On Developing High-Speed Heterogeneous and Composite ES Network through Multi-Master Interface

J Rajasekhar[1], Dr. JKR Sastry[2]
Koneru Lakshmaiah Education foundation Vaddeswaram

***Abstract*—These days, many heterogeneous and composite embedded systems contain many subnets developed using different bus-based protocols, such as I2C, CAN, USB, and RS485. There is always a requirement to Interface and interconnect the heterogeneous ES networks to achieve and establish a composite network. The ES networks developed using different protocols differ in many ways, considering the speed of communication, Arbitration, Synchronization, and Timing. Many solutions are being offered using heterogeneous embedded systems, especially in implementing automation systems, without addressing integration and proper interfacing. In this paper, a Multi-Master based interfacing of a CAN and I2C networking through Ethernet-based interfacing has been presented especially to find the optimum speeds at which the networks must be operated for different data packet sizes. It has been shown in the paper that it is quite efficient and effective when a data packet of size 40 bytes is driven using an I^2C speed of 5120 bits, Ethernet speed of 20480 bits, and CAN speed of 500 bits.**

Keywords*—Embedded systems; embedded networks; hybridization of embedded networks; hybridizations through multi-master communication

I. Introduction

Many kinds of embedded networks are in existence and in use for implementing different types of Applications. The Most important Embedded System networking being in use includes the networks built using the communication systems that Include I2C, CAN, RS485, USB. But as the technologies are emerging, a necessity arises that require bridging the ES networks built around ES networking standards.

All ES networking standards differ in many ways: network termination, device identification, type and format of data packets, type of signals used, and communication speeds.

Hybridization of Embedded Networks can be achieved through different interconnecting types of wired networks that include I2C, CAN, USB, and RS48. Serial communication takes place among the hybridized networks. Hybridization can also be achieved through establishing wireless networks or through a combination of wireless and wired networks. The major issue in such networking is the management of communication speeds and data rates. May architectures can achieve the networks' hybridization, including single master integrations, multi-master integrations, a hardware-based bridge, multi-master integration, etc. The way a hybridized system works is dependent on the type of hybridization method used. As of date, hybridization through a Bridge device is proposed by [17].

Hybridization can be achieved through other methods that include Single master catering for the communication, using a Multi-master interface, and developing a Universal bus that caters to most of the ES-based communication Standards. In this paper, hybridization through Multi-master Interface has been presented.

A. Motivation

The interfacing of the heterogeneous ES networks is necessary these days as many Systems developed using different networking protocols need to be interfaced and interconnected. The response time does not suffer. The electronics industry, as such, needs these kinds of solutions.

B. Rest of the Coverage in the Paper

In the rest of the paper, in Section 2, a review of the contributions made in the research related areas has been presented. In Section 3, application development using CAN and I^2C communication systems has been presented along with a comparison that shows how the networking systems differ and the issues related to interfacing between the communication systems. In Section 4, an architecture that focuses on the I^2C and CAN networks' hybridization has been presented. In Section 5, a computation method for determining optimum speeds of I^2C, CAN, and Ethernet has been presented, and conclusions have been drawn in Section 6.

II. Related Work

The speed of communication through I^2C and RS23 gets et influenced because of the complex electromagnetic conduit. Such a domain will result in the inappropriate disposal of the signals to the interfaces. Inappropriate transfer of electromagnetic will result in low speed and dependability. A few strategies are to be invented that help handle speed through legitimate cushion the executives [1].

Utilizing remote advances to improve conveyed implanted frameworks is testing contrasted with wired systems because of vulnerability and less unwavering quality caused by attractive obstruction, blurring, reflection, and so on. The consistency of remote correspondence is an issue. It turns out to be very unsafe for a framework when it needs to meet some basic security prerequisites by getting information through a remote system. The vulnerabilities existing with the remote correspondence can be settled through the utilization of mixed-

race models or undertaking hybridization of structural models. The idea of hybridization is increasingly common in the automotive area where everything must work consummately, disregarding un- sureness. Hybridization helps when the correspondence needs to occur in questionable circumstances. Engineering has been displayed that can be utilized to actualize applications requiring the idea of hybridization. Numerous Interface gives that must be considered and taken care of through a programming model that makes the framework hybridization-mindful [2].

Two gauges are utilized as often as possible. They incorporate field bus and CAN bus for executing industry-based applications; field bus benchmarks are not uniform and incredibly vary from industry to industry. Correspondence between the gadgets utilizing Fieldbus in that capacity is entangled. In industry, both the transport based systems administration frameworks are often utilized. This has prompted a prerequisite of converting one kind of correspondence to the next, which can be accomplished through convention transformation. Convention transformation can be planned and actualized at the equipment level [3].

CAN transport based correspondence can be utilized for systems administration with frameworks. The engineers need to comprehend the CAN convention, Interface, controller, and Physical associations before the applications can be created utilizing CAN-based correspondence. Actualizing CAN-based correspondence at the net root level is entangled and needs thorough testing. The advancement of utilizations utilizing CAN is made to be straight through the CAN module. The CAN is a convention suite that can be coordinated with any inserted framework Software. Sending and getting the information can be accomplished through the CAN module [4]. When CAN is to be utilized alongside other correspondence conventions, for example, I2C, protocol transformations can be actualized at work level rather than net root level. If another module that modified works I2C communication is grown, at that point, convention transformation can be accomplished at the work level.

A different gadget can be structured and built to do conventional change utilizing numerous Microcontroller based frameworks, fast double port RAM information sharing innovation, and continuous multi- entrusting framework C/OS-II. A convention converter that helps correspondence somewhere in the range of RS232C and RS485 has been built up to be actualized inside savvy instruments, information securing frameworks, etc. [5]. The gadget can be interfaced with a remote checking framework through Ethernet-based correspondence. Along these lines, sequential gadgets can be connected to the system control layer. This gadget built up along these lines has unwavering high quality and continuous execution and acknowledge information trade, information sharing, and data handling among various Microcontroller based frameworks

Field bus convention is nonstandard and has been actualized in various forms. There is additionally an issue of interconnecting distinctive field buses. At the point when two systems are manufactured utilizing distinctive Fieldbus correspondence, conventional transformation is required. ARM controller can be used for accomplishing the change. The convention transformation is accomplished through the advancement of a protocol utilizing a standard information parcel. The strategy isn't constrained to coordinated transformation and is free of the transport area and convention utilized by the field buses [6].

A USB and I2C convention varies in numerous angles, considering how the correspondence is attempted. The information bundle groups, length of the system, number of gadgets that can be associated, number of ace, transport discretion, synchronization strategy, stream control, and so on vary greatly. A mapping between

I2C and USB have been done both at the equipment and product level. The product structure that can be utilized for accomplishing the change has been displayed [7].

Modbus and Profibus are two correspondence frameworks utilized for accomplishing modern mechanization. Both specialized strategies are generally utilized in the modern control field. Anyway, these two means of transport can't be associated straightforwardly because of the presence of extraordinary fluctuation between them. An entryway is required for interfacing two unique means of transport through which convention transformation can be conveyed. A passage is created utilizing the AT89C52 Microcontroller. Profibus and Modbus are two progressively basic mechanical field transport; they were generally utilized in the modern control field. Since the two means of transport can't inter-connect with one another, a Profibus and Modbus convention conversion is required. SPC3 is incorporated with the Micro Controller to achieve Profibus and Modbus conversions [8].

Numerous kinds of sub-frameworks are to be created and actualized, utilizing distinctive correspondence frameworks. For example, s Flight control, banking, therapeutic, and other high affirmation frameworks should be executed most unequivocally since the correspondence framework utilizes distinctive signaling, sheathing, commotion separating, signal disconnection, and so forth. One should structure and build up the framework so that one sub-framework doesn't meddle with the other.

Following a data stream at the equipment level is one strategy that can be utilized to distinguish and channel the differences. The door level in-arrangement stream following (GLIFT) framework is built to provide a technique for testing data streams inside I2C and USB. Time-division various access (TDMA) has been utilized that can confine a gadget on the BUS from the streams [9].

Mechanical Ethernet innovation (EPA) and Modbus correspondence innovation (MODBUS) are now and again utilized correspondence frameworks to actualize modern procedures. No immediate communication in that capacity can be conveyed in the middle of these two frameworks as there is no immediate similarity between them. A corresponding passage is created for accomplishing the necessary Interface between these two innovations. A passage has been created utilizing an ARM-based smaller scale controller and COS continuous working system. Bidirectional correspondence can be accomplished through the utilization of the entryway.

The correspondence entryway can give a steady, secure, constant, and adaptable answer to power plants [10].

Two kinds of industry explicit correspondence frameworks are utilized in the power segments intended for National power dispatching and control the breeze factories. To achieve integration, the framework theories correspondence frameworks must be interfaced with one another. A method has been developed to accomplish conventional transformation that executes capacities like interconnectivity, convention information type, configuration change, and scaling, information approval, the board of neighborhood/remote directions, recreation of information parcels transmission, etc. solicitations, communication bundles investigation, and repetitive correspondence joins [11].

Field buses are utilized for trading information between several microcontrollers and field gadgets by affecting communication among them. Numerous adaptations of the field bus communication frameworks exist. Structuring a correspondence framework for impacting correspondence among the gadgets that keep diverse Fieldbus correspondence benchmarks is mind-boggling and, by and large, prompts convoluted usage of equipment and programming. An effective correspondence interface is a requirement for executing a solid framework utilizing restrained field bus correspondence norms. CAN transport is additionally being utilized nowadays for actualizing a significant number of mechanical procedures. There is a need to interconnect between CANBUS and MODBUS. The correspondence conventions are to be mapped considering various parts of correspondence, and afterward, an interface is required to be created. An interface that associates both the transport based correspondence frameworks has been introduced [12][13].

The way networking of the embedded systems is carried on the kind of communication standard used I2C [14], USB [15], CAN [16] and RS485[17] and also the kind of Microcontroller based systems used for networking.

Many issues related to networking have been discussed [18][19][20][21][22][23][24] concerning testing Distributed Embedded Systems. Various methods and strategies have been proposed by Rajasekhar et al. [25] for hybridizing the networking of heterogeneous embedded networks. The interconnection between an I2C network and a CAN network can be achieved by developing a device that bridges both the networks. Speed Matching is one of the most important considerations that must be handled when it comes to the ES networks' hybridization [26]. Rajasekhar et al. have presented an efficient architecture that considers hybridization using Multi-master Interface [27], which is further extended in this paper to drive the architecture with the main considering driving through optimum speeds.

III. Application Development using Heterogeneous and Composite Embedded Network

A. Application Development using CAN Interface

An ES Application is developed through a CAN network, which is meant for monitoring and controlling temperature and humidity within an Engine.

CAN-based networking is archived through one Master and two slaves. The slaves are implemented through Arduino-UNO, and the Master is implemented through STEM 32. All three systems are connected through MCP 2515 CAN Module. The Master is connected to a SWITCH through its native Ethernet port for onward networking with another network developed using a different protocol such as I2C.

One of the slaves is connected with the DHT 11 sensor situated near an automobile system engine. The sensor continuously monitors the temperature of the engine and sends the information to the CAN master. CAN Master will send the information to the I2C Master through the Ethernet interface. I2C Master drives a different network based on its native protocol; The I^2C Master sends the data that it received from the CAN network to the I2C slave, a kind of actuator to cool through a FAN connected to the slave. The FAN is controlled through a relay system. Whenever the engine's temperature goes beyond a level, the FAN is switched on or otherwise switched off. The Functional requirements of CAN-based Application is shown in Table I.

The Networking Diagram for the Application is shown in Fig. 1.

CAN or Controller Area Network is a two-wired asynchronous, half-duplex fast sequential system-bus width of CAN is 217 bits. CAN is used for communication among devices in a closed distance, such as in a vehicle. CAN is based on CSMA-CD/ASM convention. CSMA guarantees that every hub must hang tight for a given period before sending any message. The crash location guarantees that the impact is kept away from choosing the messages dependent on their endorsed need. It gives a flagging rate from 125kbps to 1 Mbps. It accommodates 2048 diverse message identifiers.

TABLE I. CAN Application Description

Hardware Device Number	Hardware description	Interface description	Functional Description
1	STM32F401RE	CAN	To receive Temperature Data from Slave
			To receive Humidity Data from Slave
			To Send Temperature data to I^2C Master
			To Send Humidity data to I^2C Master
			To Receive Distance data from I2C Master
			To send distance data to the slave
2	ARDUINO UNO	CAN	To Receive distance data from Master
			To Control the Lighting System
3	ARDUINO UNO	CAN	To sense the temperature
			To sense the humidity
			To send Temperature to Master
			To send humidity to the Master

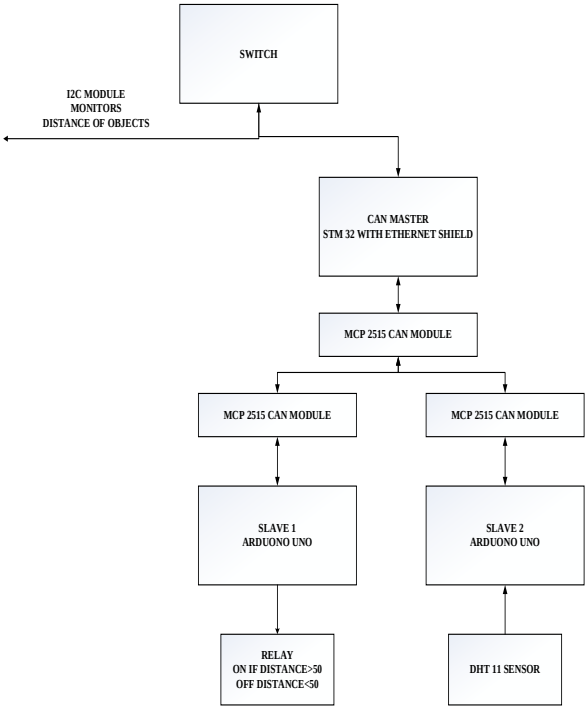

Fig. 1. CAN Networking Diagram.

CAN bus is the multi-master protocol. When the bus is idle, any device can be attached to the CAN bus and starts messaging. The can bus versatile, so devices attached to the bus do not have addressing. Each device in the CAN bus receives every message transmitted over the bus, and it is up to the device to decide whether to use the message that it receives or simply ignore it when the message is no more related to its own.

CAN bus provide remote transmission request (RTR), meaning that one node on the bus can request information from the other nodes. A request for information is sent to a node instead of waiting for a node to send information continuously. Any device in CAN bus can identify the error that occurred on the bus while transmitting the data and generates the error frames. The node which identifies the error alerts all other nodes about the error. There are no limitations for attaching and detaching the CAN bus devices, so devices are easy to attach and detach. Depending on the bus delays time and electrical loads, we can only decide the number of devices attached to the bus.

CAN protocol send messages in different types of data packets that include data frames, remote frames, error frames, and overload frames shown in Fig. 2, Fig. 3, Fig. 4, and Fig. 5. Data Frames are used to transmit data from Master to Slave and vice versa. Remote frames are used to seek permission from another node to transmit messages. Error frames are used for transmitting the errors that occur due to transmission, which can be classified as Bit errors, CRC errors, Form errors, Acknowledgement errors, and Stuffing errors. Overflow frames are used to create extra delays required for transmission of data or response.

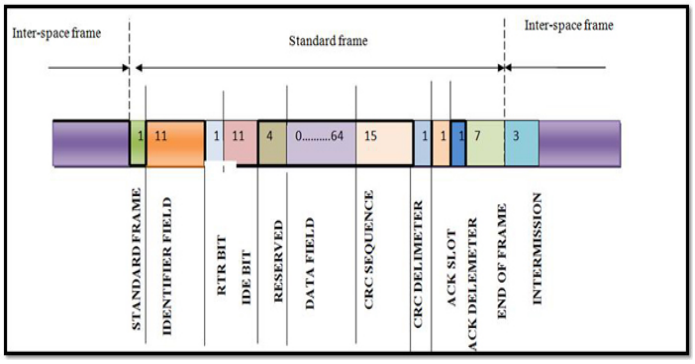

Fig. 2. Data Frame.

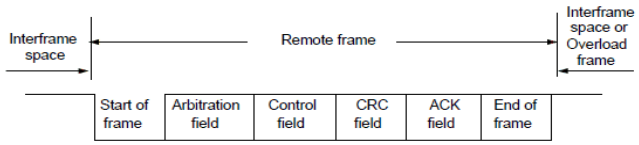

Fig. 3. Remote Frames.

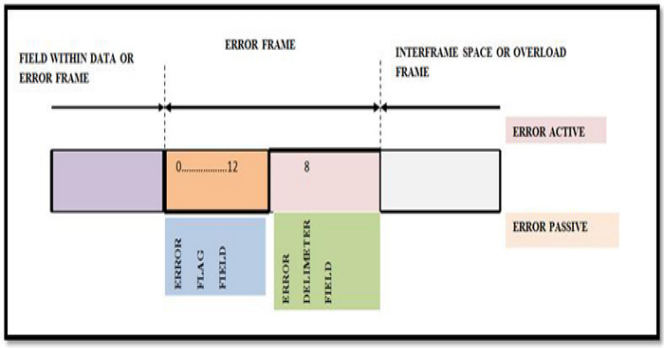

Fig. 4. Error Frame.

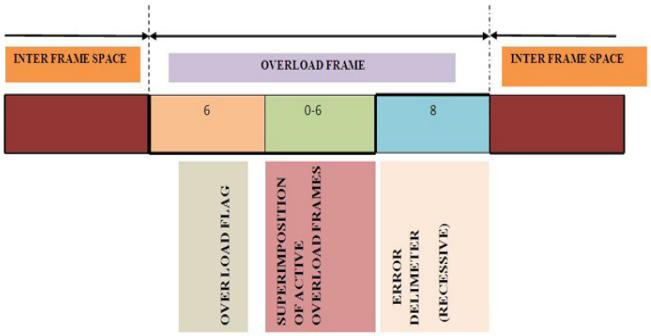

Fig. 5. Error Frame.

CAN-based communication between the Master and slave can be undertaken using speeds ranging from 125kbs to 1Mbps following the protocol sequences. The sequence in which the data packets are transmitted depends on the type of Application implemented over the network. The most appropriate choice of speed is the most important issue to be dealt with for realizing effective communication to take place among the heterogeneous network. The choice, however, needs to take into consideration many other important parameters.

B. Application Development using I2C Interface

Another I2C network is used as a subnetwork integrated into a composite network along with the CAN network. The ES Application developed through I2C Interface is related to measuring the distance of the objects Located behind a motor Vehicle and controlling the speed of a FAN fitted within the engine based on the engine's temperature, which is transmitted through the CAN network. The Application-specific functions implemented through I2C based networking is shown in Table II.

I2C Network is built with a single master and two slaves. One slave is connected with an ultrasonic sensor to monitor the nearest object's distance while the car is backing up. The monitored distances are sent to CAN master through Ethernet. The Second slave is connected with a DC Motor for controlling FAN's speed, which is connected to an engine. The FAN controlling is done based on the Temperature and Humidity data received from the CAN master. Two different protocols are used within I^2C, each for reading and writing data.

The networking Diagram for the I^2C based Application is shown in Fig. 6.

TABLE II. I2C APPLICATION DESCRIPTIONS

Hardware Device Number	Hardware description	Type of Device	Functional Description
1	STM32F401 RE	Master	To Receive Distance data from the slave
			To receive Temperature data from CAN Master
			To receive Humidity data from CAN Master
			To send Distance data to CAN Master
			To send Temperature data to the slave
			To send Humidity data to the slave
2	STM 32 F301 RE	Slave	To sense the distance of the object while reversing the car
			To send the Distance data to the Master
3	STM 32 F301 RE	Slave	To receive Temperature data from the Master
			To receive Humidity data from Master
			To actuate the DC motor for controlling the FAN

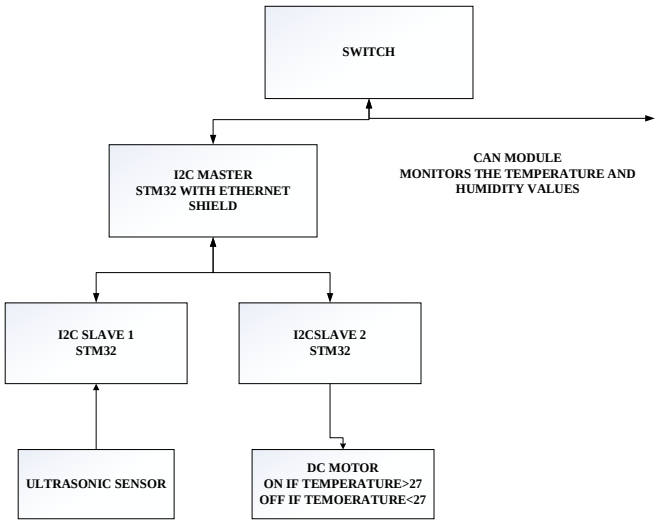

Fig. 6. I2C Networking Diagram.

The following sequence of operations is carried when data transmitted by a slave is to be read by the Master.

1) The master device sets the Read/Write bit to '1' instead of '0', which signals the targeted slave device that the master device is expecting data from it.

2) The slave device sends the 8 bits corresponding to the data block, and the master device sets the ACK/NACK bit.

3) Once the master device receives the required data, it sends a NACK bit. Then the slave device stops sending data and releases the SDA line.

4) Suppose the master device reads data from a specific internal location of a slave device. It first sends the location data to the slave device using the steps in the previous scenario. It then starts the process of reading data with a repeated start condition.

The following sequence of operations takes place when a master device tries to send data to a particular slave device through the I^2C bus:

1) The master device sends the start condition.

2) The master device sends the seven address bits, which corresponds to the slave device to be targeted.

3) The master device sets the Read/Write bit to '0', which signifies a write.

4) Now two scenarios are possible.

5) If no slave device matches with the address sent by the master device, the next ACK/NACK bit stays at '1' (default). This signals the master device that the slave device identification is unsuccessful. The master clock will end the current transaction by sending a Stop condition or a new Start condition.

6) If a slave device exists with the same address as the one specified by the master device, the slave device sets the ACK/NACK bit to '0', which signals the master device that a slave device is successfully targeted.

7) If a slave device is successfully targeted, the master device now sends 8 bits of data only considered and received

by the targeted slave device. This data means nothing to the remaining slave devices.

8) If the slave device successfully receives the data, it sets the ACK/NACK bit to '0', which signals the master device to continue.

9) The previous two steps are repeated until all the data is transferred.

*10)*After all the data is sent to the slave device, the master device sends the Stop condition, which signals all the slave devices that the current transaction has ended.

C. Data Frames

I2C data is transferred in messages. Messages are broken into frames of data. Each message has an address frame that contains the binary address of the slave and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame. The format of the message used within an I2C system is shown in Fig. 7.

Start Condition is initiated by making the SDA line switched from a high voltage level to a low voltage level before the SCL line switches from high to low. The Stop condition is achieved through switching the SDA line from LOW voltage to HIGH voltage after SCL is switched from LOW to HIGH. The Bits 7-10 contain the address of the slave with which the Master wants to communicate. The Read/Write Bit specifies whether the Master is sending data to the slave (low voltage level) or requesting it (high voltage level). The ACK/NACK Bit specifies whether the Master requires Acknowledgment from the slave or otherwise. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

D. Comparison of Application Specific CAN and I2C Networks

I2C and CAN networks differ in many ways: speeds, protocols used for transmission and reception of the data, addressing the devices within the networks, the data frames used for transmission and reception of the data, error control implemented, etc. Making a device as a slave to both the networks is cumbersome. There can be many ways of interconnecting both the networks, including connectivity through a single Master, Connectivity through Multiple Masters, Connectivity through a Bridge, and by implementing a Universal Bus.

E. Interconnecting between CAN and I2C Networks through Multi-master Interface

The interconnection between the I2C network and CAN network is achieved by interfacing the MASTERS using an Ethernet Interface is shown in Fig. 8.

The Master on the I2C network has both the I2C and Ethernet Interface, and similarly, the Master on the CAN network has both the CAN and Ethernet Interface. Whenever data from an I2C slave is transmitted to a CAN salve, the data is first transmitted to the I2C Master using the I2C protocol. The data packets are de-assembled and then assembled into Ethernet packets. The Ethernet packets are then transmitted to the CAN master through peer to peer connection established through Ethernet. The CAN-master receives the Ethernet packets, and the packets are dissembled and assembled into CAN packets, which are then transmitted to the CAN slave. The process of transmission from a CAN slave to the I2C slave similarly takes place.

The entire communication process involves selecting the proper speeds considering I2C, CAN, and Ethernet communication systems. The communication is completed with an acceptable response time. The delay caused due to de-assembling and assembling the packets must also be taken into account while calculating the response time.

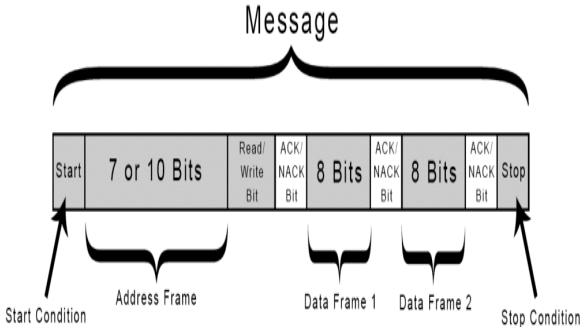

Fig. 7. I^{2}C Message Format.

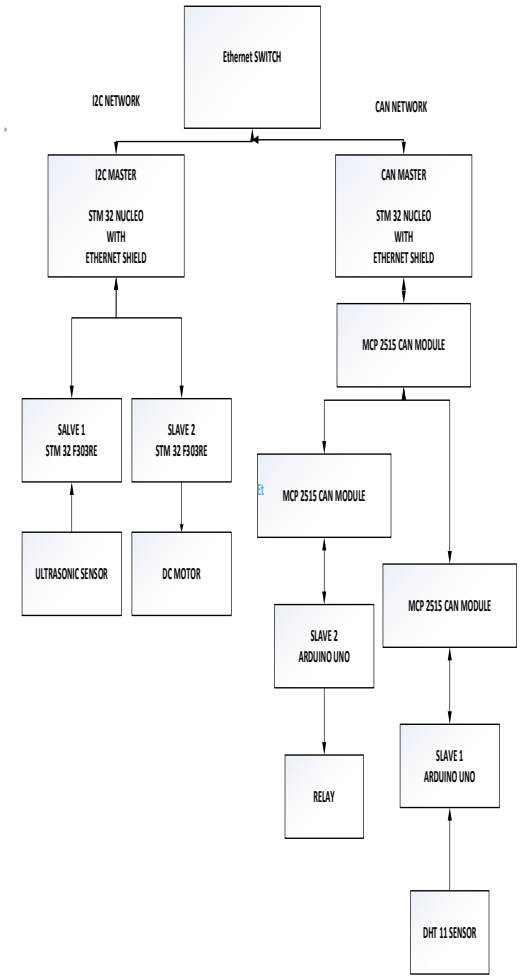

Fig. 8. Interconnecting I^{2}C and CAN Networks through Ethernet Interface.

IV. Architecture for Hybridised Communication in between I2C and CAN Networks

The architecture for establishing communication among I2C and CAN network through Ethernet-based Multi-master Interface is shown in Fig. 9. The most important issue is the arbitration among the I2C Master and CAN master on the kind of speeds used for effecting communications intra I2C, intra CAN, and between the masters using internet Interface. Speed matching is necessary so that the required response time is met. There should be time allowance for assembling and de-assembling the packets of different types. The software components contained within the slaves and the Masters shall carry designated functions as shown in the Architectural Diagram. The masters agree on speed based on the amount of data to be submitted, considering the type of packets and the packets' size to be transmitted, and considering the amount of delay time caused due to de-assembling and assembling processes. Once the speed agreements are achieved, the Master shall communicate the agreed speeds to the respective slaves to follow the speeds, especially setting the slave's internal timers.

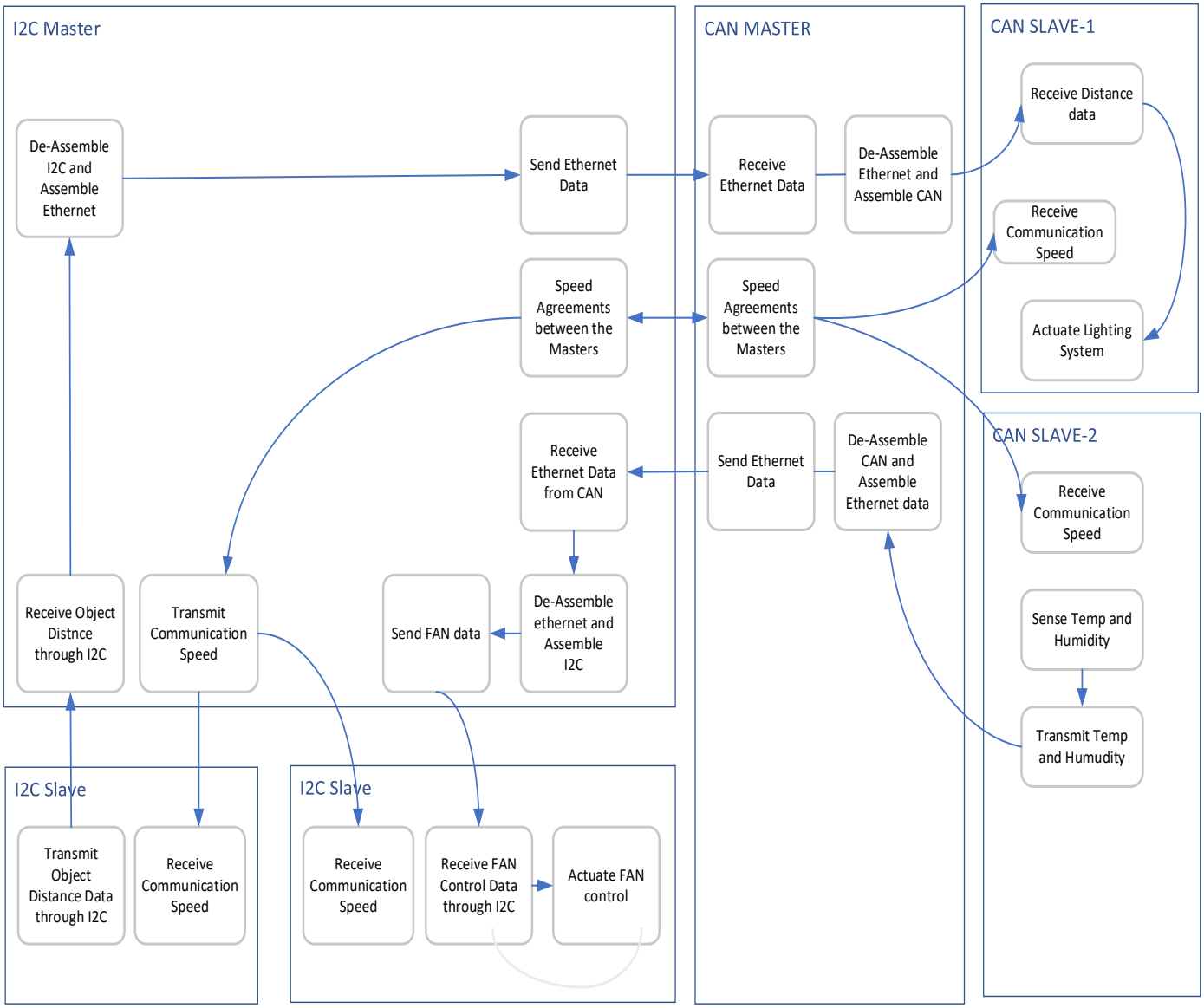

Fig. 9. Architecture for Implementing MULTI-MASTER Ethernet-based Interface for Interconnecting I2C and CAN Networks.

V. Decision Making on Fixation of the Speeds of I2C, CAN and Ethernet for achieving Effective and Fast Communication

While data moves from one type of protocol to another, the data packet size increases or decreases. When an I2C Data packet is to be moved using the Ethernet protocol, the data packet size increases while the transmission speed increases; when a data packet is received through Ethernet into a CAN-based system, the same is de- pocketed and assembles into a packet size of small packets, which can be handled through fewer transmission speeds. It is rather challenging to decide on the transmission speeds chosen when communication has to be effected using a specific I2C, Ethernet, and CAN speeds that reduce the transmission time. The choice of speeds is also dependent on the total raw data transmitted from Time to Time.

In the typical example stated in section 5.0, the temperature data needs to be moved from the I2C network into the CAN network through the Ethernet interface. Similarly, the distance data must be moved in the other way. It is sufficient to analyze from either end of I2C and CAN for computing the time taken to transmit from either end. For this reason, the data analysis from I2C end to CAN is presented in this paper.

Communication time computations have been carried considering the data size that includes 16 Bits, 32Bits, 40Bits and 48Bits, I2C speeds that Include 100kbps, 400kbps, 3482kbps, and 5120kbps. CAN speeds include 500kbps, 250kbps, 125kbps, 10Kbps, and the Ethernet speeds that include 10240kbps, 20480kbps, 30720kbps, and 5120 Kbps to find the combination of speeds that provide the least response time. Table III, Table IV, Table V, and Table VI show the computations regarding data sizes 16bits, 32Bits, 40Bits, and 48Bits, respectively. The response time computations are made considering time taken to transmit using I2C protocol, data receiving time using Ethernet protocol, time taken to de-pocketing I2C packets and Pocketing to Ethernet packets, time is taken to transmit using Ethernet protocol, time is taken for receiving the data on the master side using the Ethernet protocol, time taken to DE packet the Ethernet packet to CAN packet, and time taken to transmit the CAN Packets to the CAN slave.

The response time computations are shown in Table VII, Table VIII, Table IX, and Table X for data sizes 16bits, 32Bits, 40Bits, and 48Bits for a different combination of speeds considering I2C, Ethernet, and CAN. It can be seen that the least response time (4.856002808 Micro Secs) is obtained when data size is 16 Bit when one considers I^2C speed of 5120 bits, Ethernet speed of 51200bits, and CAN speed of 500bits. The least response time obtained is 5.241206 Micro Seconds when data size is 32Bits and when one considers I2C speed of 3482bits, Ethernet speed of 51200bits, and CAN speed of 500bits. The least response time obtained is 4.517013550 microseconds when the data Size 40 Bits considering I2C speed 5120bits, Ethernet Speed of 20480bits, and CAN speed of 500bits. The least response time obtained is 5.542037964 Microseconds when the data size is fixed are 48bits considering the I2C speed being 5120bits, Ethernet speed being 51200bits, and CAN speed 500bits.

TABLE III. Communication Time Computations for Data Size 16 Bits with Different I2C, Ethernet, and CAN Speeds

I2C Packet Size in Bits	Number of I^2C Packets	Total Data to be Transmitted in Bits through I^2C	Speed in KBPS	Transmission Time Secs	Ethernet reconceiving Time in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted	Ethernet Speeds in KBPS	Time Taken to Transmit through Ethernet in Secs	Time Taken to Receive through Ethernet in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted in Bits	CAN Speed in KBPS	Time Taken to Transmit Data	Time Taken to Receive Data - CAN Slave Side	Time Taken to de pocketing the Packets on CAN Slave Size	Total Time Taken for Data Transmission
31	1	31	100	0.0003	0.00030	0.0001	512	10240	4.88E-05	4.88E-05	0.0001	87	500	0.00017	0.00017	0.0001	0.00080
			400	0.0001	0.00008	0.0001		20480	2.44E-05	2.44E-05	0.0001		250	0.00034	0.00034	0.0001	0.00073
			3482	0.0000	0.00001	0.0001		30720	1.63E-05	1.63E-05	0.0001		125	0.00068	0.00068	0.0001	0.00100
			5120	0.0000	0.00001	0.0001		51200	9.77E-06	9.77E-06	0.0001		10	0.0085	0.008496	0.0001	0.00870

TABLE IV. COMMUNICATION TIME COMPUTATIONS FOR DATA SIZE 32 BITS WITH DIFFERENT I2C, ETHERNET, AND CAN SPEEDS

I2C Packet Size in Bits	Number of I2C Packets	Total Data to be Transmitted in Bits through I2C	Speed in KBPS	Transmission Time Secs	Ethernet receiving Time in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted	Ethernet Speeds in KBPS	Time Taken to Transmit through Ethernet in Secs	Time Taken to Receive through Ethernet in Secs	Ethernet De-Pocketing and Pocketing Time	CAN Packet Suze in Bits	CAN Speed in KBMS	Time Taken to Transmit Data	Time Taken to Receive Data - CAN Slave Side	Time Taken to De pocketing the Packets on CAN Slave Size	Total Time Taken for Data Transmission
47	1	47	100	0.0005	0.00046	0.0001	512	10240	4.88281E-05	4.88281E-05	0.0001	103	500	0.000201172	0.000201172	0.0001	0.00098
			400	0.0001	0.00011	0.0001		20480	2.44141E-05	2.44141E-05	0.0001		250	0.000402344	0.000402344	0.0001	0.00083
			3482	0.0000	0.00001	0.0001		30720	1.6276E-05	1.6276E-05	0.0001		125	0.000804688	0.000804688	0.0001	0.00113
			5120	0.0000	0.00001	0.0001		51200	9.76563E-06	9.76563E-06	0.0001		10	0.010058594	0.010058594	0.0001	0.01027

TABLE V. COMMUNICATION TIME COMPUTATIONS FOR DATA SIZE 40 BITS WITH DIFFERENT I2C, ETHERNET, AND CAN SPEEDS

Total Data to be Transmitted in Bits through I2C	Speed in KBPS	Transmission Time Secs	Ethernet receiving Time in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted	Ethernet Speeds in KBPS	Time Taken to Transmit through Ethernet in Secs	Time Taken to Receive through Ethernet in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted in Bits	CAN Speed in KBMS	Time Taken to Transmit Data	Time Taken to Receive Data - CAN Slave Side	Time Taken to De pocketing the Packets on CAN Slave Size	Total Time Taken for Data Transmission
55	100	0.0005	0.00054	0.0001	512	10240	4.88281E-05	4.88281E-05	0.0001	111	500	0.000216797	0.000216797	0.0001	0.00108
	400	0.0001	0.00013	0.0001		20480	2.44141E-05	2.44141E-05	0.0001		250	0.000433594	0.000433594	0.0001	0.00088
	3482	0.0000	0.00002	0.0001		30720	1.6276E-05	1.6276E-05	0.0001		125	0.000867188	0.000867188	0.0001	0.00119
	5120	0.0000	0.00001	0.0001		51200	9.76563E-06	9.76563E-06	0.0001		10	0.010839844	0.010839844	0.0001	0.01105

TABLE VI. COMMUNICATION TIME COMPUTATIONS FOR DATA SIZE 48BITS WITH DIFFERENT I2C, ETHERNET, AND CAN SPEEDS

Total Data to be Transmitted in Bits through	Speed in KBPS	Transmission Time Secs	Ethernet receiving Time in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted	Ethernet Speeds in KBPS	Time Taken to Transmit through Ethernet in Secs	Time Taken to Receive through Ethernet in Secs	Ethernet De-Pocketing and Pocketing Time	Total Data to be Transmitted in Bits	CAN Speed in KBMS	Time Taken to Transmit Data	Time Taken to Receive Data - CAN Slave Side	Time Taken to De pocketing the Packets on CAN Slave Size	Total Time Taken for Data Transmission
63	100	0.000615234	0.000615234	0.0001	512	10240	4.88281E-05	4.88281E-05	0.0001	119	500	0.000232422	0.000232422	0.0001	0.00117
	400	0.000153809	0.000153809	0.0001		20480	2.44141E-05	2.44141E-05	0.0001		250	0.000464844	0.000464844	0.0001	0.00093
	3482	1.7671E-05	1.7671E-05	0.0001		30720	1.6276E-05	1.6276E-05	0.0001		125	0.000929688	0.000929688	0.0001	0.00126
	5120	1.20163E-05	1.20163E-05	0.0001		51200	9.76563E-06	9.76563E-06	0.0001		10	0.011621094	0.011621094	0.0001	0.01183

TABLE VII. RESPONSE TIME COMPUTATIONS WHEN DATA SIZE = 16 BITS

Response Time Computations when data Size = 16Bits		
I2C, Ethernet, CAN Speeds	**Response Time in Seconds**	**Normalized Response time**
100 ,1024 ,500	0.000821484375000	8.214843750
100 ,20480 ,500	0.000797070312500	7.970703125
100 ,30720,500	0.000788932291667	7.889322917
100, 51200,500	0.000782421875000	7.824218750
400,1024 ,500	0.000594433593750	5.944335938
400 ,20480 ,500	0.000570019531250	5.700195313
400 ,30720,500	0.000561881510417	5.618815104
400, 51200,500	0.000555371093750	5.553710938
3482,1024 ,500	0.000527445265826	5.274452658
3482 ,20480 ,500	0.000527445265826	5.274452658
3482 ,30720,500	0.000494893182493	4.948931825
3482, 51200,500	0.000488382765826	4.883827658
5120,1024 ,500	0.000485600280762	4.856002808
5120 ,20480 ,500	0.000500248718262	5.002487183
5120 ,30720,500	0.000492110697428	4.921106974
5120, 51200,500	0.000485600280762	4.856002808
100 ,1024 ,250	0.000991406250000	9.914062500
100 ,20480 ,250	0.000966992187500	9.669921875
100 ,30720,250	0.000958854166667	9.588541667
100, 51200,250	0.000952343750000	9.523437500
400,1024 ,250	0.000764355468750	7.643554688
400 ,20480 ,250	0.000739941406250	7.399414063
400 ,30720,250	0.000731803385417	7.318033854
400, 51200,250	0.000725292968750	7.252929688
3482,1024 ,250	0.000697367140826	0.697367141
3482 ,20480 ,250	0.000672953078326	0.672953078
3482 ,30720,250	0.000664815057493	0.664815057
3482, 51200,250	0.000658304640826	0.658304641
5120,1024 ,250	0.000694584655762	6.945846558
5120 ,20480 ,250	0.000670170593262	6.701705933
5120 ,30720,250	0.000662032572428	6.620325724
5120, 51200,250	0.000655522155762	6.555221558
100 ,1024 ,125	0.001331250000000	13.312500000
100 ,20480 ,125	0.001306835937500	13.068359375
100 ,30720,125	0.001298697916667	12.986979167
100, 51200,125	0.001292187500000	12.921875000
400,1024 ,125	0.001104199218750	11.041992188
400 ,20480 ,125	0.001079785156250	10.797851563
400 ,30720,125	0.001071647135417	10.716471354
400, 51200,125	0.001065136718750	10.651367188
3482,1024 ,125	0.001037210890826	10.372108908
3482 ,20480 ,125	0.001012796828326	10.127968283
3482 ,30720,125	0.001004658807493	10.046588075
3482, 51200,125	0.000998148390826	9.981483908
5120,1024 ,125	0.001034428405762	10.344284058
5120 ,20480 ,125	0.001010014343262	10.100143433
5120 ,30720,125	0.001001876322428	10.018763224
5120, 51200,125	0.000995365905762	9.953659058
100 ,1024 ,10	0.009147656250000	91.476562500
100 ,20480 ,10	0.009123242187500	91.232421875
100 ,30720,10	0.009115104166667	91.151041667
100, 51200,10	0.009108593750000	91.085937500
400,1024 ,10	0.008920605468750	89.206054688
400 ,20480 ,10	0.008896191406250	88.961914063
400 ,30720,10	0.008888053385417	88.880533854
400, 51200,10	0.008881542968750	88.815429688
3482,1024 ,10	0.008853617140826	88.536171408
3482 ,20480 ,10	0.008829203078326	88.292030783
3482 ,30720,10	0.008821065057493	88.210650575
3482, 51200,10	0.008814554640826	88.145546408
5120,1024 ,10	0.008850834655762	88.508346558
5120 ,20480 ,10	0.008826420593262	88.264205933
5120 ,30720,10	0.008818282572428	88.182825724
5120, 51200,10	0.008811772155762	88.117721558

TABLE VIII. RESPONSE TIME COMPUTATIONS WHEN DATA SIZE = 32 BITS

Response time Computations when Data size is 32Bits		
I2C, Ethernet, CAN Speeds	**Response time Seconds**	**Response time Normalized**
100 ,10240 ,500	0.00100898	10.089844
100 ,20480 ,500	0.00098457	9.845703
100 ,30720,500	0.00097643	9.764323
100, 51200,500	0.00096992	9.699219
400,10240 ,500	0.00066475	6.647461
400 ,20480 ,500	0.00084150	8.415039
400 ,30720,500	0.00063219	6.321940
400, 51200,500	0.00052412	5.241206
3482 ,10240 ,500	0.00056318	5.631831
3482 ,20480 ,500	0.00053877	5.387691
3482 ,30720,500	0.00053063	5.306311
3482, 51200,500	0.00052412	5.241206
5120,10240 ,500	0.00055896	5.589645
5120 ,20480 ,500	0.00053455	5.345505
5120 ,30720,500	0.00052641	5.264125

Response time Computations when Data size is 32Bits		
I2C, Ethernet, CAN Speeds	**Response time Seconds**	**Response time Normalized**
5120, 51200,500	0.00051990	5.199020
100 ,10240 ,250	0.00121016	12.101563
100 ,20480 ,250	0.00118574	11.857422
100 ,30720,250	0.00117760	11.776042
100, 51200,250	0.00117109	11.710938
400,10240 ,250	0.00086592	8.659180
400 ,20480 ,250	0.00084150	8.415039
400 ,30720,250	0.00083337	8.333659
400, 51200,250	0.00082686	8.268555
3482,10240 ,250	0.00076436	7.643550
3482 ,20480 ,250	0.00073994	7.399410
3482 ,30720,250	0.00073180	7.318029
3482, 51200,250	0.00072529	7.252925
5120,10240 ,250	0.00076014	7.601364
5120 ,20480 ,250	0.00073572	7.357224
5120 ,30720,250	0.00072758	7.275843
5120, 51200,250	0.00072107	7.210739
100 ,10240 ,125	0.00156133	15.613281
100 ,20480 ,125	0.00158809	15.880859
100 ,30720,125	0.00157995	15.799479
100, 51200,125	0.00157344	15.734375
400,10240 ,125	0.00126826	12.682617
400 ,20480 ,125	0.00124385	12.438477
400 ,30720,125	0.00123571	12.357096
400, 51200,125	0.00122920	12.291992
3482,10240 ,125	0.00116670	11.666988
3482 ,20480 ,125	0.00114228	11.422847
3482 ,30720,125	0.00113415	11.341467
3482, 51200,125	0.00112764	11.276363
5120,10240 ,125	0.00116248	11.624802
5120 ,20480 ,125	0.00113807	11.380661
5120 ,30720,125	0.00112993	11.299281
5120, 51200,125	0.01037732	103.773239
100 ,10240 ,10	0.01086641	108.664063
100 ,20480 ,10	0.01084199	108.419922
100 ,30720,10	0.01083385	108.338542
100, 51200,10	0.01082734	108.273438
400,10240 ,10	0.01052217	105.221680
400 ,20480 ,10	0.01049775	104.977539
400 ,30720,10	0.01048962	104.896159
400, 51200,10	0.01048311	104.831055
3482,10240,10	0.01042061	104.206050

Response time Computations when Data size is 32Bits		
I2C, Ethernet, CAN Speeds	**Response time Seconds**	**Response time Normalized**
3482 ,20480 ,10	0.01039619	103.961910
3482 ,30720,10	0.01038805	103.880529
3482, 51200,10	0.01038154	103.815425
5120,10240 ,10	0.01041639	104.163864
5120 ,20480 ,10	0.01039197	103.919724
5120 ,30720,10	0.01038383	103.838343
5120, 51200,10	0.01028709	102.870895

TABLE IX. RESPONSE TIME COMPUTATIONS WHEN DATA SIZE = 40 BITS

Response Time Computations when Data Size = 40Bits		
I2C, Ethernet, and CAN Speeds	**Response Time in Seconds**	**Normalized Response time**
100 ,10240 ,500	0.0011027344	11.027343750
100 ,20480 ,500	0.0010783203	10.783203125
100 ,30720,500	0.0010701823	10.701822917
100, 51200,500	0.0010636719	10.636718750
400,10240 ,500	0.0006999023	6.999023438
400 ,20480 ,500	0.0006754883	6.754882813
400 ,30720,500	0.0006673503	6.673502604
400, 51200,500	0.0006608398	6.608398438
3482,10240 ,500	0.0005810521	5.810520845
3482 ,20480 ,500	0.0005566380	5.566380220
3482 ,30720,500	0.0005485000	5.485000012
3482, 51200,500	0.0005419896	5.419895845
5120,10240 ,500	0.0005761154	5.761154175
5120 ,20480 ,500	0.0004517014	4.517013550
5120 ,30720,500	0.0005435633	5.435633341
5120, 51200,500	0.0005370529	5.370529175
100 ,10240 ,250	0.0013195313	13.195312500
100 ,20480 ,250	0.0012951172	12.951171875
100 ,30720,250	0.0012869792	12.869791667
100, 51200,250	0.0011804688	11.804687500
400,10240 ,250	0.0009166992	9.166992188
400 ,20480 ,250	0.0008922852	8.922851563
400 ,30720,250	0.0008841471	8.841471354
400, 51200,250	0.0008776367	8.776367188
3,48,21,02,40,250	0.0007978490	7.978489595
3482 ,20480 ,250	0.0007734349	7.734348970
3482 ,30720,250	0.0007652969	7.652968762
3482, 51200,250	0.0007587865	7.587864595
5120,10240 ,250	0.0007929123	7.929122925
5120 ,20480 ,250	0.0007684982	7.684982300
5120 ,30720,250	0.0007603602	7.603602091

Response Time Computations when Data Size = 40Bits		
I2C, Ethernet, and CAN Speeds	**Response Time in Seconds**	**Normalized Response time**
5120, 51200,250	0.0007538498	7.538497925
100 ,10240 ,125	0.0017531250	17.531250000
100 ,20480 ,125	0.0017287109	17.287109375
100 ,30720,125	0.0017205729	17.205729167
100, 51200,125	0.0017140625	17.140625000
40,01,02,40,125	0.0013502930	13.502929688
400 ,20480 ,125	0.0013258789	13.258789063
400 ,30720,125	0.0013177409	13.177408854
400, 51200,125	0.0013112305	13.112304688
3482,10240 ,125	0.0012314427	12.314427095
3482 ,20480 ,125	0.0012070286	12.070286470
3482 ,30720,125	0.0011988906	11.988906262
3482, 51200,125	0.0011923802	11.923802095
5120,10240 ,125	0.0012265060	12.265060425
5120 ,20480 ,125	0.0012020920	12.020919800
5120 ,30720,125	0.0011939540	11.939539591
5120, 51200,125	0.0011874435	11.874435425
100 ,10240 ,10	0.0117257813	117.257812500
100 ,20480 ,10	0.0117013672	117.013671875
100 ,30720,10	0.0116932292	116.932291667
100, 51200,10	0.0116867188	116.867187500
400,10240,10	0.0113229492	113.229492188
400 ,20480 ,10	0.0112985352	112.985351563
400 ,30720,10	0.0112903971	112.903971354
400, 51200,10	0.0112838867	112.838867188
3482,10240 ,10	0.0112040990	112.040989595
3482 ,20480 ,10	0.0111796849	111.796848970
3482 ,30720,10	0.0111715469	111.715468762
3482, 51200,10	0.0111650365	111.650364595
5120,10240 ,10	0.0111991623	111.991622925
5120 ,20480 ,10	0.0111747482	111.747482300
5120 ,30720,10	0.0111666102	111.666102091
5120, 51200,10	0.0111503342	111.503341675

TABLE X. RESPONSE TIME COMPUTATIONS WHEN DATA SIZE = 48 BIT

Response time Computation when the Data Size = 48Bits		
I2C, Ethernet, CAN Speed	**Response time in Seconds**	**Normalized Response time**
100 ,10240 ,500	0.0011964844	11.964843750
100 ,20480 ,500	0.0011720703	11.720703125
100 ,30720,500	0.0011639323	11.639322917
100, 51200,500	0.0011574219	11.574218750
400,10240 ,500	0.0007350586	7.350585938

Response time Computation when the Data Size = 48Bits		
I2C, Ethernet, CAN Speed	**Response time in Seconds**	**Normalized Response time**
400 ,20480 ,500	0.0007106445	7.106445313
400 ,30720,500	0.0007025065	7.025065104
400, 51200,500	0.0006959961	6.959960938
3482,10240 ,500	0.0005989210	5.989210241
3482 ,20480 ,500	0.0005745070	5.745069616
3482 ,30720,500	0.0005663689	5.663689408
3482, 51200,500	0.0005598585	5.598585241
5120,10240 ,500	0.0005932663	5.932662964
5120 ,20480 ,500	0.0005688522	5.688522339
5120 ,30720,500	0.0005607142	5.607142131
5120, 51200,500	0.0005542038	5.542037964
100 ,10240 ,250	0.0014289063	14.289062500
100 ,20480 ,250	0.0014044922	14.044921875
100 ,30720,250	0.0013963542	13.963541667
100, 51200,250	0.0013898438	13.898437500
400,10240 ,250	0.0009674805	9.674804688
400 ,20480 ,250	0.0009430664	9.430664063
400 ,30720,250	0.0009349284	9.349283854
400, 51200,250	0.0009284180	9.284179688
3482, 10240, 250	0.0008313429	8.313428991
3482 ,20480 ,250	0.0008069288	8.069288366
82 ,30720,250	0.0007987908	7.987908158
3482, 51200,250	0.0007922804	7.922803991
5120,10240 ,250	0.0008256882	8.256881714
5120 ,20480 ,250	0.0008012741	8.012741089
5120 ,30720,250	0.0007931361	7.931360881
5120, 51200,250	0.0007866257	7.866256714
100 ,10240 ,125	0.0018937500	18.937500000
100 ,20480 ,125	0.0018693359	18.693359375
100 ,30720,125	0.0018611979	18.611979167
100, 51200,125	0.0018546875	18.546875000
400, 10240, 125	0.0014323242	14.323242188
400 ,20480 ,125	0.0014079102	14.079101563
400 ,30720,125	0.0013997721	13.997721354
400, 51200,125	0.0013932617	13.932617188
3482,10240 ,125	0.0012961866	12.961866491
3482 ,20480 ,125	0.0012717726	12.717725866
3482 ,30720,125	0.0012636346	12.636345658
3482, 51200,125	0.0012571241	12.571241491
5120,10240 ,125	0.0012905319	12.905319214
5120 ,20480 ,125	0.0012661179	12.661178589
5120 ,30720,125	0.0012579798	12.579798381

Response time Computation when the Data Size = 48Bits		
I2C, Ethernet, CAN Speed	**Response time in Seconds**	**Normalized Response time**
5120, 51200,125	0.0012514694	12.514694214
100 ,10240 ,10	0.0125851563	125.851562500
100 ,20480 ,10	0.0125607422	125.607421875
100 ,30720,10	0.0125526042	125.526041667
100, 51200,10	0.0125460938	125.460937500
400,10240,10	0.0121237305	121.237304688
400 ,20480 ,10	0.0120993164	120.993164063
400 ,30720,10	0.0120911784	120.911783854
400, 51200,10	0.0120846680	120.846679688
3482,10240 ,10	0.0119875929	119.875928991
3482 ,20480 ,10	0.0119631788	119.631788366
3482 ,30720,10	0.0119550408	119.550408158
3482, 51200,10	0.0119485304	119.485303991
5120,10240 ,10	0.0119819382	119.819381714
5120 ,20480 ,10	0.0119575241	119.575241089
5120 ,30720,10	0.0119493861	119.493860881
5120, 51200,10	0.0119428757	119.428756714

TABLE XI. COMPARATIVE ANALYSIS OF I2C, ETHERNET AND CAN SPEEDS @ DIFFERENT DATA SIZES

Data Size	**I2C, Ethernet, CAN speeds**	**Response time in Seconds**	**Response Time in Micro Seconds**
16	5120, 51200,500	0.00048560	4.85600
32	3482, 51200,500	0.00052412	5.24121
40	5120 ,20480 ,500	0.00045170	4.51701
48	5120, 51200,500	0.00055420	5.54204

The overall assessments of comparable speeds considering local minimums of different data sizes and their related speeds are shown in Table XI. It can be seen from Table XI and Fig. 10 that optimum response time can be achieved when I2C speed is 5120 Bits, Ethernet speed is fixed at 20480 Bits, and the CAN speed fixed at 40bits and that too when the data size is fixed at 40 Bytes. However, if there is a limitation on the data size, then the local minimum can be determined considering all the combinations of the I2C, Ethernet, and CAN speeds.

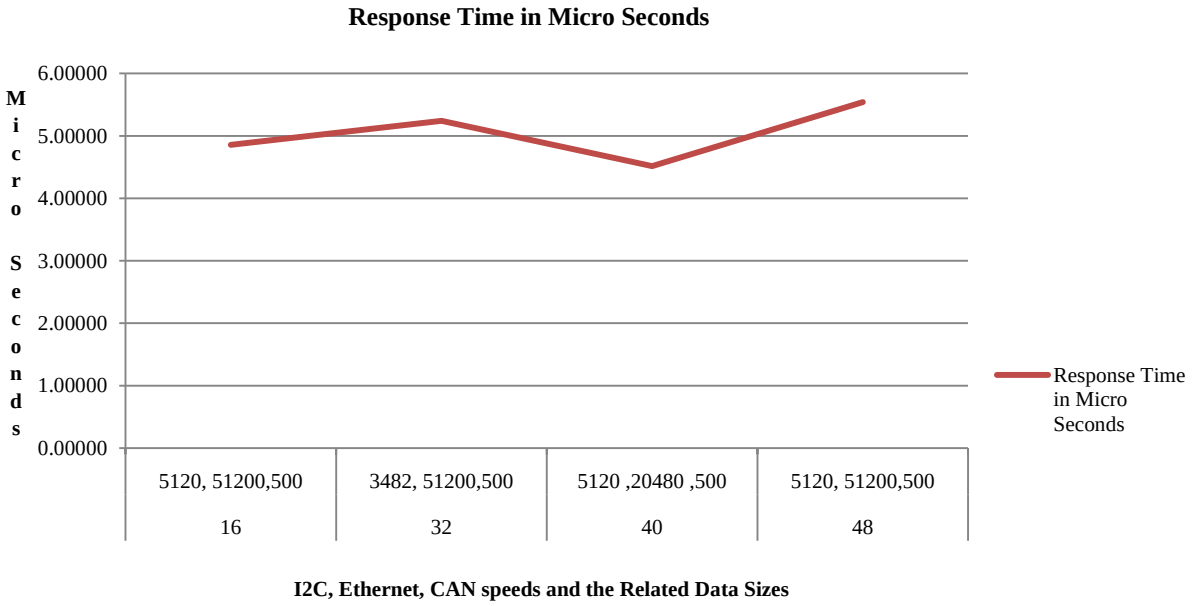

Fig. 10. Optimum Response Time among Many Optimum Locals.

VI. Conclusions

Interfacing different heterogeneous ES networks into a composite network are quite challenging. Many issues that include synchronizing, arbitration, timing, and speed control have to be tackled properly so that optimum acceptable response time is achieved, failing which the system's expected response time cannot be supported. Many mechanisms can be implemented for achieving the issues of hybridization that include introducing a bridge device, adapting a universal bus, a single master interfacing, and a Multi-Master interfacing. A detailed comparison of protocols shows the issues that must be considered for interacting and integrating heterogeneous communication systems.

In this paper, a Multi-Master interfacing interconnects an I2C network with a CAN network using an Ethernet interface. The detailed working of data communication considering the different combination of I2C, CAN, and Ethernet speeds have been presented, and the best combination of these speeds that gives the least response time has been presented.

Further research can be carried to find the speed combinations that must be considered when networking is to be done considering I2C+USB, I2C+RS485, CAN+USB, CAN+RS485, USB_RS485. Interfacing using a single master through a common bus is also one of the important approaches that can be considered.

References

[1] CHAI Yan-Jie, SUN Ji-yin, GAO Jing, TAO Ling-jiao, JI Jing, BAO Fei-hu," Improvement of I2C Bus and RS-232 Serial Port un-der Complex Electromagnetic Environment" International Conference on Computer Science and Software Engineering, 2008, PP 178 - 181.

[2] Antonio Casimiro, Jose Rufino, Luis Marques, Mario Calha, and Paulo Verissimo "Applying Architectural Hybridization in Net-worked Embedded Systems," IFIP International Federation for In-development Processing, 2009, PP 264-275.

[3] Lou Guohuan, ZhangHao, Zhao Wei, "Research on Designing Method ofCAN Bus and Modbus Protocol Conversion Interface" International Conference on Future BioMedical Information Engineering,2009, PP 180-182.

[4] Xiaoming Li, Mingxiong Li, "An Embedded CAN-BUS Communication Module for Measurement and Control System" International conference on ICEEE, 10.119/ICEEE.2010.5661248, 2010.

[5] Peng Daogang, Zhang Hao, Li Hui, Xia Fei," Development of the Communication Protocol Conversion Equipment Based on Embedded Multi-MCU and μC/OS-II," International Conference on Measuring Technology and Mechatronics Automation, 2010, PP 15-18.

[6] Tae-Won Kim, Jin Ho Kim, Do Eon Lee, Jun young Moon, Jae Wook Jeon," Development of Gateway based on BroadR-Reach for Application in Automation Network," International Conference on Computing, Communication and Automation, 2016, PP 421-426.

[7] CHEN Huijuan," Heterogeneous Network Integration Based on Protocol Conversion" Control Conference (CCC), 2016, PP 6888-6893.

[8] Chen Wei, Wang Xijun, Sun Wenxia," The Design of profinet- Modbus protocol conversion Gateway Based on the ERTEC 200P", International Conference on Software, Knowledge, Information Management & Applications, 2016, PP 87-91.

[9] Jason Oberg, Wei Hu, Ali Irturk," Information Flow Isolation in I2C and USB", Design Automation Conference, 2011, PP 254-259.

[10] Li Hui, Zhang Hao, Peng Daogang," Design and Application of Communication Gateway of EPA and MODBUS on Electric Power System" International Conference on Future Electrical Power and Energy Systems, 2012, PP 286 – 292.

[11] Gheorghe G.Florea, Oana, Rohat, Monica G. Dragan, "Contributions to the Development of Communication Protocols Conversion Equipment," 2nd IFAC Workshop on Convergence of Information Technologies and Control Methods with Power Systems, 2013, VOLUME 46, Issue 6, PP 84-88.

[12] Umesh Goyal, Gaurav Khurana," Implementing MOD bus and CAN bus Protocol Conversion Interface" IJETT, 2013, VOLUME 4, Issue 4, PP 630-635.

[13] Roopak Sinha," Conversing at Many Layers: Multi-layer System-On-chip Protocol Conversion," 20th International Conference on Engineering of Complex Computer Systems, 2015, PP 170-173.

[14] JKRSastry, J. Viswanadh Ganesh, and J. SasiBhanu, "I2C based Networking for Implementing Heterogeneous Microcontroller based Distributed Embedded Systems", Indian Journal of Science and Technology, 2015, Vol 8(15), PP 1-10.

[15] JKRSastry, Valluru Sai Kumar Reddy, Smt J SasiBhanu," Net-working Heterogeneous Microcontroller based Systems through Universal Serial Bus," IJECE, 2015, Vol. 5, No. 5, PP. 992-1002.

[16] JKR. Sastry, M. Vijaya Lakshmi, and Smt J. SasiBhanu," Optimizing Communication Between Heterogeneous Distributed Embedded Systems Using CAN Protocol," ARPN Journal of Engineering and Applied Sciences, 2015, VOL. 10, NO. 18, PP 7900-7911.

[17] JKR Sastry, T. Naga Sai Tejasvi and J. Aparna, Dynamic scheduling of message flow within a distributed embedded system connected through RS485 network, ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 9, MAY 2017.

[18] J. K. R. Sastry, A. Suresh, and Smt J. Sasi Bhanu, Building Heterogeneous Distributed Embedded Systems through RS485 Communication Protocol, ARPN Journal of Engineering and Applied Sciences, issue. 16, vol.10, 2015.

[19] K. Chaitanya, Sastry JKR, K. N. Sravani, D. Pavani, Ramya, and K. Rajasekhara Rao, Testing Distributed Embedded Systems Using Assert Macros, ARPN Journal of Engineering and Applied Sciences, 2017, page no.3011-3021.

[20] Sastry JKR, K. Chaitanya, K. Rajasekhara Rao, DBK Kamesh, Testing Distributed Embedded Systems Through Instruction Set Simulators, PONTE, International Journal of Sciences and Research, issue.7, vol.73, July 2017, page no.353-382.

[21] JKR Sastry, K. Chaitanya, K. Rajasekhara Rao, DBK Kamesh, An Efficient Method for Testing Distributed Embedded Systems using In-circuit Emulators, PONTE, International Journal of Sciences and Research, issue.7, vol.73, 2017, page no.390-422.

[22] K. Chaitanya, JKR Sastry, K. Rajasekhara Rao, Testing Distributed Embedded Systems Using Logic, Analyzer, International Journal of Engineering and Technology, March 2018, page no. 297-302.

[23] K Chaitanya1, Dr. K Rajasekhra Rao, Dr. JKR Sastry, A Framework for Testing Distributed Embedded Systems, International Journal advanced Trends in computer science and engineering, 2019, Volume 8, No.4, PP. 1104-1227.

[24] K Chaitanya, Dr. K Rajasekhra Rao, Dr. JKR Sastry. A Formal and Enriched Framework for Testing Distributed Embedded Systems, International Journal Emerging Trends in Engineering Research, Volume 7, No. 12, 2019, PP. 867-878.

[25] J. Rajasekhar, Dr. JKR Sastry, An approach to hybridization of embedded system networks, International Journal of Engineering & Technology, 7 (2.7) (2018) 384-389.

[26] Jammalamadaka Rajasekhar, JKR Sastry, Building Composite Embedded Systems Based Networks Through Hybridization and Bridging I2C and CAN, Journal of Engineering Science and Technology, Vol. 15, No. 2 (2020) 858 – 88.

[27] E. Mounika, Dr. JKR Sastry, J. Rajasekhar, A Hybridised Heterogeneous Embedded System Networking through Multi-Master Interface, International Journal of Emerging Trends in Engineering Research, Volume 8. No. 3 March 2020, https://doi.org/ 10.30534/ijeter/2020/45832020.