Journal of Engineering Science and Technology
Vol. 15, No. 2 (2020) 858 - 881

BUILDING COMPOSITE EMBEDDED SYSTEMS BASED NETWORKS THROUGH HYBRIDISATION AND BRIDGING I^2C AND CAN

JAMMALAMADAKA RAJASEKHAR, JKR. SASTRY*

Department of Electronics and Computer Engineering,
Koneru Lakshmaiah Education Foundation University,
Vaddeswram, Guntur District, Andhra Pradesh, India
*Corresponding Author: drsastry@kluniversity.in

Abstract

Composite embedded networks are evolving day by day. Such kinds of networks achieved through networking two or more heterogeneous networks. Heterogeneity leads to the requirement of too many conversions leading to too heavy time delays resulting in wastage of time than effective usage of the time for transmission. Choice of proper communication speeds on both sides of reception and transmission is the most crucial aspects to ensure that the time delays reduced. In this paper, a multi-buffer-based system presented that implements concurrent transmission process with output streaming through a single dedicate process. In this paper, experimental results presented, which prove that any combination of speed selection considering the I^2C and CAN, latency cannot be avoided. Bridging the sub-nets in composite ES networks will reduce the latency heavily. The experimental results prove the latency is minimum when transmission speeds of CAN and I^2C are multiples of each other. CAN (500 kbps), and I^2C (1 Mbps) lead to multiples of 2 while the transmission speed: CAN (125 kbps), I2C (400 kbps) leads to multiple of 4. In both cases, the delay time is minimum. Idealistically both CAN and I^2C driven using 1 Mbps speed, however, exact speed matching is not possible.

Keywords: Communication bridge, Concurrent processing, Heterogeneous embedded systems, Serial transmission through streaming.

1. Introduction

Embedded systems are designed to be adaptive systems. A single embedded system generally used for sensing, monitoring, and controlling the behaviour of the external environment. Some of the embedded systems are designed to achieve real-time controlling of the actuating devices. Over the days, one can observe that many of the embedded systems must be selected and networked to cater for a composite application that caters varied processes that involve sensing and actuating.

Networking of several embedded systems is employed to implement applications that have multi-node sensing and actuating. Many networking protocols are in use, which includes I^2C, CAN, USB, and RS485 that is bus-based and effects serial communication. All the devices that can communicate using the same protocol can be connected and implement a homogeneous communication system.

For implementing some of the applications such as automotive systems, aerospace systems, there are requirements of interconnecting several embedded networks as they keep evolving. The sub-nets that need to be interconnected to form composite networking are generally heterogeneous especially the variations in the networking systems that lead to many differences in the communication speeds, handling heterogeneous issues, error detection and control, arbitration, timing, addressing, etc. The heterogeneous embedded systems differ in many ways considering network length, the number of devices that can be connected, Timing, arbitration, address resolution, synchronization, data and message formats, and the message lengths. These stiff requirements do not allow devices that follow different protocols connected on to the same network.

A complex ES based application requires connecting heterogeneous ES devices, into a set of sub-networks implemented using different communication systems. Hybrid networking carried to connect subnets developed over different communication systems. There are many ways to achieve hybridization at different levels of the IoT network say PIN level, Device-level, Gateway level, multi-master level, etc. In this paper, Device-level Hybridization is presented considering an application in the Automotive domain.

Initially, at the beginning of 2000, Engine control systems, braking systems have been introduced, which are enabled through I^2C networks using the embedded sensors. More and more sensing and actuating systems added over the days. Since 2005 onwards the light-sensing, reverse braking, and door closing system introduced, which all required high speed in actuating the controlling mechanisms. I^2C and CAN-based subnets have been used in the past to connect the devices with specific communication speeds. CAN-based communication used when very high-speed communication required among the devices connected in a sub-net.

However, with the increase in automation, a need has raised in moving sensing inputs in between I^2C and CAN making the systems more versatile and dependent. This aspect led to inventing and implementing a solution that bridge networks so that the data moved either way in between I^2C and CAN.

For the communication to take place between the two subnets that are heterogeneous, there can be many approaches, one of which, is the construction of a bridge, which acts as a channel between the two subsystems. There are many issues involved in effecting communication between heterogeneous networks.

Some of them include data transfer rates, addressing, packet format, synchronization, etc.

The bridge must be intelligent and dynamic so that appropriate and relevant communication speeds for reception and transmission achieved, which leads to no delay in the entire data flow from sending to the reception. One such bridge that connects an I^2C network to CAN network presented in this paper.

2. Related work

New Devices developed every day by VLSI industry, which conforms to the latest communication standards released in the market. There are interfacing problems when these devices interfaced with the ongoing applications built on standard boards. Generally, to interface new devices with standard boards, converters are developed that bridges the interface exposed by the devices to be converted into interfaces exposed by the standard boards. The conversions as such are complicated when huge diversity exits in building the interface or for that matter when communication interfaces are to be bridged as the communication standards are available in many different versions.

Cao and Nymeyer [1] presented a theoretical model of a converter that will enable two given arbitrary protocols to communicate. The model presented by them includes buffers and correctness conditions and considers that the protocols to be non-deterministic. Verification of the conditions done in the process and valid data will only be allowed for transmission-model checkers used for verification of the conditions imposed during the process of conversion.

Two standards are used, which include field bus and CAN bus for implementing industry-based applications. Field bus standards are not uniform and greatly differ from industry to industry while CAN bus is a standard communication system. Various applications implemented in the industry require communication between the field bus and CAN bus, which lead to the requirement of developing various convertees either hardware-based, software-based, or by considering both hardware and software. Development of such a converter is complex as field bus is non-standard.

Guohuan et al. [2] have presented an analysis of both the protocols and proposed an interfacing method, which considers the use of both Hardware and software and all the interacting ambiguities. ES networks are built using different communication protocols, especially using

CAN-based system. At times these CAN-based system needs to be interfaced with PC to monitor and control various devices that get connected to the CAN network. PCs not supported with CAN ports on the other hand RS232C ports mostly supported for interfacing with serial devices. When CAN networks are to be connected to PC converters are required to establish communication both ways, RS232C to CAN and vice versa. Wang and Guo [3] have presented a protocol converter that converts RS232C to CAN and Vice versa. The converter is developed using PIC18F2580 microcontroller that has native support within the controller.

Field bus comes in different standards-many fields bus-based network built over the days for implementing different applications. At times, situations aroused that

requires interconnection between two networks that are built using different bus standards requiring the development of conversion from one filed bus standard to the other. Guohuan et al. [4] have developed a conversion method based on ARM technologies. The method is designed to convert from any field bus-based standard to any other field bus standard leading to the development of many to many fields bus-based protocol conversion. They have developed a gateway, which converts the data packets to release through different standards into a standard gateway packet, which is understandable to both the protocols.

Profibus and Modbus are the protocols used for the development of ES based networking and communication, especially in the Industrial front. At times, needs have risen to establish communication between these two types of the network leading to the development of a converter that facilitates the conversion of the protocol either way. Zhang et al. [5] have developed a gateway based on AT895C52 technologies released by ATMEL. In the gateway, a protocol chip Siemens SPC3 is built to facilitate protocol conversions

I^2C communication represented as state machines. The data received from the I^2C transmitter can be placed in a buffer, which can be picked up by CAN transmission system as proposed by Benachinamardi and Wali [6]. The conversion from I^2C to CAN developed through Verilog. The authors have not presented conversion from CAN to I^2C. The authors have just tried to interface at the data level ignoring many aspects, which include speed matching, error detection, synchronization, timing, heterogeneity, etc.

EPA (Ethernet for plant Automation) is developed for monitoring, controlling, and interconnecting plant mounted devices for implementation of industrial automation. MODBUS is also used simultaneously for interconnecting the devices, which follow MODBUS protocols. These two independent networks simultaneously mounted on a plant requiring communication between the devices that connected on either of the networks. Hui et al. [7] mentioned that a gateway developed using ARM technologies that provide the interface required to convert EPA to MODBUS and vice versa. The gateway proposed by them is claimed to be supporting real-time data requirements posed by either of the communication systems.

IEC61850 is a standard used for automation of sub-station related systems. The online monitoring system developed by state Grid Corporation used for building smart grids. A communication system designed based on IEC 61850 standard and the same used for effecting communication while building the smart grids. While that being the case, other types of networking systems based on MODBUS and CAN bus are implemented to support different parts of the automation system. This aspect necessitated establishing communication between IEC61850 and Modbus / CAN Bus leading to the development of a protocol conversion method.

Zhang et al. [8] proposed conversion method between Modbus and IEC61850. Object-oriented technology used for information modelling on Modbus. The model mapping relationship between IEC61850 and Modbus based on the principle of the minimum information point, which maps to one to one correspondence with IEC61850.The protocol conversion method presented has been verified using interval controller.

SPI is the interface present on most of the embedded systems, which support full-duplex communication with high throughput, whereas I^2C is a two-wire bus used for communication between two or more devices normally situated on the same board. And as technology is trending, many features are being built into single portable devices for which, the power is the main concern. Kiran and Vinilanagraj [9] have proposed conversion of SPI to I^2C and vice versa and develop a protocol convertor with flexibility in interoperability and power concern. An effective low power technique like clock gating insertion and dynamic power gating implemented in this design.

In recent days, many devices built with wireless technologies like RF, GSM, Bluetooth, and ZigBee for establishing alternate communication paths to achieve fault-less communication for transmission of sensed data. The sensed data needs to be transmitted to a remote location using a cellular communication system requiring translation of radio-based data to a cellular data. This kind of an issue aptly leads to conversion of wireless data to cellular data, thus, achieving integration of a local communication to a remote communication [10].

Networking of embedded system achieved through many protocols like RS232, RS485, I^2C, CAN, SPI, and USB. Among these, RS485 is the most used protocol in the industry for effective communication. However, the major problem in implementing RS485 based communication system is lack of native support in individual microcontrollers. Sastry et al. [11] proposed RS485 based networking of distributed embedded systems by interfacing MAX485 to RS232C existing in a microcontroller.

Sastry et al. [12] have proposed a method of concerting an RS232C based interface to I^2C communication interface. A design flow method, which uses priority queues has been presented to affect the communication according to the flow required. In similar lines, Sastry et al. [13] and Shwetha and Karunavathi [14] presented the conversions from RS232C to CAN and USB. However, the methods proposed by them are limited to connecting a heterogeneous device to a selected network. These methods do not propose anything related to interconnecting sub-networks built with the different communication system

Due to the recent developments in communication protocols, interoperability problems are emerging. One such example is where client working on Ethernet, USB, and SPI and server is on RS232C protocol. Narayanan and Murthy [15], designed an interface for the conversion of multiple protocols into a single protocol. The design carried for the conversion of Ethernet, USB V2.0 and SPI to RS 232C using DSP processor.

The numbers of wireless IoT devices in automation networks are growing rapidly. The concept of IIOT called Industry 4.0 is emerging, and many industries are striving to take benefits out of it. The first issue, which comes into mind is interoperability issues. Compatibility issues might arise while connecting standard and nonstandard equipment's.

Therefore, protocol conversion equipment is to be used. While using this, Overhead increased. Murty et al. [16] analysed accurate estimation of network performance in IIOT networks by taking Protocol conversion overhead during data transmission. Results obtained are used for forecasting the overhead while designing, thereby deploying efficient traffic flows.

Many hybridisation issues have been discussed in the literature for communication between a pair of devices [17-25] however, have not discussed the communication that must happen between two devices that were situated on two different networks.

3. Comparative Analysis of Approaches to Bridging ES sub-nets

Comparison of the contributions made to the literature has been made considering the following parameters, which are the features required for bridging two ES based sub-networks implemented through different bus-based serial communication protocols. Table 1 shows a comparison. The features required include hardware conversion, software conversion, use of interfacing circuits, use of a Bridge, Gateway, conversion of protocols, and type of networking, which include either bus-based or peer to peer connection. The features further include use of middleware for handling heterogeneity issues, consideration to the data transfer speeds, bus synchronization, timing, data buffering, error detection and control, and use of conditions for verification of the data.

From the comparison, one can see that the solutions proposed do not quite meet the requirements of developing a bridge that interconnects two embedded sub-networks. As such the bridge that connects two ES networks must address various issues that include data speed matching, data buffering, synchronization of data transfer, and timing of communication that must hold good at either of the sub-nets, error detection and control. The issue of connecting many protocols to one and versa is not required when it comes to developing a bridge that connects two sub-networks.

In this report, investigations and findings presented that have led to the development of a bridge that comprehensively interfaces two different ES sub-networks.

4. Overview of an Application with Two Different Sub-Systems

Many sectors evolved over a period implementing different technological solutions especially in the field of automobile Industry focussing on different aspects of monitoring and controlling starting from engine control to GPS systems that provide tracking of the movement of automobiles system.

Many sensing, monitoring, and control systems have been introduced into automobile systems over the days through the introduction of several sub-systems that aim at sensing, monitoring, and controlling some partial aspects of the entire automobile systems. Many sophisticated gadgets added not only to function independently but also function as a part of either sub-system or the entire system.

In a typical automobile domain, the system was chosen that got evolved, especially concerning automation, sensing, monitoring, and actuating. In the typical automobile system, two subsystems considered and implemented over time. One subsystem is built using I^2C network, connected with engine temperature monitoring and control and a braking system. The network is built using 4 Microcontroller based systems, which are heterogeneous and having inbuilt I^2C native interface ports. Age the time progressed; a second CAN-based network introduced that has in it, the car reversing system, Interior lighting system, and door control system.

Table 1. Comparative analysis-bridging heterogeneous ES networks.

Author serial	Author name	Hardware conversion	Software conversion	Use of interfacing circuits	Use of bridge/gateway	Protocol conversion type and mapping	Type of interfacing	Use of middleware	Data transfer speed matching	Bus synchronisation	Timing	Data buffering	Error detection and control	Use conditions for verification of data
1	Jing Cao	-	-	-	Bridge	One to one	BUS	-	-	-	-	√	-	√
2	Lou Guohuan	√	√	√	-	One to one	BUS	-	-	-	-	-	-	-
3	Xianjun Wang	√	-	-	-	One to one	Peer to peer interfacing	-	-	-	-	-	-	-
4	Lou Guohuan	-	-	-	Gateway	Many to many	-	-	-	-	-	-	-	-
5	HaoZhanga	√	-	-	Gateway	One to one	-	-	-	-	-	-	-	-
6	Anupama	-	-	-	-	-	Data level	-	-	-	-	-	-	-
7	Li Hui, Zhang Hao, PengDaogang	-	-	-	Gateway	One to one	-	-	-	-	-	-	-	-
8	Zhang	-	-	-	-	One to one	-	-	-	-	-	-	-	√
9	Kiran	√	-	-	-	One to one	-	-	-	-	-	-	-	-
10	Jaskirat Kaur	-	-	-	-	One to one	-	-	-	-	-	-	-	-
11	Sastry	√	√	-	-	One to one	-	-	-	-	-	-	-	-
12	Shwetha	√	√	-	-	Many to one	-	-	-	-	-	-	-	-
13	Revathy	√	√	-	-	-	-	-	-	-	-	-	-	-

4.1. Overview of prototype sub-system development of I^2C network for experimentation and results

4.1.1. Operational description of the network

Figure 1 shows the topology of the I^2C networked embedded system**.** The I^2C network built with a single master and multiple slaves based on four microcontroller-based systems.

One microcontroller-based system developed for monitoring the temperature within a motor car engine through a temperature sensor (LM35) interfaced with the master.

The master directs another microcontroller-based system for actuating a PUMP or otherwise when the engine temperature is either within or outside the threshold value. The master device also keeps enquiring the status of a braking system, which is monitored and controlled through another microcontroller-based system.

The master also provides instructions to the decoupling system to either couple or decouple the engine from the brake shaft based on whether a brake is applied or not.

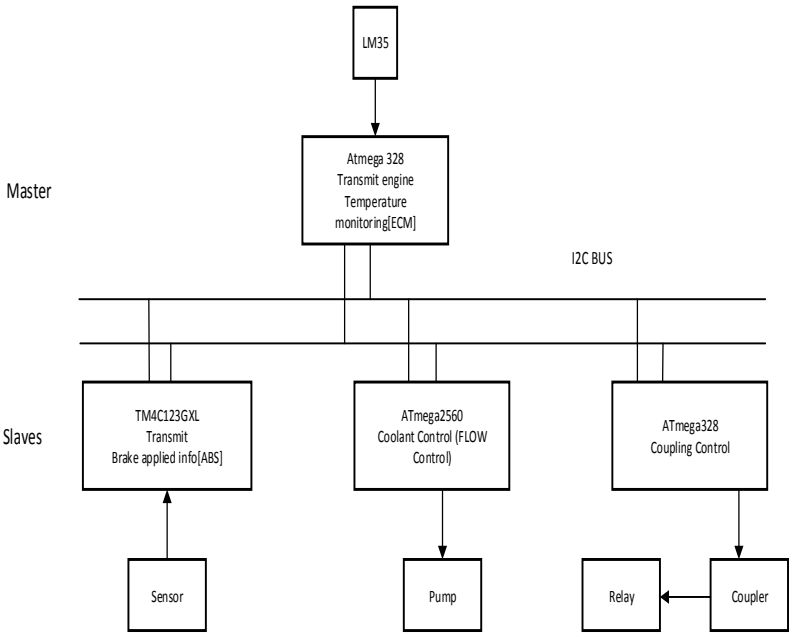

Fig. 1. I²C topology.

4.1.2. Functional requirements of I²C based system

The functional requirements of an I²C system shown in Table 2.

Table 2. Functional requirements of an I²C system.

Functional requirements number	Functional description
1	To sense the engine heat
2	To actuate pumps when the sensed temperature > threshold value
3	To receive the status of the braking system periodically
4	To decouple the engine shaft and when a brake is applied

4.1.3. Hardware description

The hardware details of an I²C system shown in Table 3.

Table 3. Hardware details of an I²C system.

Hardware device number	Hardware description	Interface description	Purpose of the device
1	AT mega 328	I²C	Sensing the engine temperature continuously
2	Tm4c123GXL	I²C	Sensing the brake applied or not
3	AT mega 2560	I²C	Coolant control through the pump
4	AT Mega 328	I²C	Coupling control

4.1.4. Software architecture

The software architecture implemented within the I^2C based embedded systems shown in Fig. 2.

In every microcontroller-based system, a task designed for effecting communication between the master and the slave. Every slave system has an application that either senses or actuates the controlling parameter. The communication through an I^2C system routed from the master system. The breaking, coupling, and pump actuating systems shall perform as per the directions initiated by the master.

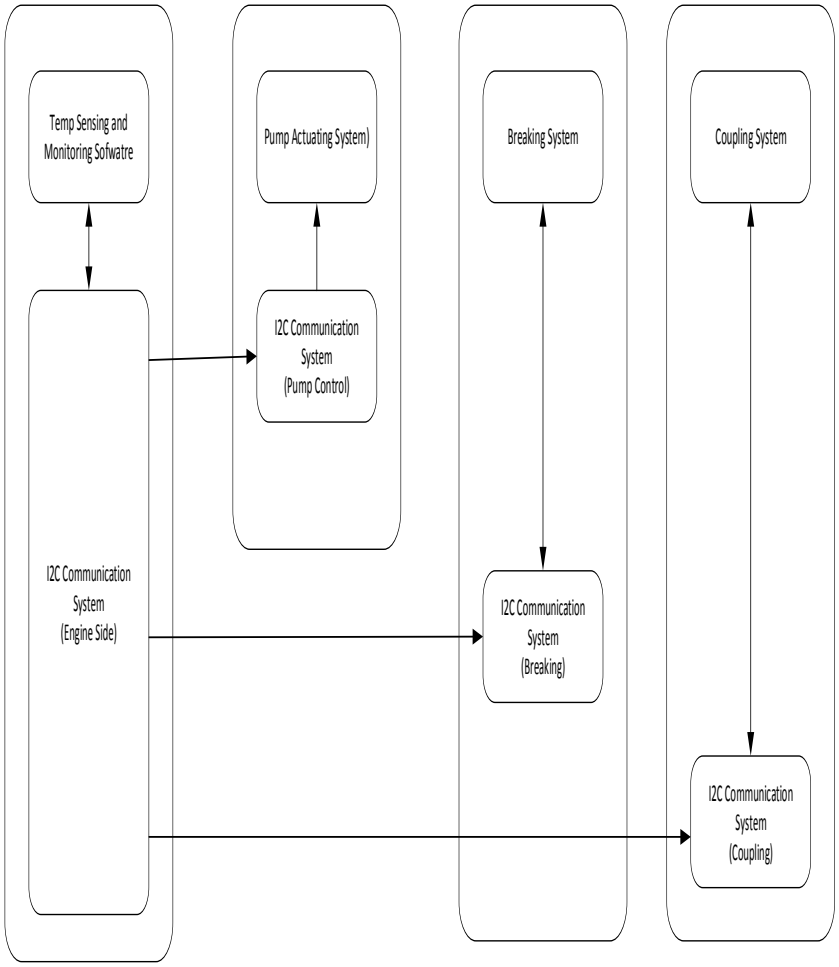

Fig. 2. Software architecture for I^2C system.

4.2. Overview of prototype development of CAN network for experimentation and results

Figure 3 shows the CAN-based networking topology.

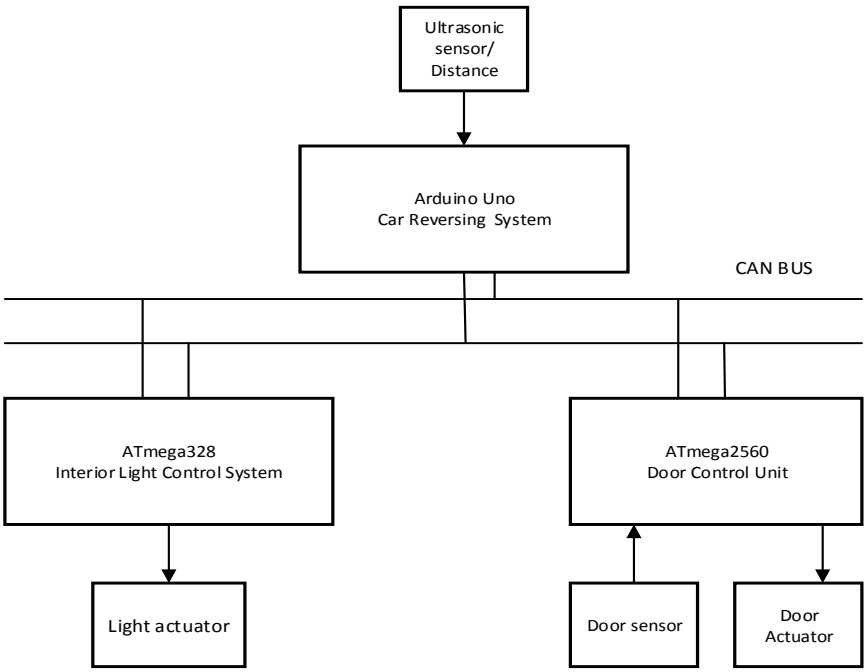

Fig. 3. CAN topology.

4.2.1 Operational description of the network

The CAN network has been built considering three microcontroller-based systems. One microcontroller-based system developed for monitoring the distance between a vehicle and a back obstacle through ultrasonic sensor interfaced to a specific controller while the vehicle is in reverse gear. When the distance value is not in threshold limit, the master sends a signal to the controller, which deals with the internal lighting system so that a flashlight is invoked to indicate short object distance. Another microcontroller-based system developed for checking the door status continuously and actuating in terms of the door closing and opening.

4.2.2. Functional requirements

The functional requirements of a CAN-based system shown in Table 4.

Table 4. Functional requirements of the CAN-based system.

Requirement serial	Functional description
1	To sense the distance of the obstacle from the car while reversing and flashing an internal light when the distance is beyond the threshold value
2	To check the door status continuously and activate a buzzer if a door opened while on the run
3	To check the status of the door and control the lighting system based on whether the door is closed or open and when continuous lighting is on for some time

4.2.3. Hardware description

The hardware details of the CAN system shown in Table 5.

Table 5. Hardware details of a CAN system.

Hardware device number	Hardware description	Interface description	Purpose of the device
1	AT Mega 328	CAN	To sense the distance while reversing and sending the information to the bus
2	AT Mega 2560	CAN	To sense the door status
3	AT Mega 328	CAN	To control the interior lightening system

4.2.4. Software architecture

The software architecture implemented within the CAN-based embedded systems shown in Fig. 4. A specific microcontroller, which acts as master implements as car reversing system in this CAN network. The Reversing system computes the distance between a nearby sighted object and the car. In the case that the car is nearing a back object while reversing an amplified light is triggered through another micro controller-based system. Another master designed for controlling the Car doors. The lighting system is controlled based on the status of doors (open or closed) or the longevity of the lights emitted.

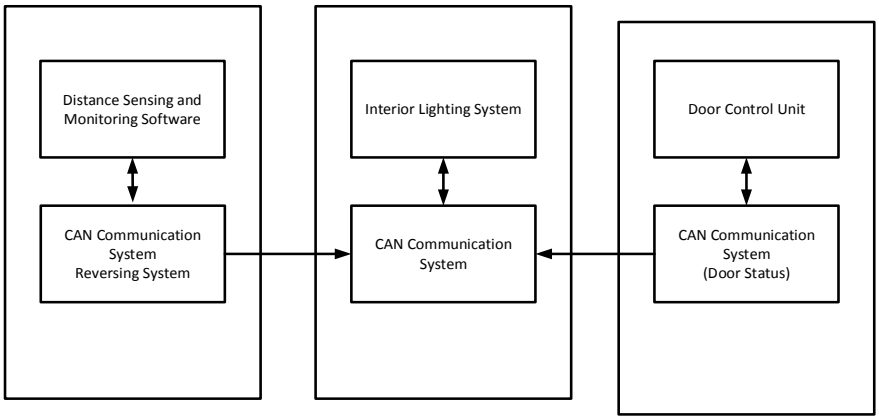

Fig. 4. Software architecture for CAN system.

4.3.Communication requirements for bridging the heterogeneous networks

While both the subsystems implemented, it is noticed that there existed a dependency between both the subsystems. The need to open the car doors as the temperature of the car engine raise beyond a threshold limit and the need to braking while the car is reversing.

These dependencies lead to the development of a "BRIDGE" between both the networks so that data can move either way. This requirement necessitated the

development of a device that is common to both the networks through which, the data flow can be achieved both ways and use the same for controlling and actuating purpose. Table 6 below shows the data flow requirements of the composite networked embedded system built using two subnets while one being I^2C based, the other developed based CAN protocol. There could be many more such data flow requirements, as stated in Table 6 below: Figure 5 shows the Bridge networking topology.

Table 6. Functional requirements of bridge.

Data flow serial	From network	To network	Purpose
1	I^2C	CAN	To receive the temperature data from an engine monitoring system for controlling the doors accordingly
2	CAN	I^2C	To send the distance data while car reversing for actuating breaking

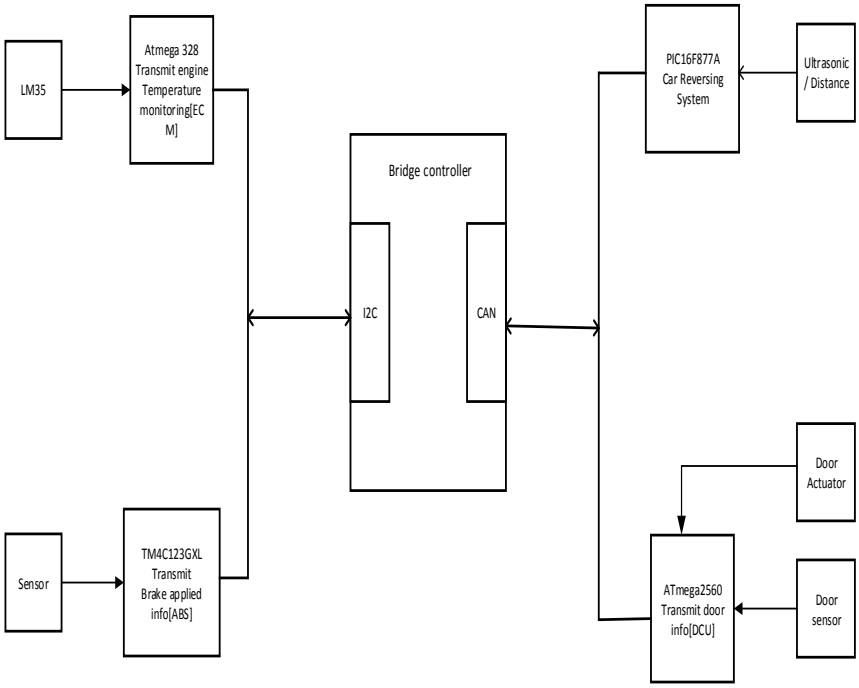

Fig. 5. Bridge topology.

5. Investigations and findings

The design and implementation of the bridge involve many factors, which include word addressing, conversion of numbering systems, application-specific message flow system, synchronisation of application-specific messages mapping to bus arbitration system, error detection and control, the timing of the process involved in sensing and actuating implemented on the two sides of the network. Response time management, data pocketing and de-pocketing, management of data transfer rates.

Heterogeneity due to endian handled through converting big-endian to small endian and vice versa. Converting from one number system to others is another heterogeneity issue that one must handle. Every application needs that messages must flow across as per the priority attached to the message. The priority as such is attached based on the criticality of the message. The message flowing across must match the bus arbitration system so that the device receives the most important message that has higher priority

Error detection and correction process implemented in different layers of each communication system. There is no uniformity as such. Transformations and translations required for building error detection and correction method implemented in one layer of communication protocol into another layer of a different communication protocol. Data transfer rates differ between two different communication protocols such as I^2C and CAN. In I^2C data can be communicated at 100 kbps, 400 kbps, 1 Mbps, 3.4 Mbps and in the case of CAN communication speeds achieved ranging from 40 kbps to 1 Mbps

One of the important aspects to be achieved is timing and completing the tasks within acceptable response times. Sensing data from a system in a network must be sent to another system in a different network for controlling a process parameter. The entire process from sensing to actuating completed within accessible responsible times. The data communication protocols defined for I^2C and CAN are different. The data packet design and the type of data packets that must be transmitted vary greatly considering both the protocols. There is a great variance in the speed of transmission of data. The data processing in the reception and the transmission side also differ greatly. Many heterogeneity issues need consideration at the reception and transmission side of the communication system, some of which, include word addressing, number conversions, endian, parity, error detection, and control.

In this paper, a novel data flow and synchronization method considering the communication systems implemented within the sub-systems presented that considers the issues of heterogeneity. The method attempts to solve the major problem of matching the speeds of data communication and synchronization between the reception and transmission sides and vice versa.

5.1. Matching data transfer speeds

Three different data transfer rates are achieved through I^2C network @ 100 kbps, 400 kbps, 1 Mbps, and 3.4 Mbps speeds and similarly different transfer speeds achieved through CAN, which includes 40 kbps to 1 Mbps. One of the major considerations is to select two speeds from the speeds supported by I^2C and CAN such that the data transfer synchronized without any delay caused between reception and transmission. An analysis of different speeds of I^2C with speeds supported by CAN considering maximum data packet size supported by either of the protocols carried.

Analysis carried by changing various parameters like the number of bytes, speeds, and the time per packet for storing or retrieving. Total time taken per packet computed by taking the number of bytes transmitted say 4, 5,6,7,8 bytes and fixing the speed for I^2C protocol and varying the speeds of the CAN protocol. Similarly, by fixing the speed of CAN and varying the speeds of I^2C

total time taken per packet is computed and plotted-five hundred packets considered for transmission and reception. Analysis has been carried considering different CAN speeds (125 kbps, 250 kbps, 500 kbps, 1 Mbps) keeping the speed on the I^2C side being 1 Mbps and considering data packet sizes. The data rates computed in terms of milliseconds.

5.2. Analysis of data flow-I^2C side reception-CAN transmission

Data analysis is carried by fixing I^2C speed at a constant rate and carrying transmission on CAN side at different rates, varying the data packet sizes. The details of data transmission rates considering I^2C speed of 1 Mbps and CAN speed of 1mbps, 500 kbps, 250 kbps, 125 kbps are varying the data packet size from 4 bytes to 8 bytes shown in Table 7, and the behaviour of the same shown in Fig. 6. From the figure, one can see that none of the CAN speed with varying data packet size converges to the data transmission speeds of I^2C.

The details of data transmission times considering I^2C speed of 400 kbps and CAN speed of 1 Mbps, 500 kbps, 250 kbps, 125 kbps varying the data packet size from 4 bytes to 8 bytes shown in Table 8, and the behaviour of the same shown in Fig. 7. From the figure, one can see that none of the CAN speed with varying data packet size converges to the data transmission speeds of I^2C. The details of data transmission times considering I^2C speed of 100 kbps and CAN speed of 1 Mbps, 500 kbps, 250 kbps, 125 kbps varying the data packet size from 4 bytes to 8 Bytes shown in Table 9, and the behaviour of the same shown in Fig. 8. From the figure, one can see that none of the CAN speed with varying data packet size converges to the data transmission speeds of I^2C.

From the above analysis, one can conclude that none of the CAN speeds converges to fixed I^2C transmission speed even when data packet sizes varied.

Table 7. Data transfer time @ data transmission rates: CAN @ 1 Mbps, 500 kbps, 250 kbps, 125 kbps, and I^2C @1 Mbps.

Number of bytes	Speed of I^2C 1 Mbps	Speed of CAN 1 Mbps	Speed of CAN 500 kbps	Speed of CAN 250 kbps	Speed of CAN 125 kbps
4	0.129	0.172	0.265	0.451	0.822
5	0.137	0.180	0.281	0.482	0.884
6	0.145	0.188	0.296	0.513	0.947
7	0.153	0.196	0.312	0.544	1.009
8	0.161	0.204	0.328	0.576	1.072

Table 8. Data transfer time @ data transmission rates: CAN @ 1 Mbps, 500 kbps, 250 kbps, 125 kbps, and I^2C @ 400 kbps.

Number of bytes	Speed of I^2C 400 kbps	Speed of CAN 1 Mbps	Speed of CAN 500 kbps	Speed of CAN 250 kbps	Speed of CAN 125 kbps
4	0.204	0.172	0.265	0.451	0.822
5	0.224	0.180	0.281	0.482	0.884
6	0.243	0.188	0.296	0.513	0.947
7	0.263	0.196	0.312	0.544	1.009
8	0.282	0.204	0.328	0.576	1.072

Table 9. Data transfer time @ data transmission rates: CAN @ 1 Mbps, 500 kbps, 250 kbps, 125 kbps, and I²C @100 kbps.

Number of bytes	Speed of I²C 100 kbps	Speed of CAN 1 Mbps	Speed of CAN 500 kbps	Speed of CAN 250 kbps	Speed of CAN 125 kbps
4	0.578	0.172	0.265	0.451	0.822
5	0.656	0.180	0.281	0.482	0.884
6	0.734	0.188	0.296	0.513	0.947
7	0.812	0.196	0.312	0.544	1.009
8	0.890	0.204	0.328	0.576	1.072

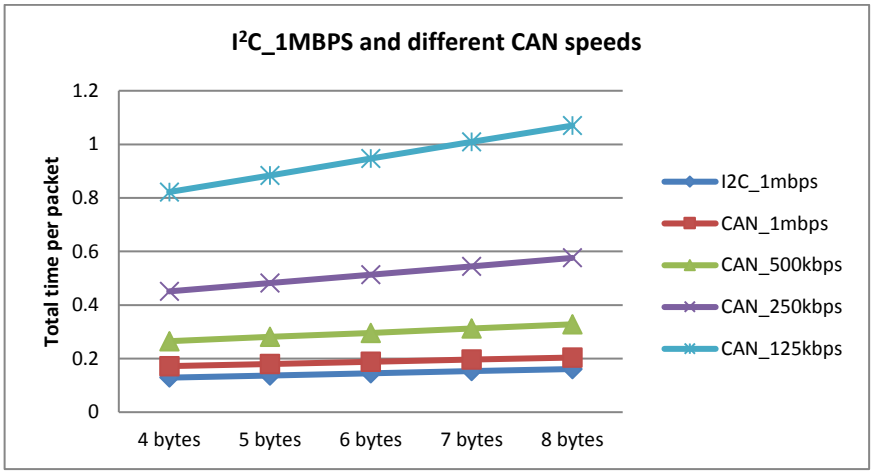

Fig. 6. Variance of data transmission times keeping I²C at a constant rate and varying CAN data rates and data packet sizes.

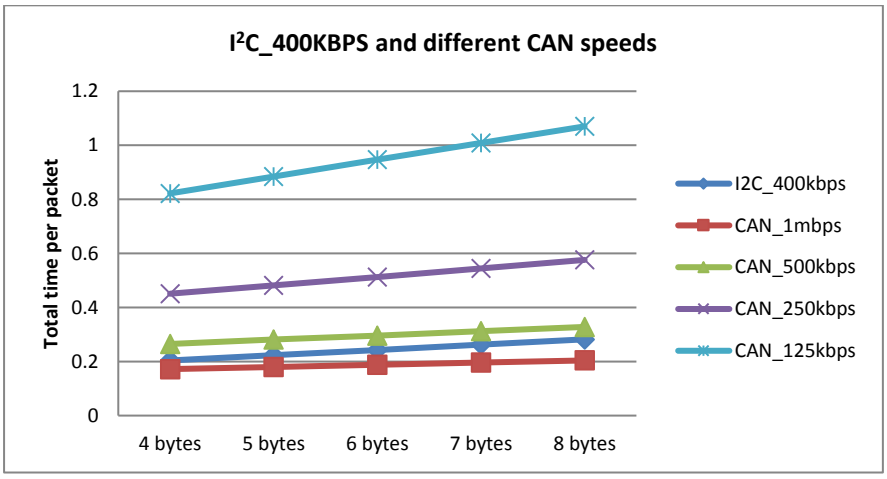

Fig. 7. Variance of data transmission times keeping I²C at a constant rate and varying CAN data rates and data packet sizes

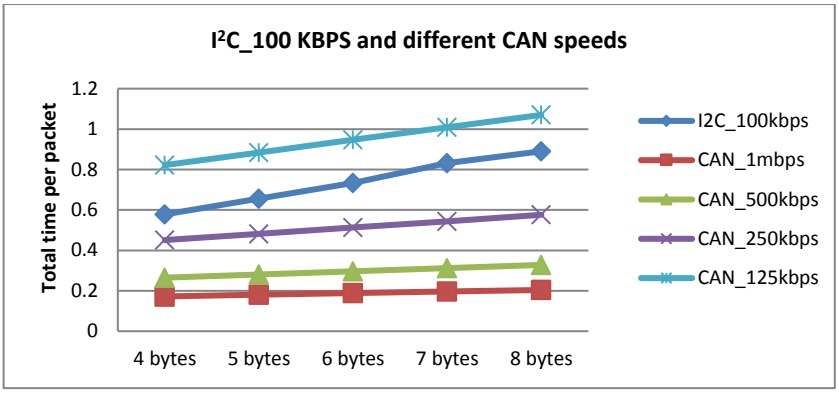

Fig. 8. Variance of data transmission times keeping I²C at a constant rate and varying CAN data rates and data packet sizes.

5.3. Analysis of data flow-CAN side reception-I²C Transmission

Data analysis is carried by fixing CAN speed at a constant rate and carrying transmission on I²C side at different rates, varying the data packet sizes. The details of data transmission times considering CAN speed of 40 kbps, and I²C speed of 100 kbps, 400 kbps, 1 Mbps varying the data packet size from 4 bytes to 8 bytes shown in Table 10, and the behaviour of the same shown in Fig. 9. From the figure, one can see that none of the I²C speeds with varying data packet size converges to the data transmission speeds of CAN.

Table 10. Data transfer time @ data transmission rates: I²C @ 1 Mbps, 400 kbps, 100 kbps and CAN @40 kbps.

Number of bytes	Speed of CAN 40 kbps	Speed of I²C 100 kbps	Speed of I²C 400 kbps	Speed of I²C 1 Mbps
4	2.399	0.578	0.204	0.129
5	2.594	0.656	0.224	0.137
6	2.789	0.734	0.243	0.145
7	2.980	0.812	0.263	0.153
8	3.180	0.890	0.282	0.161

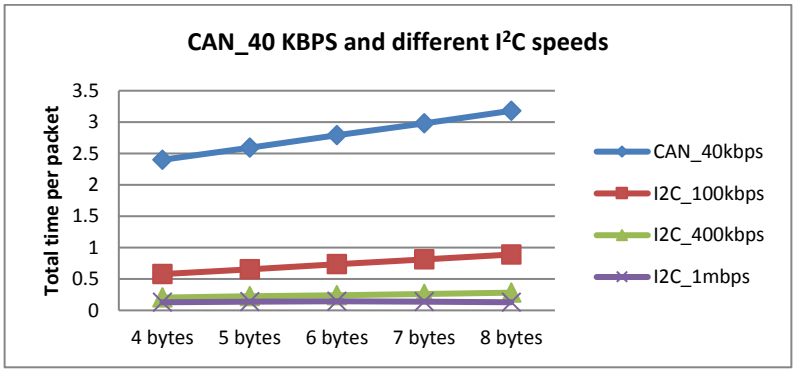

Fig. 9. Variance of data transmission times keeping CAN at a constant rate and varying I²C data rates and data packet sizes.

The details of data transmission times considering CAN speed of 125 kbps and I^2C speed of 100 kbps, 400 kbps, 1 Mbps varying the data packet size from 4 bytes to 8 bytes shown in Table 11, and the behaviour of the same shown in Fig. 10. From the figure, one can see that none of the I^2C speeds with varying data packet size converges to the data transmission speeds of CAN.

The details of data transmission times considering CAN speed of 250kbps and I^2C speed of 100 kbps, 400 kbps, 1 Mbps varying the data packet size from 4 bytes to 8 bytes shown in Table 12, and the behaviour of the same shown in Fig. 11. From the figure, one can see that none of the I^2C speeds with varying data packet size converges to the data transmission speeds of CAN.

The details of data transmission times considering CAN speed of 500 kbps and I^2C speed of 100 kbps, 400 kbps, 1 Mbps varying the data packet size from 4 bytes to 8 bytes shown in Table 13, and the behaviour of the same shown in Fig. 12. From the figure, one can see that none of the I^2C speeds with varying data packet size converges to the data transmission speeds of CAN.

Table 11. Data transfer time @ data transmission rates: I^2C @ 1 Mbps, 400 kbps, 100 kbps, and CAN @125 kbps.

Number of bytes	Speed of CAN 125 kbps	Speed of I^2C 100 kbps	Speed of I^2C 400 kbps	Speed of I^2C 1 Mbps
4	0.822	0.578	0.204	0.129
5	0.884	0.656	0.224	0.137
6	0.947	0.734	0.243	0.145
7	1.009	0.812	0.263	0.153
8	1.072	0.890	0.282	0.161

Table 12. Data transfer time @ data transmission rates: I^2C @ 1 Mbps, 400 kbps, 100 kbps and CAN @ 250 kbps.

Number of bytes	Speed of CAN 250 kbps	Speed of I^2C 100 kbps	Speed of I^2C 400 kbps	Speed of I^2C 1 Mbps
4	0.451	0.578	0.204	0.129
5	0.482	0.656	0.224	0.137
6	0.513	0.734	0.243	0.145
7	0.544	0.812	0.263	0.153
8	0.576	0.890	0.282	0.161

Table 13. Data transfer time @ data transmission rates: I^2C @ 1 Mbps, 400 kbps, 100 kbps and CAN @ 500 kbps.

Number of bytes	Speed of CAN 500 kbps	Speed of I^2C 100 kbps	Speed of I^2C 400 kbps	Speed of I^2C 1 Mbps
4	0.265	0.578	0.204	0.129
5	0.281	0.656	0.224	0.137
6	0.296	0.734	0.243	0.145
7	0.312	0.812	0.263	0.153
8	0.328	0.890	0.282	0.161

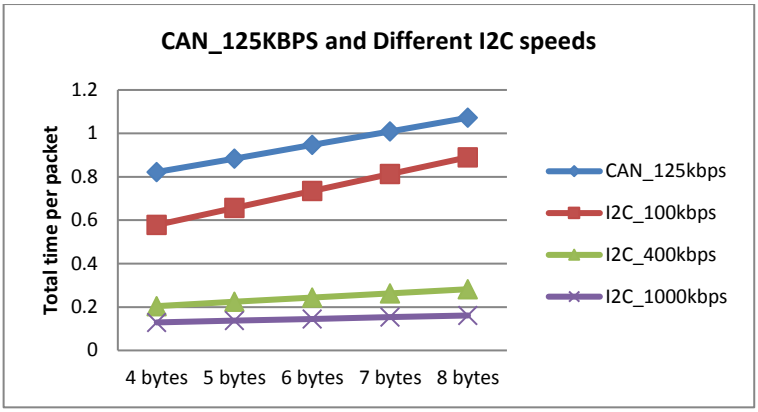

Fig. 10. Variance of data transmission times keeping CAN at a constant rate and varying I²C data rates and data packet sizes.

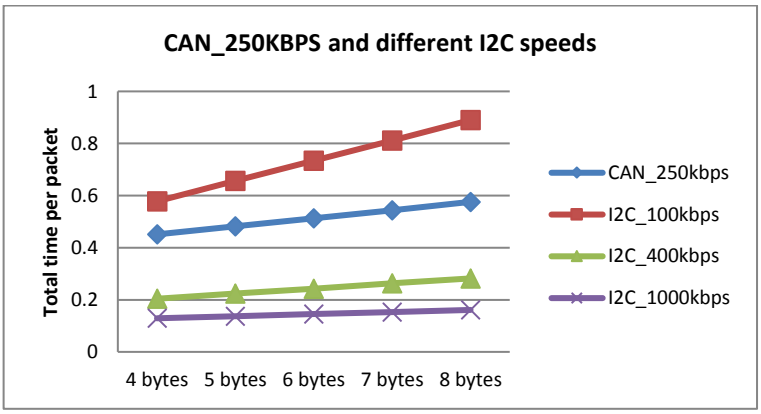

Fig. 11. Variance of data transmission times keeping CAN at a constant rate and varying I²C data rates and data packet sizes.

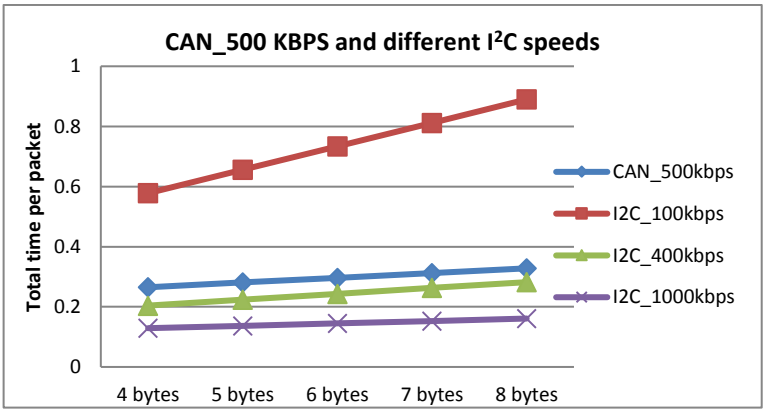

Fig. 12. Variance of data transmission times keeping CAN at a constant rate and varying I²C data rates and data packet sizes.

The details of data transmission times considering CAN speed of 1mbps and I^2C speed of 100 kbps, 400 kbps, 1 Mbps varying the data packet size from 4 bytes to 8 bytes shown in Table 14, and the behaviour of the same shown in Fig. 13. From the figure, one can see that none of the I^2C speeds with varying data packet size converges to the data transmission speeds of CAN.

Table 14. Data transfer time @ data transmission rates: I^2C @ 1 Mbps, 400 kbps, 100 kbps and CAN @1 Mbps.

Number of bytes	Speed of CAN 1 Mbps	Speed of I^2C 100 kbps	Speed of I^2C 400 kbps	Speed of I^2C 1 Mbps
4	0.172	0.578	0.204	0.129
5	0.180	0.656	0.224	0.137
6	0.188	0.734	0.243	0.145
7	0.196	0.812	0.263	0.153
8	0.204	0.890	0.282	0.161

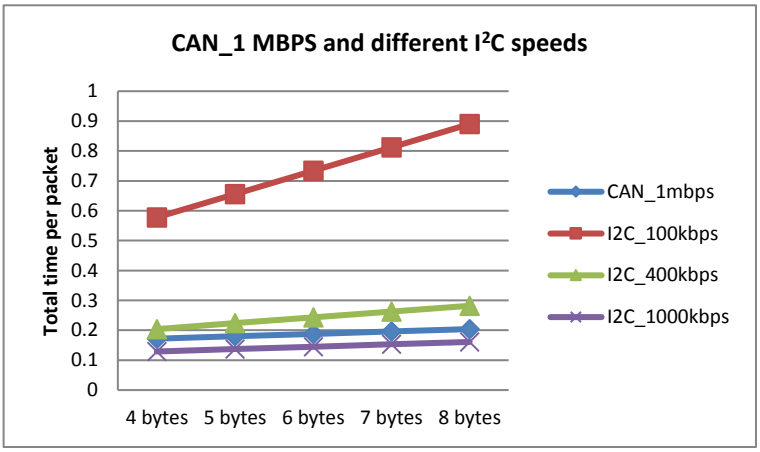

Fig. 13. Variance of data transmission times keeping CAN at a constant rate and varying I^2C data rates and data packet sizes.

5.3. Data buffering within the bridge

From the data analysis presented in section 5.2 and 5.3 that varying either the CAN speeds or I^2C speeds and also varying the data size, no time matching without any time delay achieved considering any combination of data transmission speeds and packet sizes. It is obvious from the analysis that there will be some delays caused between the reception and delay and there should be a proper buffering strategy to minimize or eliminate time delay between data reception and transmission and vice versa considering both sides of CAN and I^2C. For achieving time convergence, the next strategy could be considering the multiplying effect, which means considering the data transmission speeds that lead to the timing of one in terms of multiple of the time of the other even considering the variance in the data size. The details of a multiplicity of timing considering I^2C to CAN and vice versa are shown in Tables 15 to 18.

5.4.1. I^2C side reception and CAN side transmission

One of the best strategies is to select the communication speeds of the transmission and reception in such a way that time taken to receive and transmit the same failing which, one must try multiple of times of the other. Reviewing the details provided in Tables 15 to 18 considering I^2C side reception and CAN side transmission, the minimum delay of 0.043 milliseconds caused when the transmission speed set to 1 Mbps. However, no guarantee assured that the transmission speeds would be exact due to several reasons.

Table 15. Transmission timing: CAN (1 Mbps), I^2C (1 Mbps).

Number of bytes	Speed of I^2C	Time taken receiving into buffer (ms)	Speed of CAN	Time taken for transmission from the buffer (ms)	Delay	Multiples
4	1 Mbps	0.129	1 Mbps	0.172	0.043	-
5	1 Mbps	0.137	1 Mbps	0.180	0.043	-
6	1 Mbps	0.145	1 Mbps	0.188	0.043	-
7	1 Mbps	0.153	1 Mbps	0.196	0.043	-
8	1 Mbps	0.161	1 Mbps	0.204	0.043	-

Table 16. Transmission timing: CAN (500 kbps), I^2C (1 Mbps).

Number of bytes	Speed of I^2C	Time for receiving into the buffer (ms)	Speed of CAN	Time taken for transmission from the buffer (ms)	Delay	Multiples
4	1 Mbps	0.129	500 kbps	0.265	0.136	0.129*2=0.258
5	1 Mbps	0.137	500 kbps	0.281	0.144	0.137*2=0.274
6	1 Mbps	0.145	500 kbps	0.296	0.151	0.145*2=0.290
7	1 Mbps	0.153	500 kbps	0.312	0.159	0.153*2=0.306
8	1 Mbps	0.161	500 kbps	0.328	0.167	0.161*2=0.322

Table 17. Transmission timing: CAN (125 kbps), I^2C (500 kbps).

Number of bytes	Speed of I^2C	Time to receive into the buffer (ms)	Speed of CAN	Time to transmit from the buffer (ms)	Delay	Multiples
4	500 kbps	0.204	125 kbps	0.822	0.618	0.204*4=0.816
5	500 kbps	0.224	125 kbps	0.884	0.660	0.224*4=0.896
6	500 kbps	0.243	125 kbps	0.947	0.704	0.243*4=0.972
7	500 kbps	0.263	125 kbps	1.009	0.746	0.263*4=1.052
8	500 kbps	0.282	125 kbps	1.072	0.790	0.282*4=1.120

Table 18. Transmission timing: CAN (1 Mbps), I^2C (100 kbps).

Number of bytes	Speed of I^2C	Time to receive into the buffer (ms)	Speed of CAN	Time for transmission from the buffer (ms)	Delay	Multiples
4	100 kbps	0.578	1 Mbps	0.822	0.244	-
5	100 kbps	0.656	1 Mbps	0.884	0.228	-
6	100 kbps	0.734	1 Mbps	0.947	0.213	-
7	100 kbps	0.812	1 Mbps	1.009	0.197	-
8	100 kbps	0.890	1 Mbps	1.072	0.182	-

The second approach is to select the transmission speeds such that the response time for transmission and reception will be multiples of each other in which, case

the delay time eliminated by introducing the buffers equivalent to the multiplying factor. For example, if the multiplying factor is two, then two buffers can be used. Separate software architecture followed such that the Input buffers when get filled, the same is emptied concurrently by using multiple transmission processes. The transmission speed CAN (500 kbps), and I^2C (1 Mbps) lead to multiples of 2 while the transmission speed: CAN (125 kbps), I^2C (500 kbps) leads to multiple of 4. In any case, the delay as such eliminated by using specific software architecture.

Figure 14 shows the proposed software architecture of the Bridge. The receiving side process keeps receiving the data and fills up the buffers serially one after the other in a cyclic fashion. Simultaneously, several transmitting processes equivalent to the number of buffers will concurrently read the data and place the same to output buffers. Further, a separate process continually and serially streams the data into output transmission lines, thereby eliminating any delay that exists between the reception and transmission.

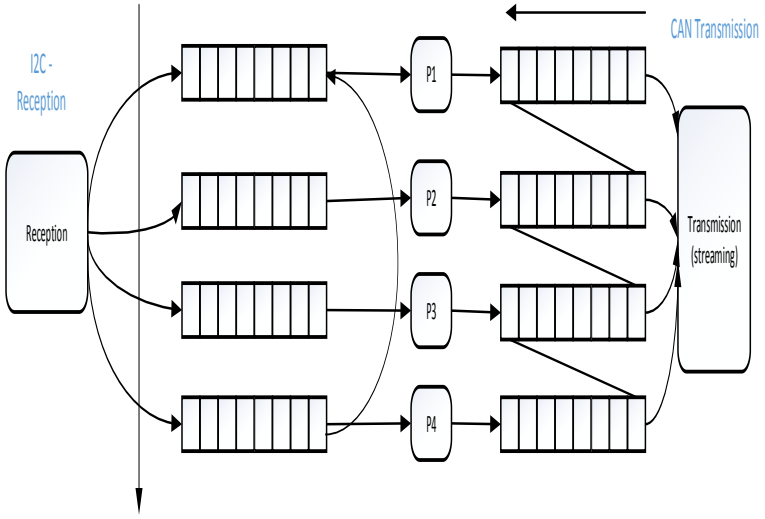

Fig. 14. Software architecture of bridge.

5.4.2. CAN side reception and I^2C side transmission

The above reception and transmission process equally applicable even in the case when CAN is on receiving side and I^2C is on the transmission side.

6. Conclusions

Composite embedded systems generally developed as the time evolves and through the interconnection of several sub-nets implemented through different communication systems. In such a case, networking becomes quite complex due to the existence of heterogeneity, especially selecting communication speeds. There is a requirement of reducing the time delay between the reception and transmission so that the network will like continuous traffic without any time delay.

It is not possible to match the speeds of the sub-nets contained in a composite network without causing a time delay. Multiplicity effect of the communication speeds can be conveniently used to match the transmission rates and to reduce the latency that exists between the reception and transmission. The multiplicity effect can be tackled through the concurrent transmission process and streaming the output into output channels continuously without the need for either stopover or wait times. An intelligent bridge that dynamically gets configured by choice of proper communication speeds and use of local buffers proposed and presented in this paper makes the heterogeneous sub-nets get interconnected seamlessly as if a single homogenous network is functioning

The minimum delay time is 0.129 milliseconds considering any combination of speeds selected for effecting the communication through CAN and I^2C. The delay time is negligible when the speed of CAN is multiples of I^2C and vice versa. When 1 Mbps speed selected for both CAN and I^2C, the delay is 0.043 milliseconds, which is the least delay caused. However, in practice, it is quite tough to match exact speeds.

The transmission speed CAN (500 kbps), and I^2C (1Mbps) lead to multiples of 2 while the transmission speed: CAN (125 kbps), I^2C (500 kbps) leads to multiple of 4. The delay caused when multiple of 2 is selected is 0.136 milliseconds while 0.618 milliseconds of delay caused when multiple of 4 is selected. The delay as such, avoided by selecting as many concurrent processes for transmission equivalent to the number of buffers chosen. The number of buffers the same as the multiples is the best choice.

Abbreviations

CAN	Controller Area Network
ES	Embedded System
GPS	Global Positioning System
I^2C	Inter-Integrated Circuit
IIOT	Industrial Internet of Things
IoT	Internet of Things
RS232C	Standard Serial Interface
RS485	Standard defining the electrical characteristics of drivers and receivers for use in serial communications
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
VLSI	Very Large-Scale Integration

References

1. Cao, J.; and Nymeyer, A. (2009). Formal model of a protocol converter. *Proceedings of Fifteenth Australasian Theory Symposium on Computing (CATS' 09)*. Wellington, New Zealand, 109-120.
2. Guohuan, L.; Hao, Z.; and Wei, Z. (2009). Research on designing method of CAN bus and Modbus protocol conversion interface. *Proceedings of the International Conference on Future on BioMedical Information Engineering (FBIE)*. Sanya, China, 180-182.

3. Wang, X.; and Guo, W. (2009). The design of RS232 and CAN protocol converter based on PIC MCU. *Computer and Information Science*, 2(3), 176-181.
4. Guohuan, L.; Haiting, C.; and Shujie, L. (2010). Research and implementation of ARM-based fieldbus protocol conversion method. *Proceedings of the International Conference on Computer and Communication Technologies in Agriculture Engineering*. Chengdu, China, 260-262.
5. Zhang, H.; Li, Y.; and Zhu, H. (2011). Development for protocol conversion gateway of Profibus and Modbus. *Procedia Engineering*, 15, 767-771.
6. Benachinamardi, A.K.; and Wali, U.V. (2011). Design of CAN transmitter with an I^2C interface. *International Journal of Computer Applications*, 46(21), 6-10.
7. Hui, L.; Hao, Z.; and Daogang, P. (2012). Design and application of communication gateway of EPA and MODBUS on electric power system. *Energy Procedia,* 17(Part A), 286-292.
8. Zhang, F.; Zhu, Y.; Yan, C.; Bi, J.; Xiong, H.; and Yuan, S. (2013). A realization method of protocol conversion between Modbus and IEC61850. *Open Journal of Applied Sciences*, 3(2), 18-23.
9. Kiran, V.; and Vinilanagraj. (2013). Design of SPI to I^2C protocol converter and implementation of low power techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(10), 3770-3774.
10. Kaur, J.; and Singh, M. (2013). Multiprotocol gateway for wireless communication in embedded systems. *International Journal of Computer Applications,* 72(18), 27-31.
11. Sastry, J.K.R.; Suresh, A.; and Sasi Bhanu, J. (2015). Building heterogeneous distributed embedded systems through RS485 communication protocol. *ARPN Journal of Engineering and Applied Sciences,* 10(16), 6793-6803.
12. Sastry, J.K.R; Ganesh, J.V.; and Bhanu, J.S. (2015). I^2C based networking for implementing heterogeneous microcontroller based distributed embedded systems. *Indian Journal of Science and Technology*, 8(15), 1-10.
13. Sastry, J.K.R.; Lakshmi, M.V.; and Bhanu, S.J.S. (2015). Optimizing communication between heterogeneous distributed embedded systems using CAN protocol. *ARPN Journal of Engineering and Applied Sciences*, 10(18), 7900-7911.
14. Shwetha, S.; and Karunavathi, R.K. (2016). The design of multiprotocol interface device. *Proceedings in 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. Bangalore, India, 474-476.
15. Narayanan, R.; and Murthy, C.S.R. (2017). Information: A probabilistic framework for protocol conversions in IIoT networks with heterogeneous gateways. *IEEE Communications Letters,* 21(11), 2456-2459.
16. Murty, A.S.R.; Teja, K.; and Naveen, S. (2018). Lathe performance monitoring using IoT. *International Journal of Mechanical Engineering and Technology (IJMET),* 9(4), 494-501.
17. Rambabu, K.; and Venkatram, N. (2018). Traffic flow features as metrics (TFFM): Detection of application layer level DDOS attack scope of IoT traffic

flows. *International Journal of Engineering and Technology (UAE)*, 7(2.7), 203-208.

18. Manasa, K.V.; Prabu, A.V.; Prathyusha, M.S.; and Varakumari, S. (2018). Performance monitoring of UPS battery using IoT. *International Journal of Engineering and Technology*, 7(2.7), 352-355.
19. Poonam, J.S.; Pooja, S.; Sripath Roy, K.; Abhilash, K.; and Arvind, B.V. (2018). Implementation of asymmetric processing on multi-core processors to implement IOT applications on GNU/Linux framework. *International Journal of Engineering and Technology*, 7(2.7), 710-713.
20. Naidu, G.R.; and VenkatRam, N. (2018). Urban climate monitoring system with IoT data analytics. *International Journal of Engineering and Technology*, 7(2), 5-9.
21. Gupta, P.; Satyanarayan, K.V.V.; and Shah, D.D. (2018). Development and testing of message scheduling middleware algorithm with SOA for message traffic control in IoT environment. *International Journal of Intelligent Engineering and Systems*, 11(5), 301-313.
22. Gupta, P.; Satyanarayan, K.V.V.; and Shah, D.D. (2018). IoT multitasking development of hybrid execution service-oriented architecture (HESOA) to reduce response time for IoT application. *Journal of Theoretical and Applied Information Technology*, 96(5), 1398-1407.
23. Yasaswini, A.; DayaSagar, K.V.; ShriVishnu, K.; Hari Nandan, V.; and Prasadara Rao, P.V.R.D. (2018). Automation of an IoT hub using artificial intelligence techniques. *International Journal of Engineering and Technology*, 7(2.7), 25-27.
24. Ramaiah, C.H.; Parimala, V.S.; Kumar, S.P.; Reddy, G.; and Rahul, Y. (2018). Remote monitoring through a tab. *International Journal of Mechanical Engineering and Technology,* 9(1), 490-498.
25. Sai, Y.S.; and Kumar, K.K. (2018). Internet of things and its applications. *International Journal of Engineering and Technology*, 7(2.7), 422-427.