) are at 60° of the bumper's horizontal plane, at which the middle sensor, SRF_2 is located (Fig. 3).

Minimization of circuit complexity is essential in the design of an efficient system, as evident in [16]; the control core of their system was a single-chip microcomputer [12] that used both a 555 timer and a pulse generator. In this work to ensure a robust and simple circuitry, an mbed LPC 1768 microcontroller (controller), with pin descriptions in Fig. 4, was used to monitor and control pulse generation of the entire circuitry.

For the SRF05 to start ranging it requires a minimum trigger pulse input of 10 µs from the controller with which the SRF05 will transmit an ultrasonic burst of 8 cycles at 40 kHz and at the same time raise its echo line high. It then waits to detect an echo, when it does it drops the echo line. This makes the echo pulse (in µs) proportional to the distance of the object. At a no detection case, after about 30 ms the echo line is lowered [14].

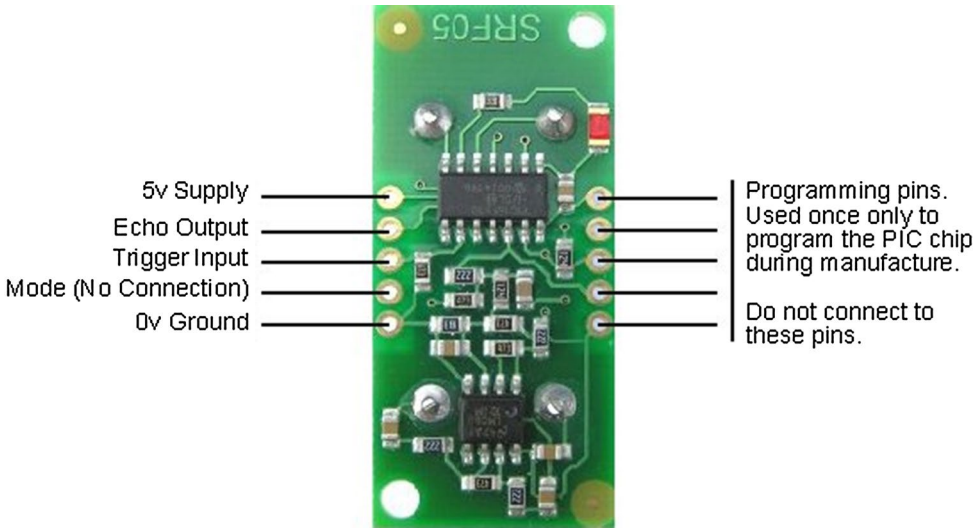

Fig. 1 SRF05 pin descriptions [14]

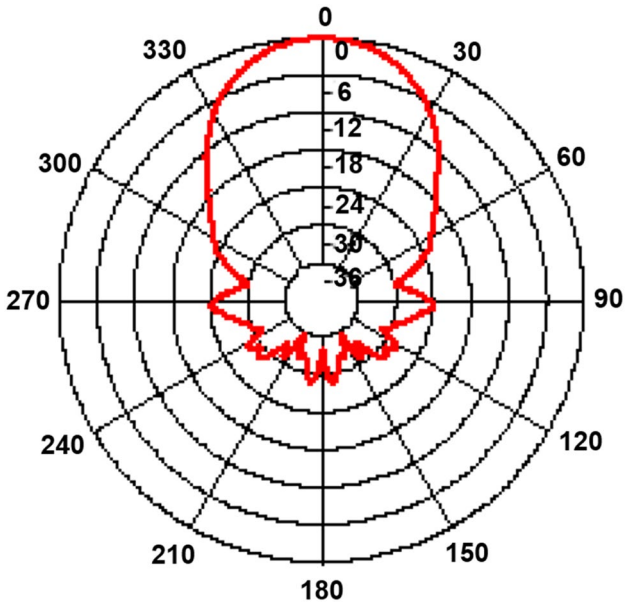

Fig. 2 SRF05 beam pattern [14]

The controller and the ultrasonic sensors communicate via the Inter-Integrated Circuit bus (I^2C) as illustrated in Fig. 5. Ranging data are simultaneously analysed from the three sensors (SRF_1, SRF_2, and SRF_3). The prioritised sensor has the least data value, hence has closest proximity to an obstacle.

In the automobile forward or reverse movement, the FOD or BOD subsystem respectively send signals to the controller on any object proximity within its coverage.

Given below is a brief discourse of the software and the hardware development for Fig. 5.

2.1.1 Software development

To commence ranging, digital-out pins are declared at the controller to supply the SRF05 sensors with a trigger input pulse. Consequently, digital-in pins are declared at the controller to receive the data from the echo pin of the SRF05.

Within the program, the echo pulse in µs is converted to distance in centimetres by dividing it by 58 [14]. This is because the approximate (back and forth) speed of sound in air at room temperature is a cm in 58 µs.

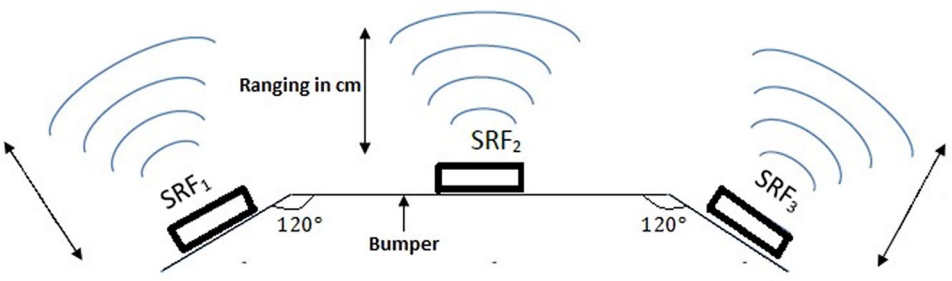

Fig. 3 Proximity sensor configuration on the bumper

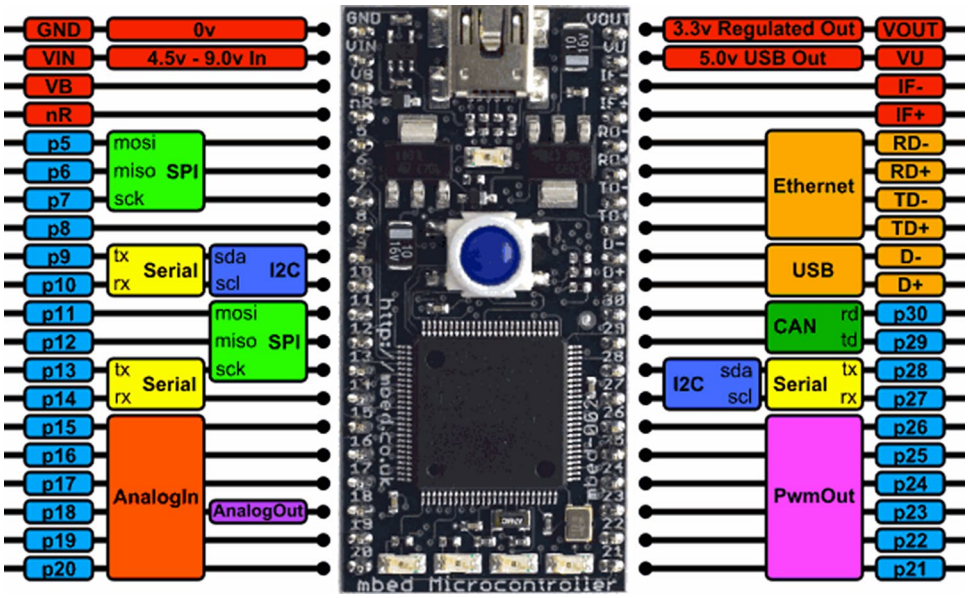

Fig. 4 mbed LPC 1768 microcontroller pin descriptions [26]

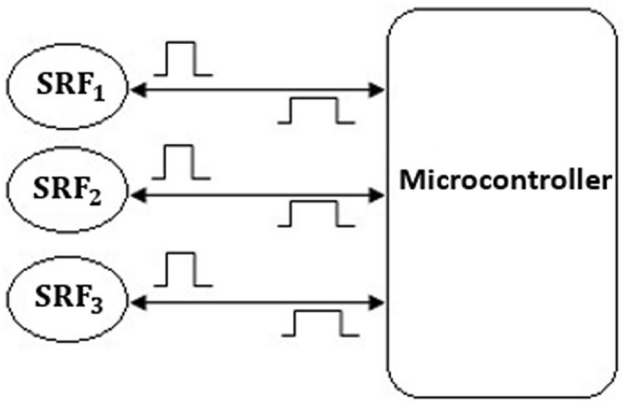

Fig. 5 A typical set-up for any of the obstacle detection subsystem with an I^2C data transfer link

2.1.2 Hardware development

The SRF05 trigger input and echo output pins are connected respectively to a digital-out pin and a digital-in pin of the controller. The controller pins 40 and 1 can bias and ground the connections respectively.

The trigger and echo pins of the SRF05 (slave) are used to create an I^2C data interface respectively with SDA (9) and SCL (10) pins of the controller (master) with a pair of 2.2 kΩ pull-up resistors.

2.2 Collision-avoidance system

The collision-avoidance (CA) system primarily comprises of the obstacle detection (OD) subsystems, the navigation subsystem (NV), and the electronic control unit (ECU) with a Controller Area Network (CAN) as an inter-subsystem communication protocol. The NV subsystem controls the electric vehicle's motor speed via the electronic speed controller (ESC) (Fig. 6).

The receiver connector is connected to the ECU while the motor connector is connected to the vehicle's motor speed terminal.

The ECU serves as a central unit for processing data from the OD subsystems and consequently sending instructions to the NV subsystem on the appropriate action to take in order to avoid a collision.

The calibration of the ultrasonic sensors ranging, at the obstacle detection subsystem, are into three states, represented by three LEDs of different colours for visual indication. In addition, this unit is flexible to link to an audio component, for an audial warning. It is a priority to ensure these warnings does not distract the driver.

The three considered states of the vehicle from an obstacle are safe, caution and halt. This operational design with the inclusion of a steer-by-wire feature would ensure the vehicle navigates efficiently to any destination without using programmed path logs as evident in [10] and [11].

Due to predetermined conditions programmed into the controller, when an obstacle is at a safe distance from the vehicle the driver's desired speed is maintained with a green LED display. Below this distance is the caution distance at which a yellow LED alerts the driver and simultaneously reduces the vehicle's speed. At an exceeded caution range, a flashing red LED comes up as the vehicle is on halt.

When determining the obstacle-to-vehicle distance it is ideal to take into account the speed of the vehicle or obstacle and the speed of obstacle detection. Considering a stationary vehicle or obstacle, with an obstacle-to-vehicle

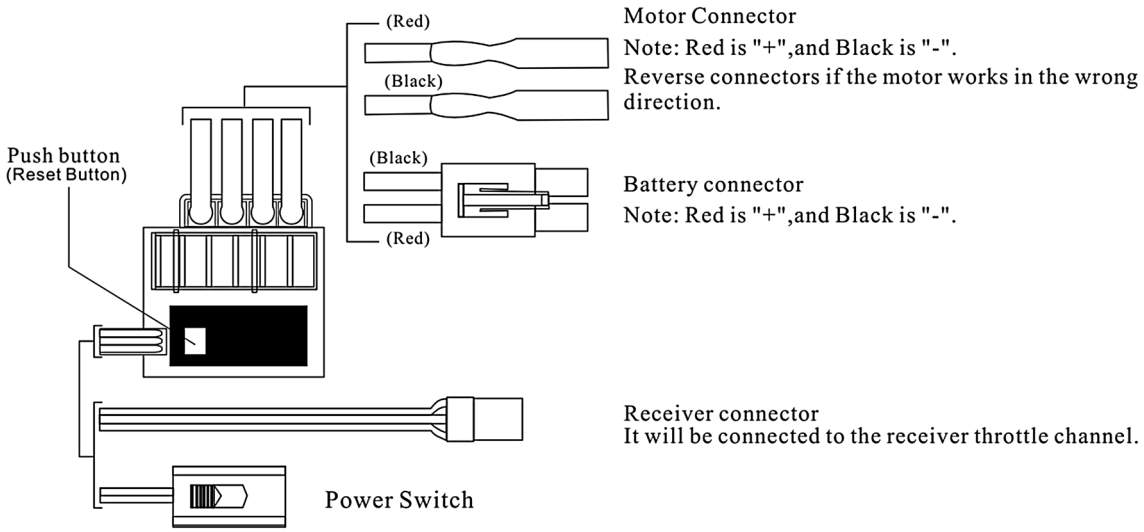

Fig. 6 ESC pin descriptions [25]

distance of 400 cm, the travel time (*Tt*) between an ultrasonic sensor sending out a pulse and receiving an echo pulse (after the sent pulse hits an obstacle) is in the order of 10^{-2} s. However, the obstacle-to-vehicle distance is the instantaneous distance between a vehicle and an obstacle, hence ignoring the speed of the vehicle or obstacle. To take into account the rapid change in *Tt* due to vehicle or obstacle speed, it is necessary to define a range for the instantaneous obstacle-to-vehicle distance.

The operating design expression of the collision-avoidance system is as such that there is a selection of a specific element of the vehicle's speed vector, S, for each instantaneous distance (obstacle-to-vehicle), d. The selection of the speed vector is in relation to a set threshold of distance vector, D.

Hence where,

$$\mathrm{d} = \frac{Tt}{58} \tag{1}$$

$$\mathrm{D} = \left[d_1, d_2\right] \tag{2}$$

$$\mathrm{S} = \left[s_1, s_2, s_3\right] \tag{3}$$

For,

$$\mathrm{d} < d_1 \text{ assign } s_1 \text{ (halt)} \tag{4}$$

$$d_1 \leq \mathrm{d} < d_2 \text{ assign } s_2 \text{ (speed reduction)} \tag{5}$$

$$d_2 \leq \mathrm{d} \text{ assign } s_3 \text{ (normal speed)} \tag{6}$$

$$\text{s.t. } d_1 < d_2\left(d_2 \to +\infty\right) \tag{7}$$

Also expressed in Eq. (6) is the effect of a drop in the SRF05 echo line due to an obstacle being either absent or existing beyond the detection zone.

This concept is similar to the adaptive auto cruise system that maintains a vehicle speed, and when appropriate automatically brakes the vehicle using the ISO standard of up to a maximum deceleration of 0.3 g [5]. Here, beyond the caution distance, the vehicle only moves when the obstacle is out of the halt range.

The control design from the caution distance to the halt distance is derived from of the assessment by [17] that a vehicle's anti-collision system should apply the brake at minimum speeds (less than 50 km/h), while the steering can be applied at maximum speeds. In addition, [4] asserted that vehicles could efficiently apply automatic brakes to a deceleration limit of 0.4 g.

The need to reduce the vehicle speed before its abrupt stop is to avoid skidding or the vehicle losing traction and then roll over. In this work, the CA system has no control over the vehicles steering as some level of control is to be left with the driver to avoid panic and also speed control is sufficient to prevent a collision in a normal traffic scenario.

To implement the operation above, the design of the aforementioned subsystems takes into account ultrasonic sensors, indicators (LED), microcontrollers and ESC. Figure 7 is the system architecture of the CA system design within this work.

Given below is a brief discourse of the software and the hardware development for Fig. 7.

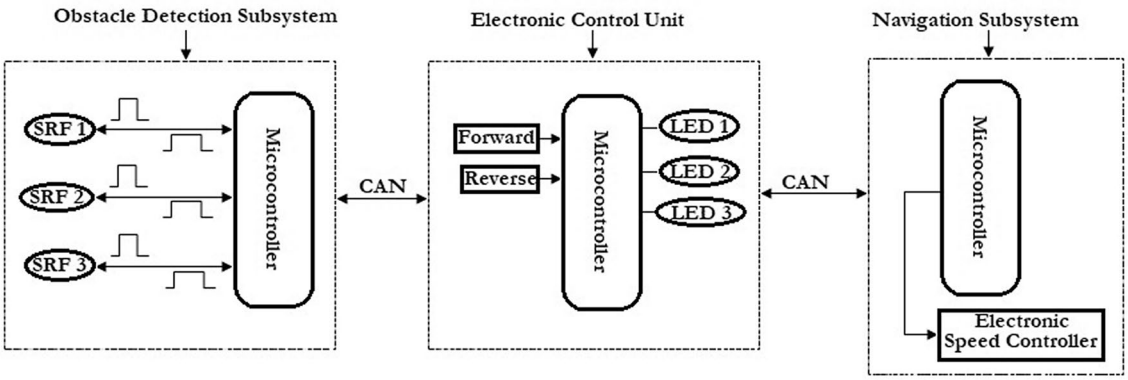

Fig. 7 Collision-avoidance system architecture

2.2.1 Software development

At the ECU, two digital-in pins are declared for the forward and reverse movement using a CAN communication interface, which in turn initiates communication respectively between the FOD or the BOD subsystems and the ECU.

Three digital-out pins on the ECU are assigned as input pins to the LEDs, which is triggered on by a logic level '1' and is used to indicate the various obstacle-to-vehicle distance states for the CA system. Declared at the NV subsystem, is a PWM-out pin to control the motor speed by sending discrete pulse width signals to the ESC in relation to the obstacle-to-vehicle distance states from the ECU.

2.2.2 Hardware development

The CAN interface transmitting and receiving pins (29 and 30) are connected to appropriate subsystems based on the given operation design.

2.3 Fault-tolerant system

A typical system design focuses only on the classical operational status of the proposed system without the depth consideration of its operational efficiency in the case of a fault. These systems are not ideal to deploy for applications with a need for high reliability and sensitivity, as required from the proposed system here, hence the necessity for incorporating a fault-tolerant (FT) control system. Points of fault in a typical electric vehicle are at the sensors, actuators, microcontrollers and communication protocols [9]; these collectively make up a typical embedded networked system.

Several works have suggested designs to ensure system reliability and safety in the case of a fault at any of these points, such as, a dual motor and dual microcontroller architecture by [18], and a dual modular redundancy for a central control unit by [19].

It is obvious that every technology comes with its peculiar fault, for the drive-by-wire (DBW) technology is its susceptibility to electromagnetic interference (EMI), in addition to vehicles exposure to high temperature and humidity variations which [20] assert to be unfavorable for any signal or electronic device. This was evident in 2013, during the recall of 23 Nissan 2014 Infiniti Q50 cars due to the susceptibility of its inherent software, disabling steering in cold temperatures [21]. Despite the vehicles having backup mechanical steering, an occurrence of the above failure could increase the latency period for the backup mechanism.

In addition, from the above narrative, it shows how important it is to design fault-tolerant systems based on both hardware and software as they both pose a significant danger to the entire system.

Assuming the road wheels actuator (servomotor) is operational during a failed subsystem state in the proposed system. This work proposes a majority voting (software) scheme for sensor data [22, 23] and a double standby subsystem (hardware) scheme with a fault-tolerant (CAN) bus [24] to provide efficient fault management in the collision-avoidance system.

For a majority-voting scheme, a $\left(\frac{n-1}{2}\right)$ point faults in 'n' comparable systems can be masked, provided the values of the other $\left(n-\left(\frac{n-1}{2}\right)\right)$ comparable systems are correct. However, this is only true for 'n' (odd number) comparable systems [22].

In Fig. 8, the subsystems enclosed in dotted blocks are the double-standby redundant subsystems, the active subsystem switches to either of them in the case of a fault. The CAN communication protocol is an ideal fault-tolerant bus with its inherent features of error detection and data security.

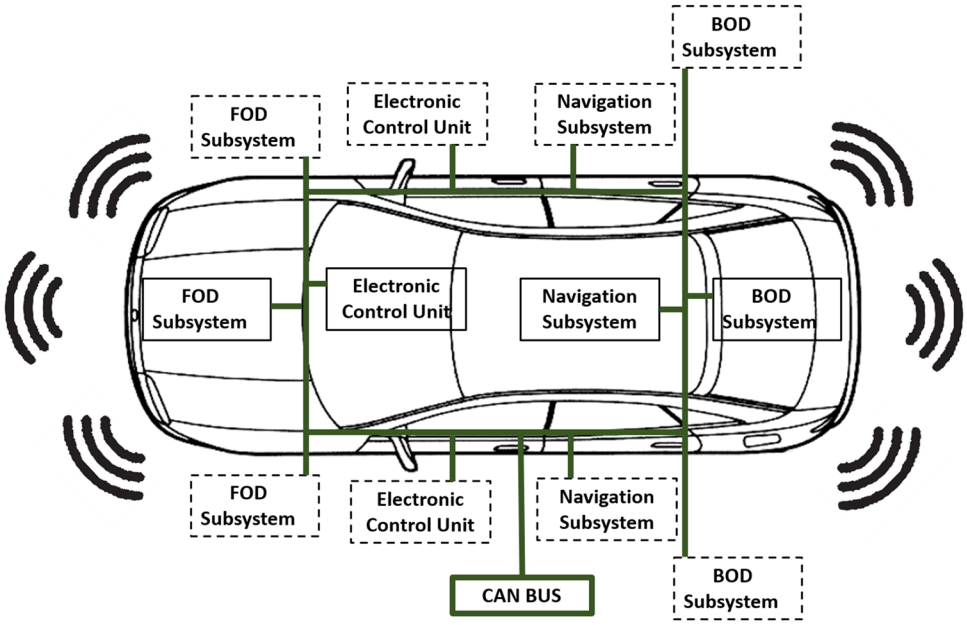

Fig. 8 System architecture overview of the prototype vehicle

Discussed in the next section are the algorithms for the theoretical frameworks of the CA and FT systems.

3 Development of subsystem algorithm

Based on the working principles of the collision-avoidance system and the fault-tolerant system given in Sect. 2, below is an illustrative technical discourse of the embedded subsystems that make up the final developed system.

3.1 Collision-avoidance system algorithm

The operation of the collision-avoidance (CA) system deploys three subsystems, the electronic control unit (ECU), the obstacle detection (OD) subsystem and the navigation (NV) subsystem, as in Fig. 7.

A CAN data transfer link is created between the ECU and both the OD subsystem and the NV subsystem. The communication protocol design is such that the NV subsystem is responsible for the appropriate control of the vehicle speed based on the obstacle-to-vehicle distance status obtained from the OD subsystem via the ECU.

Two separate data inputs initiate a communication between the ECU and either the FOD or the BOD, to establish the instantaneous direction of the vehicle. This could also create a forward or reverse movement; hence, they are more like shift-by-wire inputs.

For the OD subsystems to commence ranging, supplied simultaneously from the ECU to the three SRF05 sensors of the FOD and the BOD is a minimum trigger input pulse ($\geq 10\ \mu s$) at their clock ports. Consequently, an echo pulse returns to the ECU's digital input port for the execution of the appropriate collision avoidance strategy.

The NV subsystem exploits the obstacle-to-vehicle distance data from the OD subsystem, via the ECU, for the motor speed control as in Eqs. (4)–(6). The NV subsystem generates the appropriate pulse width signals, used to control the motor speed by sending discrete pulse width signals to the ESC receiver throttle channel in relation to the obstacle-to-vehicle distance from the ECU. As from Sect. 2.2, the output of the LEDs indicates the speed of the car.

Hence, the basis of assigning the CA strategies from the OD subsystem and the NV subsystem are respectively on the obstacle-to-vehicle distance and the vehicle speed.

Further illustration of the CA system algorithm using Eqs. (1)–(7) is in Fig. 9.

To ascertain the instantaneous motion of the vehicle is relevant to prevent controlling the vehicles speed because of an obstacle in the opposite direction, hence the decision blocks, forward and reverse motion.

3.2 Fault-tolerant system algorithm

The initial and fundamental consideration for the design of a fault-tolerant (FT) control system is the choice of an appropriate fault detection technique that majorly monitors the data process and identify irregularities.

The features of the microcontroller are exploited in order to develop the FT control system, such as to perform a majority voting for sensor values in the ECU's source code, and the use of the controller area network (CAN)

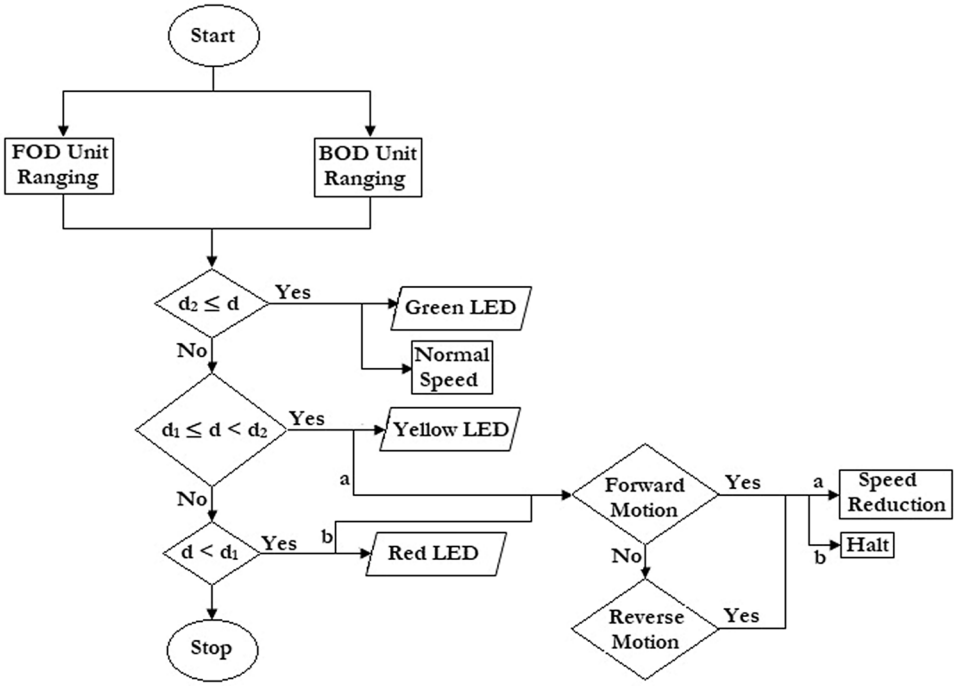

Fig. 9 Algorithm flowchart of the collision-avoidance system

communication protocol to detect and manage subsystem faults.

For the hardware FT algorithm as in Fig. 10, each of the subsystems has two redundant subsystems connected via fault-tolerant bus on cold standby. Continuously monitored during the operation of any subsystem, is its ability to either write or read to another subsystem. An active subsystem is detected faulty when it cannot read or write, at which the FT system automatically switches it to one of the next two redundant subsystems.

This ensures adequate functionality of the entire system in the case of a fault even at the element level (e.g. ultrasonic sensor). An LED indicates the functionality status of the subsystem.

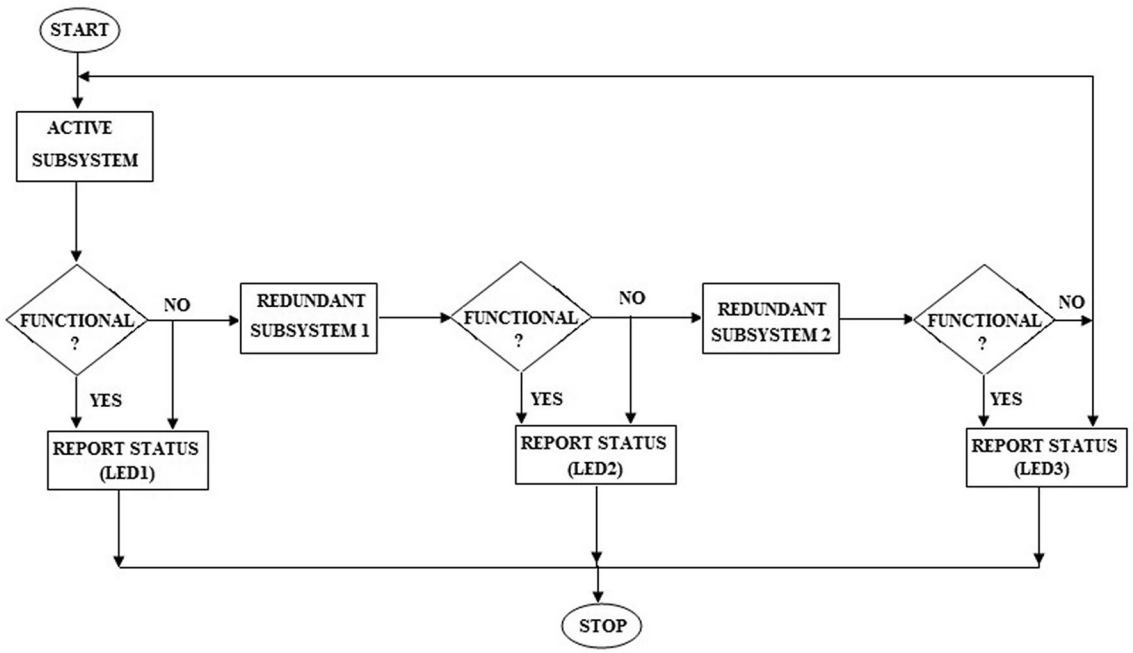

Fig. 10 Double cold standby master–slave hardware scheme flowchart

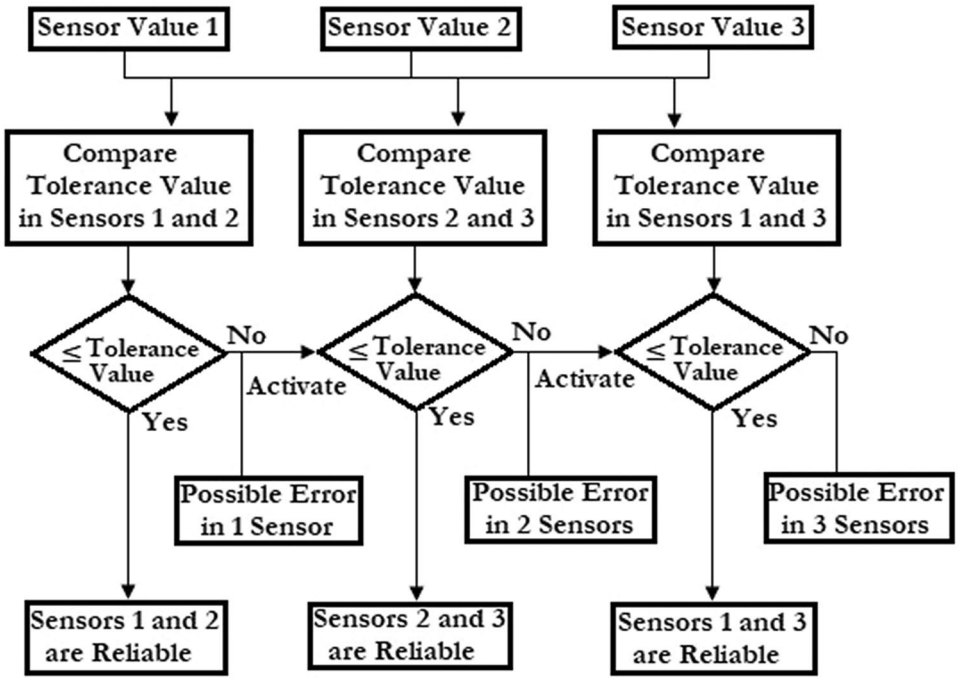

Fig. 11 Majority voting program flowchart

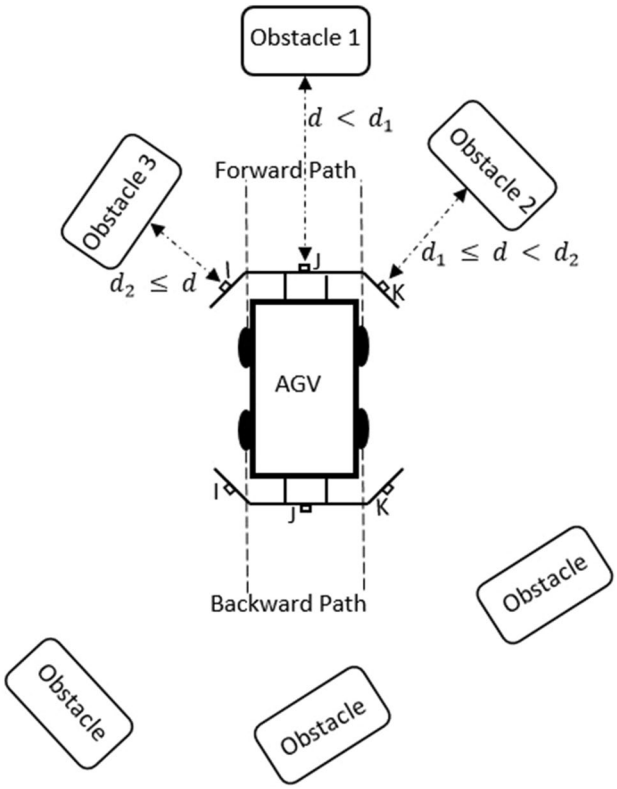

Fig. 12 Experimental scene showing an AGV with the developed system and the obstacles along its path

Figure 11 illustrates the adopted software FT scheme, which is a majority voting strategy for independent sensor values from three comparable systems. To ascertain the reliability of each sensor's data, a selected sensor value is subtracted from the value of any of the other two sensors to compare the result within a pre-determined tolerance range. If the difference between the two sensor values are within the tolerance range only then are both sensors reliable.

Figure 11 masks a single-point error for three comparable systems with majority voting; provided the values of the other two systems are correct otherwise the entire system shuts down until the faults are rectified.

4 Numerical data and performance analysis

Figure 12 gives an illustration of the used experimental scene for the developed system. The implementation of the proposed system was on an off-the-shelf AGV, Himoto Rock Crawler Electric Vehicle [25]. The system comprises of the collision-avoidance (CA) system and the fault-tolerant (FT) System.

4.1 Testing and validation

Given in Table 1 are the data used in the experiment of the developed CA system on an AGV.

Table 1 Operating conditions of the collision-avoidance system on the AGV

Condition (LED)	Distance (cm)	Switch word	Received value
Safe (green)	$d \geq 60$	0×01	1
Caution (yellow)	$30 \leq d < 60$	0×02	2
Halt (red)	$d < 30$	0×03	3

The calibration of the distance threshold ($d_1 = 30$ cm and $d_2 = 60$ cm) for collision avoidance can be based on application or rationally arbitrary. The OD subsystem sends appropriate switch words to the ECU, which in turn sends received values to the navigation subsystem for the appropriate speed control of the wheels.

During testing, given in Fig. 13 is the observed response of the CA system to an obstacle in terms of PWM duty cycle. Further in Table 2, are the obtained results from the experimental scene given in Fig. 12, assuming the given obstacle placements for the AGV forward motion are identical to its reverse motion.

Shown in Fig. 13 are slopes for the change both in PWM and in distance, in-between the boundaries of each condition. The slopes are denoted as $(r_{s\leftrightarrow c}, f_{s\leftrightarrow c})$, and $(r_{c\leftrightarrow h}, f_{c\leftrightarrow h})$ respectively for switching conditions safe-to-caution, and caution-to-halt, (and vice versa) for the reverse, and the forward direction. The slopes tend to zero with the resolution of the threshold distance and the switchover time in-between conditions is negligible due to the fast switching done by the microcontroller with its 32-bit ARM Cortex-M3 core running at 96 MHz [26].

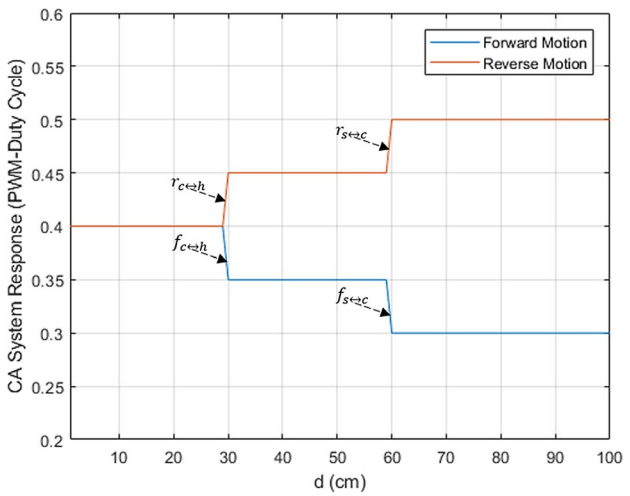

Fig. 13 CA system response for any instantaneous obstacle-to-vehicle distance

Table 2 CA system response to the considered experimental scene

Obstacles present			CA system response (PWM-duty cycle)	
1	2	3	Forward	Reverse
1	0	0	0.3	0.5
0	1	0	0.35	0.45
0	0	1	0.4	0.4
1	1	0	0.35	0.45
1	1	1	0.4	0.4

Based on the considered AGV, there are three PWM duty cycle calibration of the ESC for both the forward, and the reverse motion. In the forward motion, normal speed, reduced speed, and halt has their respective PWM duty cycles as 0.3, 0.35, and 0.4. The aforementioned corresponds in the AGV reverse motion as duty cycles, 0.5, 0.45, and 0.4.

In the OD subsystems, the three SRF05 (I, J, and K, from Fig. 12) were calibrated individually but raising similar flags for identical conditions (Eqs. 4–6). This was to exercise the advantage of creating an independent instantaneous effect of all three SRF05s on the CA system.

During testing for instance, from Table 2 for an instantaneous forward direction of the AGV and only obstacle 1 (from Fig. 12) present the vehicle would maintain its normal speed (PWM = 0.30). Further, if obstacle 2 is introduced the speed of the AGV reduces (PWM = 0.35), and when obstacle 3 is added the AGV stops (PWM = 0.40, same for both forward and reverse motion). Interestingly, each SRF05 although responded independently to the presence of obstacles but most importantly, there was an override of command when any of the SRF05 has an obstacle within its halt condition. Therefore, if all three obstacles were present it is only the instruction routine for the CA condition of 'I' for obstacle 3 takes effect, thus prioritizing halt conditions to avoid a collision.

At an initial test, the serial peripheral interface (SPI) communication was used for the FOD and the BOD subsystem (as the network slaves) to interact with the ECU (as the network master). Here, the network slaves were passive until the network master initiated communication, this was ideal so that the ECU only interacted with the FOD or the BOD when the vehicle was moving either forward or on reverse respectively. This communication protocol was discontinued, due to complex and tedious wiring, and hence its susceptibility to electromagnetic interference (EMI).

The two I^2C parallel connections of each of the three SRF05 running from the AGV bumper to the OD subsystem were affected by stray capacitance due to their proximity hence a form of crosstalk was observed at the controller

echo output. The remedy was to establish a common ground for all the SRF05 circuitry while ensuring firm and short connections.

4.2 Integration of the collision-avoidance system and the fault-tolerant system

In the CAN protocol terminology, data refers to CANMessage, and the logic values dominant and recessive are logic 1 and 0 respectively. Message identifiers (IDs) replaced the addresses of subsystems as used in the I^2C protocol, with can1, can2 and can3 used within the ECU source code to represent FOD, BOD and NV subsystems respectively.

The FOD and BOD subsystems only wrote data to the ECU, while the ECU only read data from the FOD and BOD subsystems and then appropriately wrote to the NV subsystem. The communication configuration of the NV subsystem was only to read data from the ECU to determine the speed of the electric vehicle.

The CAN protocol aids to monitor each read or write task of any subsystem and if executed appropriately, a logic 1 status is reported, otherwise, the status is logic 0. At a logic 0 status, the active subsystem switches to the next redundant subsystem, which if not functional, activates the second redundant subsystem. LEDs served as indicators to report the logic 1 and the logic 0 status. In addition, tera term, a video terminal emulator of the PC, monitored the status of the subsystems.

There were various irregularities during data transfer between subsystems, especially from the proximity sensors and these deviated a lot from the initial design of operation. To remedy this differential signalling was implemented, transceivers (MCP2551) were used to convert the digital signals from the mbed CAN port to a differential signal for the subsystems attached to the CAN bus. In addition, a 100Ω resistor was connected between each end of the CAN bus to cancel out data distortion (noise) due to the unavoidable lengthy connection, especially from each of the obstacle detection subsystems closer to the AGV bumper to the ECU.

4.3 Performance evaluation

The performance evaluation of the proposed system is on operation latency, which is a significant metric for a case of redundant systems. Analytically discussed further is the operation latency of the developed system based on the proposed majority-voting scheme.

Typically, data from faulty sensors are not correlated; hence, they cannot fall into the pre-set tolerance range. Table 3 shows different cases when three sensors data are either correct or wrong.

Table 3 Cases of correct and wrong sensor data for the fault-tolerant majority-voting scheme

Case	Sensor 1	Sensor 2	Sensor 3
1	C	C	C
2	C	C	W
3	C	W	C
4	C	W	W
5	W	C	C
6	W	C	W
7	W	W	C
8	W	W	W

Correct sensor data (C) refers to data that fall within the tolerance range based on the actual data, while the wrong sensor data (W) are data that are not within the tolerance range of the actual data. The decisions from the majority-voting scheme, as in Fig. 11, at cases 1, 2, 3 and 5 will be reliable. In cases 4, 6, 7 and 8, the scheme determines the system faulty; hence, the entire subsystem shuts down while the standby hardware scheme switches to a redundant subsystem.

However, for a situation where there are correlations between wrong data sensors, hence they fall within the pre-set tolerance range of a data that is not the actual data. The uniqueness of this scheme would then judge cases 4, 6, 7 and 8, to be reliable data. Based on this, the probability of an error decision from the FT system, $P_{e(FT)}$, in relation to both the probabilities of a wrong data and a correct data, $P_{e(SNR)}$ and 1-$P_{e(SNR)}$ respectively, from the sensors, could be expressed generally using Bernoulli's distribution as,

$$P_{e(FT)} = \sum_{k}^{n} \binom{n}{k} P_{e(SNR)}^{k} \left(1 - P_{e(SNR)}\right)^{n-k} \tag{8}$$

$$\text{s.t.} \quad k = \left(n - \left(\frac{n-1}{2}\right)\right) \tag{9}$$

where at any case, n is the total (odd) number of considered sensors, and k is the number of least sensor errors that cannot be masked. In respect to the majority-voting scheme proposed in this work for $n = 3$, and $k = 2$, if a target $P_{e(SNR)}$ of 10^{-3} is considered the probability of an error decision from the FT system, $P_{e(FT)}$ is 3×10^{-6} if the wrong sensors data are correlated. As data reliability is considered, for each case the total data processing time T (if considering a single sensor) is extended by n, a major degradation contribution of $3 \times T$. Hence, traded for data reliability is data processing time.

Despite the evaluation above, incorporating both an analytical and a more software based fault-tolerant scheme would aid to mitigate the expense of the data processing time and the complex circuitry of redundant

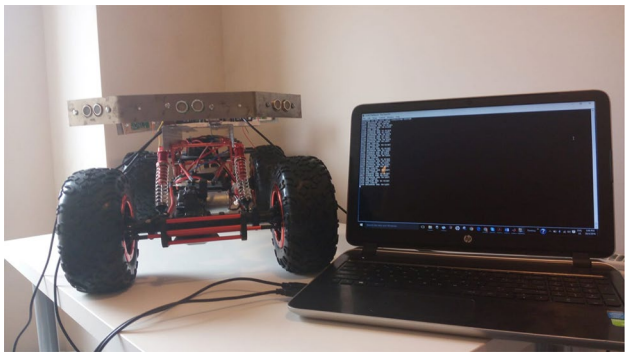

Fig. 14 The AGV installed with the developed system and its status display via tera term on a laptop

systems in the proposed fault-tolerant system. In Fig. 14 is an AGV mounted with the developed system, and the system's operational status is displayed on a laptop screen.

5 Conclusion and future work

Presented is a collision-avoidance system of four networks of near real-time embedded subsystems: obstacle detection subsystems, navigation subsystem and electronic control unit, deployed on an AGV. Implemented is the speed reduction and halting of the AGV at a pre-set tolerable and critical range respectively to avoid an imminent collision with an obstacle. Further proposed in the architecture of the collision-avoidance system, is a multi-level redundancy strategy, using a majority-voting pattern and a fault-tolerant bus. The majority-voting pattern is able to mask a single point error in a sensor value while the fault-tolerant bus ensures data security and switching from a failed active subsystem to a reliable redundant subsystem.

Future research includes the development of an analytical online fault-tolerant system with utmost consideration to reduced redundant elements.

Acknowledgements I acknowledge the support of the Control and Instrumentation Department of the University of Derby, United Kingdom, especially the insightful contributions of Mahmoud Shafik, Tim Wilmshurst and Musa Sumalia.

Funding The Government of Ebonyi State of Nigeria, Scholarship Board (2015/16) funded this work.

Compliance with ethical standards

Conflict of interest The author declare that they have no competing interest.

Appendix 1: Circuit diagrams

See Figs. 15, 16, 17, 18.

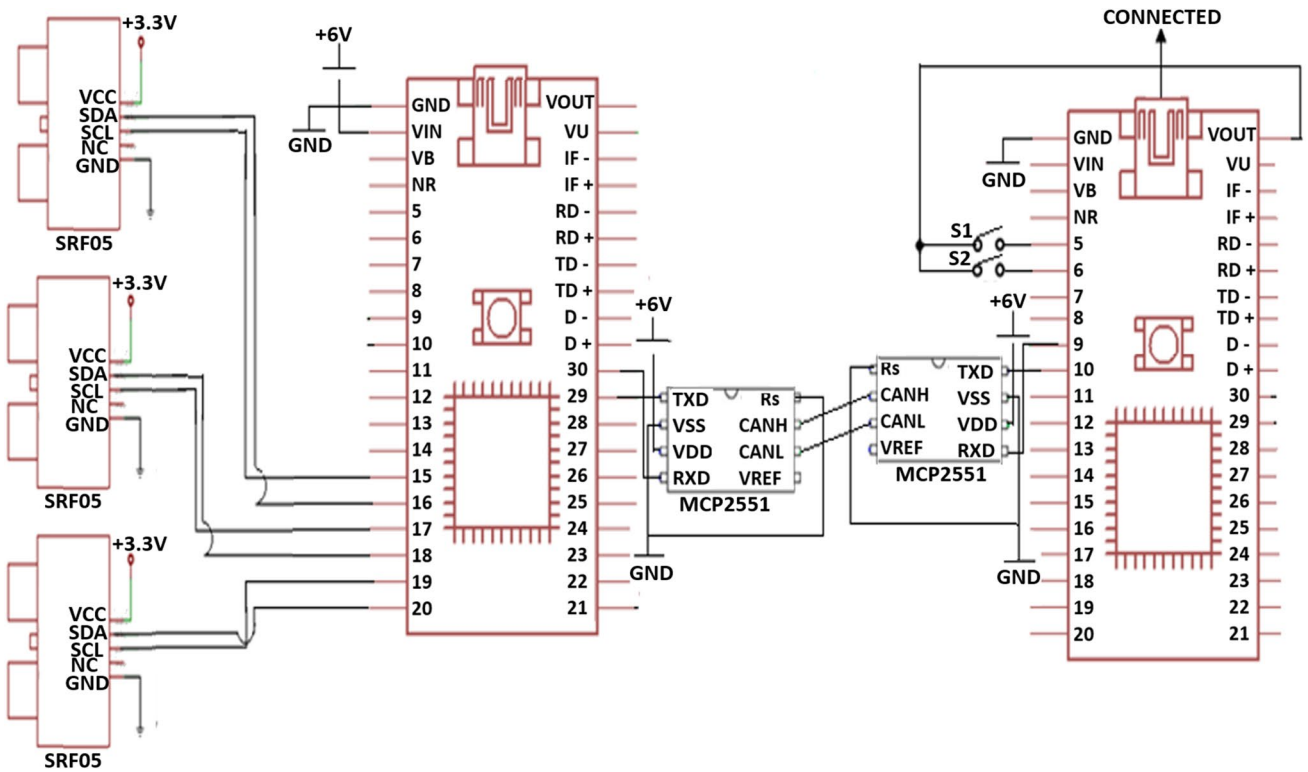

Fig. 15 FOD subsystem to ECU circuit diagram

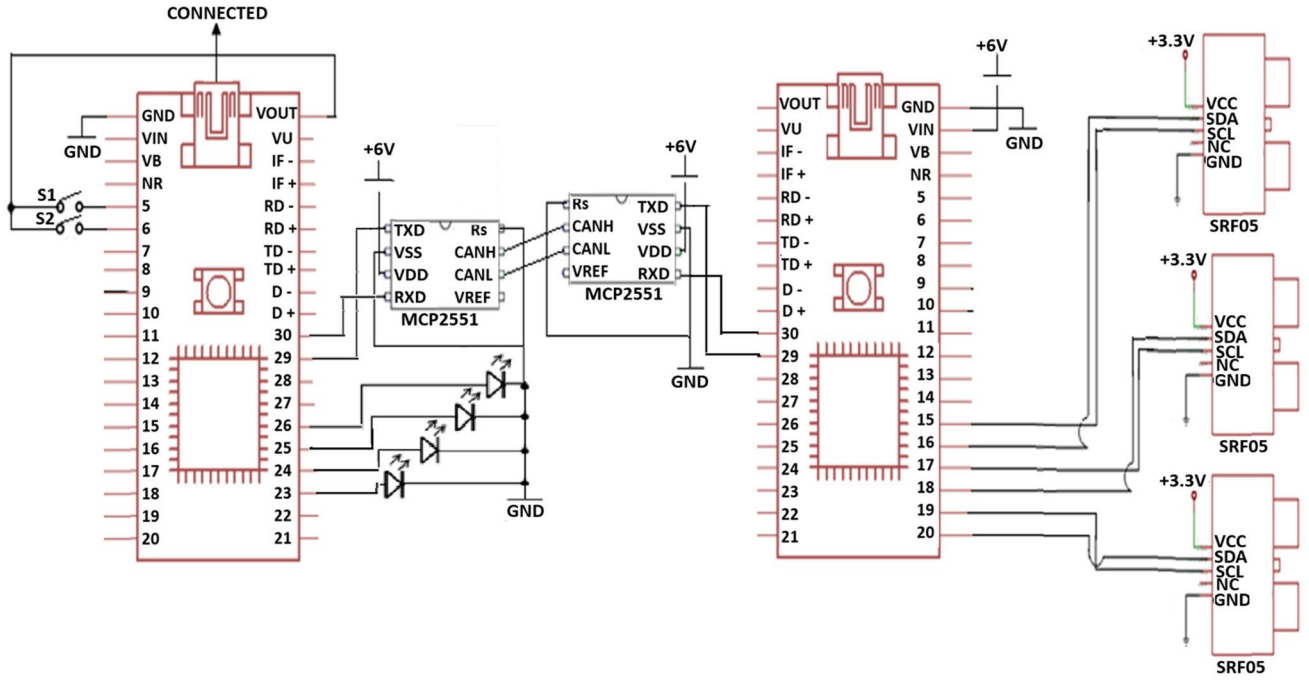

Fig. 16 BOD subsystem to ECU circuit diagram

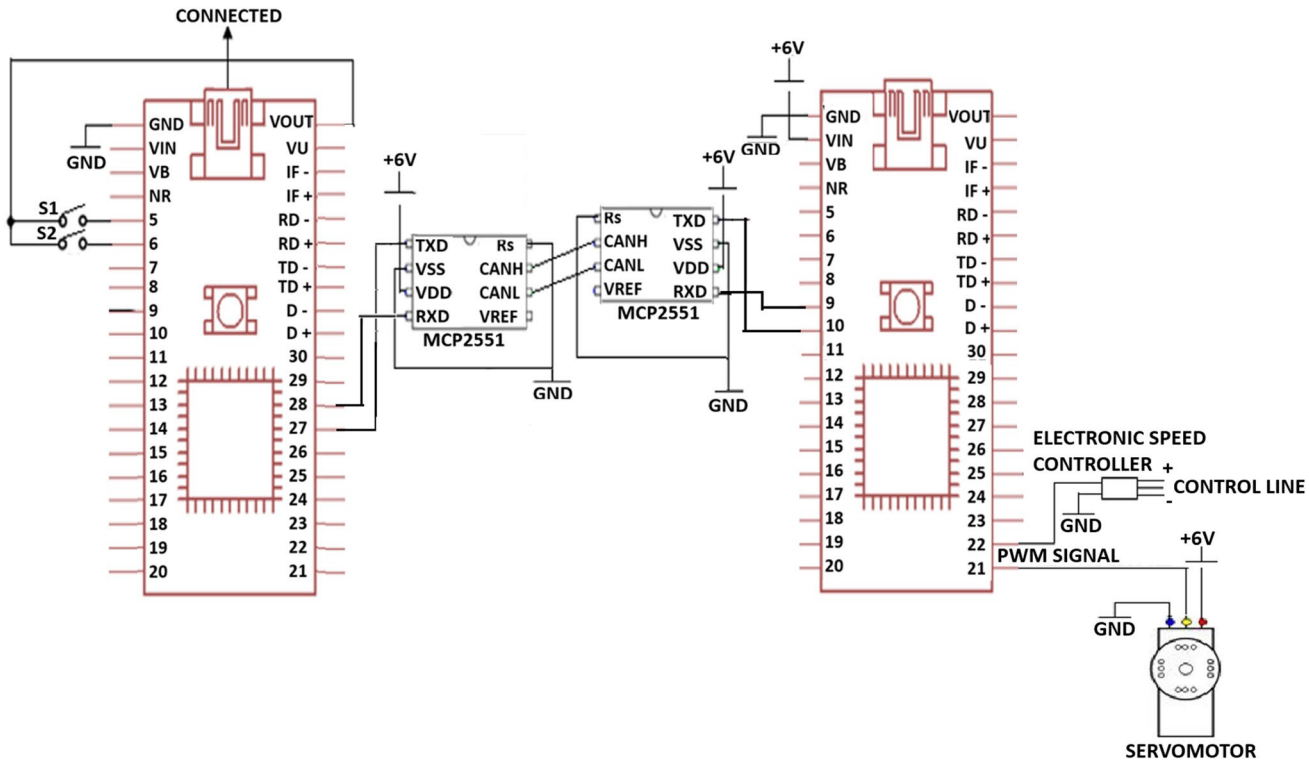

Fig. 17 NV subsystem to ECU circuit diagram

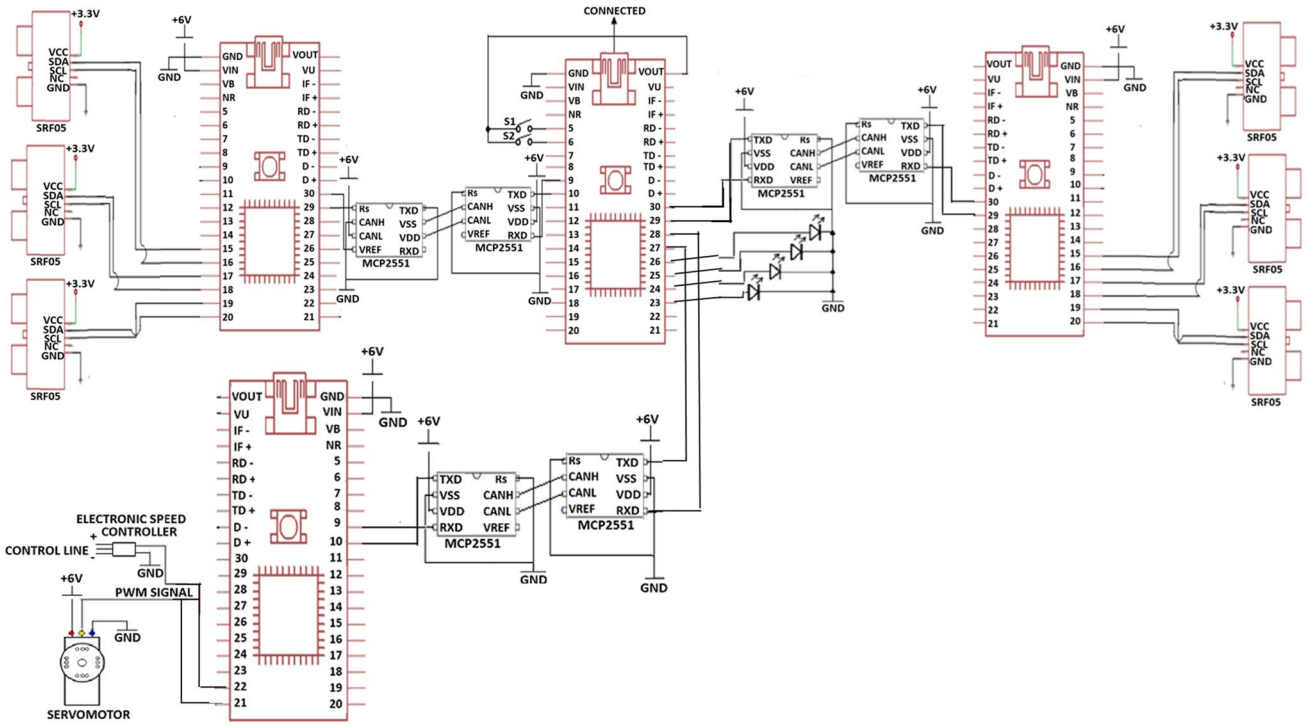

Fig. 18 Complete system circuit diagram

Appendix 2: Electronic control unit mbed source code

```
/*Program 1: ECU communication with the FOD, BOD and NV subsystems*/
#include "mbed.h"
Serial pc (USBTX, USBRX); // tx, rx for Tera Term display
DigitalOut leda(LED1); // ROD status
DigitalOut ledb(LED2); // FOD status
DigitalOut ledc(LED3); // NV status
DigitalOut led1(p23); // safe status
DigitalOut led2(p24); // warning status
DigitalOut led3(p25); // ASR status
DigitalOut led4(p26); // BP status
CAN can1(p30, p29); // ROD CAN interface
CAN can2(p9, p10); // FOD CAN interface
CAN can3(p28, p27); // NV CAN interface
DigitalIn  switch_ip1(p5); // forward condition
DigitalIn  switch_ip2(p6); // reverse condition
char switch_word ;      //data sent from ECU
char recd_val1;         //data received in ECU from ROD
char recd_val2;         //data received in ECU from FOD
int main() {
CANMessage msg; // create empty CAN message
pc.printf("An Auto Anti-Collision System Status Monitor");
while (1) {
if (switch_ip1==1){ // if  ROD message is available, read into msg
  recd_val1= can1.read(msg);
leda = 1;
} else{
leda = 0;
{
if (switch_ip2==1){
 recd_val2= can2.read(msg); // if  FOD message is available, read into msg
ledb = 1;
} else{
ledb = 0;
{
//set leds according to word received from the obstacle detection subsystem
  led1=0;          //preset both to 0
  led2=0;
  led3=0;          //preset both to 0
  led4=0;
switch_word=0xa0;          //set up a recognisable output pattern

  recd_val1=recd_val1&0x0f; //AND out unwanted bits
  if (recd_val1==1){
  switch_word=switch_word|0x01;      //OR in lsb
  led1=1;
        }
  if (recd_val1==2){
switch_word=switch_word|0x02;      //OR in lsb
  led2=1;
      }
  if (recd_val1==3){
switch_word=switch_word|0x03;      //OR in lsb
  led3=1;
      }
  if (recd_val1==4){
switch_word=switch_word|0x04;      //OR in lsb
  led4=1;
      }
  wait(0.004);

recd_val2=recd_val2&0x0f; //AND out unwanted bits
  if (recd_val2==1){
  switch_word=switch_word|0x05;      //OR in lsb
  led1=1;
```

```
        }
    if (recd_val2==2){
switch_word=switch_word|0x06;       //OR in lsb
    led2=1;
        }
    if (recd_val2==3){
switch_word=switch_word|0x07;       //OR in lsb
    led3=1;
        }
    if (recd_val2==4){
switch_word=switch_word|0x08;       //OR in lsb
    led4=1;
        }
    wait(0.004);

// send value to CAN bus and monitor return value to check if CAN
// message was sent successfully. If so light NV status LED

If(can3.write(CANMessage(1, & switch_word, 1))){
ledc = 1;
}else{
ledc = 0;
}
  wait(0.004);
}
}
```

Appendix 3: Obstacle detection subsystem mbed source code

```
/*Program 2: FOD or BOD communication with the ECU*/
#include "mbed.h"
CAN can1(p9, p10); // FOD CAN interface
DigitalOut led1(LED1); // status LED
DigitalIn echo3(p20);
DigitalOut trigger3(p19);
DigitalIn echo2(p18);
DigitalOut trigger2(p17);
DigitalIn echo1(p16);
DigitalOut trigger1(p15);
char switch_word ;    //word we will send
int S;
int W;
int A;
int H;
Timer t;
float l,j,k;
int main() {
   slave.address(0x52);
   t.start(); //start timer
   while(1) {
        //set up switch_word from switches that are pressed
    switch_word=0xa0;         //set up a recognisable output pattern

      trigger1 = 1;
      wait_ms(1);
      trigger1=0; //stop sending pulses
      while(!echo1); //listen for echo pulse
      t.reset(); //reset timer to measure echo pulse width
      while(echo1);
      l=t.read_us(); //attach i to echo pulse width of sensor 1 in us

      trigger2 = 1;
      wait_ms(1);
      trigger2=0; //stop sending pulses
      while(!echo2); //listen for echo pulse
      t.reset(); //reset timer to measure echo pulse width
      while(echo2);
      j=t.read_us(); //attach i to echo pulse width of sensor 1 in us
```

```
trigger3 = 1;
wait_ms(1);
trigger3=0; //stop sending pulses
while(!echo3); //listen for echo pulse
t.reset(); //reset timer to measure echo pulse width
while(echo3);
k=t.read_us(); //attach i to echo pulse width of sensor 1 in us

l=l/58; //converting to cm
j=j/58;
k=k/58;

if(l>100){ // condition for no obstacle
S=1;
W=0;
A=0;
H=0;
}

if(j>100){
S=1;
W=0;
A=0;
H=0;
}
if(k>100){
S=1;
W=0;
A=0;
H=0;
}

if(l>70 && l<99){ // condition for warning
S=0;
W=1;
A=0;
H=0;
}
if(j>70 && j<99){
S=0;
W=1;
A=0;
H=0;
}
if(k>70 && k<99){
S=0;
W=1;
A=0;
H=0;
}
if (l>30 && l<69) {  // condition for reduced speed
S=0;
W=0;
A=1;
H=0;
}
if (j>30 && j<69) {
S=0;
W=0;
A=1;
H=0;
}
if (k>30 && k<69) {
S=0;
W=0;
A=1;
```

```
        H=0;
    }
      if(l<29){    // condition for halt
        S=0;
        W=0;
        A=0;
        H=1;
        }
        if(j<29){
        S=0;
        W=0;
        A=0;
        H=1;
        }
        if(k<29){
        S=0;
        W=0;
        A=0;
        H=1;
        }
        if(S==1){ // condition for safe
        switch_word=switch_word|0x01;
        S=1;
        W=0;
        A=0;
        H=0;
        }
        if(W==1){ // condition for warning
        switch_word=switch_word|0x02;
        S=0;
        W=1;
        A=0;
        H=0;
        }
        if(A==1){ // condition for speed reduction
        switch_word=switch_word|0x03;
        S=0;
        W=0;
        A=1;
        H=0;
        }
        if(H==1){ // condition for halt
        switch_word=switch_word|0x04;
        S=0;
        W=0;
        A=0;
        H=1;
        }
    If(can1.write(CANMessage(1, & switch_word, 1))){
    led1 = 1;
    }else{
    led1 = 0;
    }
      wait(0.004);
    }
    }
```

Appendix 4: Navigation subsystem mbed source code

```
/* Program 3: NV subsystem communication with the ECU*/
#include "mbed.h"
#include "Servo.h"
CAN can1(p9, p10); // ECU CAN interface
DigitalOut led1(LED1); // status LED
PwmOut steering (p21);// Define PWM Output to wheels servo
Servo motor (p22); // Define PWM Output to the ESC
char recd_val;          //data received in NV from ECU
int main() {
CANMessage msg; // create empty CAN message
while (1) {
if(can1.read(msg)) { // if message is available, read into msg
recd_val= can1.read(msg);
led1 = 1;
} else{
led1 = 0;
{
recd_val=recd_va1&0xf7; //AND out unwanted bits
  if (recd_val==1){
  steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.3); //normal speed
          wait(1);
        }
  if (recd_val==2){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.3); //normal speed
          wait(1);
        }
    }
  if (recd_val==3){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.35); //reduced speed
          wait(1);
        }
    }
  if (recd_val==4){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.4); //halt
          wait(1);
        }
    }
if (recd_val==5){
  steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.5); //normal speed
          wait(1);
        }

        }
  if (recd_val==6){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.5); // normal speed
          wait(1);
        }
    }
  if (recd_val==7){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.45); //reduced speed
          wait(1);
        }
    }
  if (recd_val==8){
steering.pulsewidth (0.0015); //centre
          wait(0.2);
          motor.write(0.5); //stop
          wait(1);
        }
    }
  wait(0.004);
}
```

References

1. Changfu Z, Kai L (2006) Development of the drive-by-wire technology. Automob Technol 3(1):1–5
2. Stewart P, Zavala JC, Fleming PJ (2005) Automotive drive by wire controller design by multi-objective techniques. Control Eng Pract 13(2):257–264
3. Europarl.europa.eu. (2017) Report on saving lives: boosting car safety in the EU-A8-0330/2017. http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+REPORT+A8-2017-0330+0+DOC+XML+V0//EN. Accessed 20 Jan 2019
4. Niveditha P, Gowri S (2014) Collision warning system using ultrasonic sensors and automatic brake system. In: Fifth international conference on recent trends in information, telecommunication and computing—ITC 2014, pp 419–424
5. Grover C, Knight I, Okoro F, Simmons I, Couper G, Massie P, Smith B (2008) Automated emergency brake systems: technical requirements, costs and benefits (TRL published project report PPR 227). Transportation Research Library, Crowthorne.

http://ec.europa.eu/enterprise/sectors/automotive/files/projects/report_aebs_en.pdf. Accessed 9 Jan 2019
6. Milanés V, González C, Naranjo JE, Onieva E, De Pedro T (2010) Electro-hydraulic braking system for autonomous vehicles. Int J Automot Technol 11(1):89–95
7. Higgins A, Koucky S (2002) Mercedes pumps 'fly by wire" brakes into new roadster. Mach Des 74:26
8. Seminarsonly.com. (2016) Electro-hydraulic brake (EHB) system|seminar report, PPT, PDF for mechanical. http://www.seminarsonly.com/mech%20&%20auto/Electro-Hydraulic-Brake-System.php. Accessed 17 Jan 2019
9. Anwar S (2012) Fault tolerant drive by wire systems: impact on vehicle safety and reliability. Bentham Science Publishers, Sharjah, pp 3–28
10. Musa S, Ajibola A, Okafor E, Ijeh I, Gibson J (2017) Design and prototyping of an autonomous un-guided vehicle (AuGV) for hazardous areas usage: moon rover space exploration vehicle prototyping. J Unmanned Syst Technol 5(2):49–56
11. Chinazaekpere Ijeh I, Shafik M (2016) An intelligent anti-collision system for electric vehicles applications. In: Proceedings of the 14th international conference on manufacturing research (ICMR 2016). IOS Press
12. Zungeru A (2012) Development of an anti-collision model for vehicles. Int J Embed Syst Appl (IJESA) 2(4):21–34
13. Feng H (2014) The design of ultrasonic distance measurement system for auto reversing anti-collision. Appl Mech Mater (AMM) 602–605:1546–1549
14. Robot-electronics.co.uk. (2018) SRF05 technical documentation. http://www.robot-electronics.co.uk/htm/srf05tech.htm. Accessed 9 Dec 2018
15. Wickramasooriya A, Hamilan G, Jayawardena LSIL, Wijemanne WMDLW, Munasinghe SR (2008) Characteristics of sonar range sensor SRF05. In: 4th international conference on information and automation for sustainability (ICIAFS 2008). IEEE, pp 475–480
16. Wang Y, Zhang Q (2013) Design of an active automotive safety system. J Eng Sci Technol Rev 6(2):155–159
17. Kanarachos S (2014) A new min-max methodology for computing optimised obstacle avoidance steering manoeuvres of ground vehicles. Int J Syst Sci 45(5):1042–1057
18. Zheng B, Anwar S (2008) Fault-tolerant control of the road wheel subsystem in a steer-by-wire system. Int J Veh Technol 2008:1–8
19. Altby A, Majdandzic D (2014) Design and implementation of a fault-tolerant drive-by-wire system. Chalmers University of Technology, Göteborg
20. Toulson R, Wilmshurst T (2012) Fast and effective embedded systems design-applying the ARM mbed. Newnes
21. NHTSA.gov. (2013) RECALL subject: software may disable steering in cold temperatures. http://www-odi.nhtsa.dot.gov/owners/SearchResults?searchType=ID&targetCategory=R&searchCriteria.nhtsa_ids=13V588000&refurl=rss. Accessed 11 Mar 2016
22. Anwar S (2010) Fault detection, isolation, and control of drive by wire systems. In: Fault detection. InTech, p 231
23. Hasan MSU, Anwar S (2008) Sliding mode observer and long range prediction based fault tolerant control of a steer-by-wire equipped vehicle (no. 2008-01-0903). SAE technical paper
24. Mei TX, Shafik M, Lewis R, Walilay H, Whitley M, Baker D (2007) Fault tolerant actuation for steer-by-wire applications. In: 3rd institution of engineering and technology conference on automotive electronics, 2007. IET, pp 1–8
25. Himotoracing (2016) Unlimited class rock crawler 1/8 instruction manual. http://www.himotoracing.com/manual/MANUAL_HI4880.pdf. Accessed 11 Mar 2016
26. Os.mbed.com. (2019) mbed LPC1768|Mbed. https://os.mbed.com/platforms/mbed-LPC1768/. Accessed 30 Mar 2019