VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

# OPTIMIZING COMMUNICATION BETWEEN HETEROGENEOUS DISTRIBUTED EMBEDDED SYSTEMS USING CAN PROTOCOL

J. K. R. Sastry[1], M. Vijaya Lakshmi[1] and Smt J. Sasi Bhanu[2]
[1]Department of Electronics and Computer Science Engineering, KL University, Vaddeswaram, Guntur District, India
[2]Department of Computer Science Engineering, KL University, Vaddeswaram, Guntur District, India
E-Mail: drjkrsastry@gmail.com

## ABSTRACT

Distributed embedded systems are being used for many purposes especially in the field of automobile automation, safety critical systems and the process systems which need high performance and response. Many communications systems are being used which include I$^2$C, CAN, and USB etc., for interconnecting distributed embedded systems. These communication systems will function effectively if all the distributed embedded systems are homogeneous. However these systems need many of the conversions when the networking of heterogeneous embedded systems is needed. Every distributed embedded system requires the design of networking, communication system architecture, design of flow of messages through proper addressing and arbitration, synchronization, error detection and control etc. These issues become complicated due to the use of heterogeneous embedded systems. This paper presents networking of a distributed embedded system through use of CAN protocol which is to be built around heterogeneous microcontroller based systems, a novel system for implementation of arbitration, the architecture for message flow and the design of data flow across the distributed embedded systems. The inventions presented in the paper have been applied to a pilot distributed system that monitors and controls the temperatures within a Nuclear reactor system.

**Keywords:** distributed embedded systems, communication protocols, CAN based communication system, networking heterogeneous embedded systems.

## INTRODUCTION

Embedded systems are being used extensively for monitoring and controlling various physical parameters. Embedded systems are reactive that they respond to changes taking place in the external environment. Almost all every electronic gadgets (which include digital cameras, washing machine etc.) used today is fitted with an embedded system. Embedded systems are also used these days as computing nodes connected on to internet, forming into internet of things.

Many embedded system based solutions are being offered these days, which requires interconnecting of many individual embedded systems. An automobile system as such has in it, many embedded systems which individually deals with controlling brakes, doors, mirrors, rare and front object indicators, engine temperature, wheel speed, tyre pressure, DVD control etc. These individual embedded systems are networked for providing information into a display unit which is fitted into a dash board. The networking of the individual embedded systems must take into consideration various heterogeneous ECUs' that are used for monitoring and controlling any one of the four said parameters. These days even multi layered networking of embedded systems are being used, each layer catering for a specific communication speed. Serial bus based communication is being used for interconnecting various embedded systems using communication protocols such as I$^2$C, CAN, USB, RS485 etc. Each type of networking leads to different communication characteristics such as baud rate, length of communication, bus termination etc.

In literature, many networking solutions have been presented by using standard protocols for which native support has been made available in specific microcontroller based systems. They do not address the issues related to networking heterogeneous microcontroller based systems. Many distributed embedded applications are in use today and each application requires that the communication system be designed that meets the specific requirements of a distributed embedded application. The serial bus based communication systems offers generic communication protocols which need to be customized considering the distributed embedded application. Every distributed embedded application is different in-terms of the processing and communication and these needs to be addressed separately through specific designs.

A distributed embedded system involves use of individual microcontroller based systems. Each microcontroller based system may have built-in interfaces using which communication with other microcontrollers can be achieved. Establishing communication among various microcontroller based systems is essential to implement a distributed embedded application. In a distributed embedded application both the hardware and software that comprise entire application is distributed. Communication is necessary among the microcontroller systems for exchanging of process information.

Networking of the microcontroller based systems becomes one of the most important criteria in implementing distributed embedded systems. The most critical aspects that must be considered to achieve the networking of embedded systems are to address the heterogeneity issue which includes interfaces, protocols, implementation of protocols etc. Networking of embedded systems can be achieved in many ways using protocols such as RS232C, RS485, RS422, SPI, fire wire, USB, CAN, I$^2$C, ETHERNET, PCI, and ESA etc. Among all, bus based serial communication protocols are used for establishing a network connecting all the individual microcontroller systems. CAN is such a protocol which is frequently used by the industry for effecting communication among individual microcontroller based systems.

One of the major problems in implementing CAN based system is due to lack of native support within some of the microcontroller systems. The CAN implementations within some of the microcontroller systems defer a lot, as CAN exists in several versions and the existence of several implementation variations. This is leading to establishing interfacing using many of the conversion devices which leads to frequent protocol conversion. Speed of communication is normally affected when communication is achieved using protocol conversion.

Every distributed embedded system requires different communication system architecture and the every communication system must be customized for implementation of specific distributed embedded systems. No generic communication system as such will meet the purposes of all types of distributed systems. Thus there is a requirement of finding mechanisms and methods using which CAN based communication is used within the network of heterogeneous embedded systems and also design application specific communication system architecture and the designing of the same considering various aspects of communication which include addressing, configuration, transmission, reception, arbitration, synchronization, error detection and control etc.

The CAN based system does not support prioritization of the slaves to respond, even though all the slaves are allotted with an identifier and not the address. The responses form the salves must be prioritized considering the application requirements based on the sequence in which the message must flow. There must be a way of prioritizing the responses as per the application requirements initiated from the slaves and controlled from the master through a mechanism built into a master microcontroller based system. This paper addresses the mechanism that prioritizes the responses from the slaves.

Every distributed embedded System has to be designed with flow of data considering the application requirement. The flow control varies from system to system. In this paper the design of data packets and the flow of the same considering the pilot application that monitors and controls the temperatures within a nuclear reactor system have been presented.

## SPECIFICATION - DISTRIBUTED EMBEDDED SYSTEMS (PILOT PROJECT)

Monitoring the temperatures within nuclear reactor tubes is one of the most important issues when it comes to uranium enrichment. The nuclear reactor tubes are distinctly situated and the sensors that are needed to sense the temperature within the nuclear reactor are also placed a part at long distances. The actuating mechanisms that control the temperature within nuclear reactor systems through injection of coolants are also situated at remote locations.

The sensing and actuating mechanisms are closely related and the functioning of the same are to be coordinated. The Embedded systems that form the distributed embedded system must be able to realise all the functional requirements of pilot project that monitors and controls the temperatures within the Microcontroller based systems. The configuration parameters must be fed from a remote PC which is connected to one of the embedded systems which is included into the distributed embedded system

These requirements leads to implementation of a distributed embedded systems, each embedded system designated to monitor and control either the sensing or actuating mechanisms with the need for the centralised coordination between the distributed embedded systems. Figure-1 shows various decentralised embedded systems with built-in respective interfaces along with an individual embedded system that provides centralised coordination. It can be seen from Figure-1 that networking should be done in such a way that one of the Microcontroller serves as a central server that regulates the flow of communication between the embedded systems. Some of the microcontrollers are used for just sensing the temperatures within Nuclear reactor tubes and some other are used for controlling the temperatures within the tubes through injecting the coolant into the tubes.

Some of the major requirements that must be met by the distributed embedded applications are shown in the Table-1. All the embedded systems that are selected for implementing various independent functional requirements are to be networked together using CAN protocol which is accepted universally especially for implementing automobile based sensing and actuating system. However it is seen that more of device conversions and protocol conversions are required when CAN is to be used for establishing networking of heterogeneous embedded systems.

The entire distributed embedded system is realized using five micro controller based systems which include 89C51, AT89S52, ATmega328, PIC18F4550 and LPC2148 which are networked to realize various distributed application requirements. All the requirements of the application system have been distributed to micro controller based systems based on the kind of functionality which are related to either sensing or actuating. Four of the micro controller systems are used for implementing respective sensing or actuating functions, one micro controller system is used as a centralized server which is primary responsible for coordinating the functions that are implemented in the respective embedded systems.

## RELATED WORK (LITERATURE SURVEY)

Control area network is a high performance and high reliable advanced serial communication protocol which supports distributed real time control. CAN has been effectively implemented to facilitate communication between various modules [1] that are situated in Automobile systems. Converters have been developed and used for converting communication from non CAN native support to CAN. Not much of attention has been made in speed matching and communication flow as required by the application.

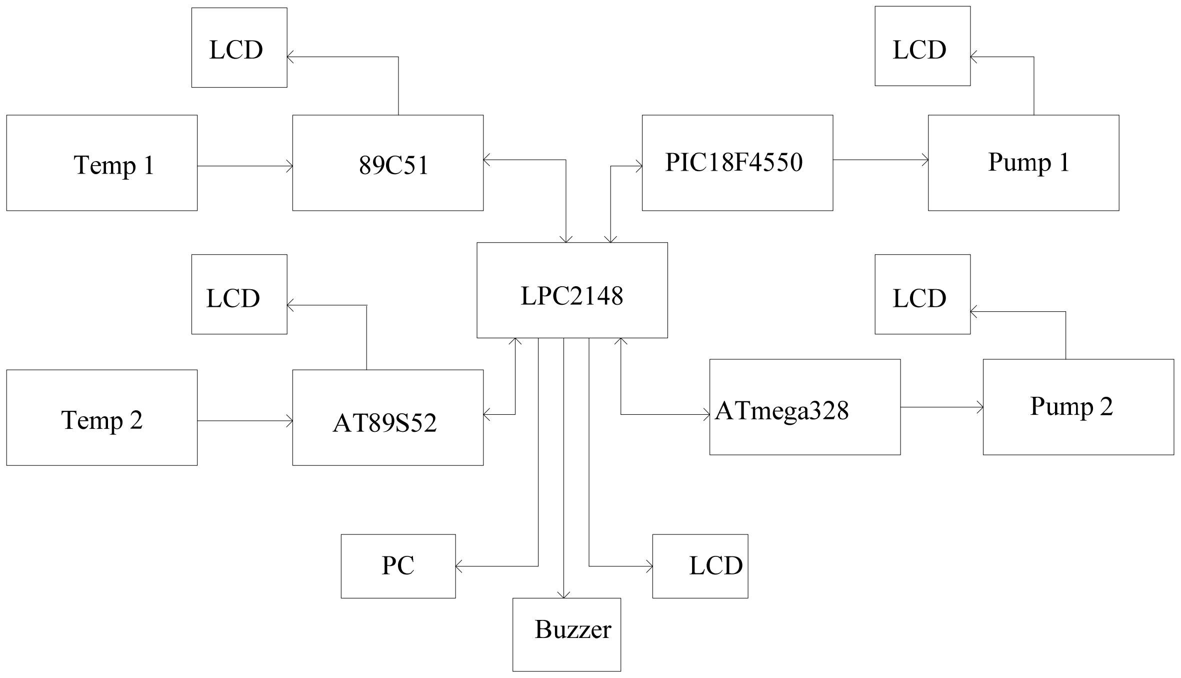

**Figure-1.** Top level view of a distributed embedded system.

**Table-1.** Requirement specification of distributed embedded application.

| Requ. Number | Requirement description |
|---|---|
| 1 | Read Temp-1 and write to LCD |
| 2 | Effect CAN based communication between the 89C51 (System-1) and the Central Microcontroller (System-5) |
| 3 | Read-Temp-1 and send to Central Micro Controller |
| 4 | Read Temp-1 and measure throughput |
|  | Temperature-1 must be sensed at least 10 times per milli second |
| 5 | Effect CAN based communication between the PIC18F4550 (System-3) and the Central Microcontroller (System-5) |
|  | If Temp-1 > Reference Temp-1 then Pump-1 must be on |
|  | If Temp-1 < Reference Temp-1 then Pump-1 must be off |
|  | Compare Temp-1 > temp-2 and if true assert buzzer on |
| 5 | Read Temp-1 and make buzzer off if < Temp-2 |
|  | If Temp-1 > temp-2 then Buzzer is on |
|  | Response time of Temp-1 must be 10µ Seconds |
|  | If Temp-1 > Reference Temp-1 then Pump-1 must be on |
|  | If Temp-1 > Reference Temp-1 then Pump-1 must be off |
|  | If Temp-1 > Reference Temp-1 then Buzzer is on |

| Requ. | Requirement description |
|---|---|

| Number | |
|---|---|
| 6 | Response between the Reading the Temp-1 and stopping the Buzzer must 10μ Seconds |
| | If Temp-1 > Reference Temp-1 then buzzer off |
| 7 | Read Temp-2 and write to LCD |
| 8 | Effect CAN based communication between the AT89S52 (System-2) and the Central Micro Controller (System-5) |
| 9 | Read-Temp-2 and send to Central Microcontroller |
| 10 | Read Temp-2 and measure throughput |
| | Effect CAN based communication between the ATmega328 (System-4) and the Central Microcontroller (System-5) |
| 11 | Read Temp-2 and make pump-2 on if Temp-2 > Reference Temp-2 |
| | If Temp-2 > Reference Temp-2 Pump-2 on |
| 12 | Read Temp-2 and make pump-2 off if Temp-2 < Reference Temp-2 |
| | If Temp-2 < Reference Temp-2 Pump-2 off |
| 13 | Read Temp-2 and make buzzer on if > Temp-1 |
| | If Temp-2 > temp-1 Buzzer On |
| 14 | Read Temp-2 and make buzzer off if < Temp-1 |
| | If Temp-2 > Temp-1 Buzzer On |
| 15 | Response between the Reading the Temp-2 and starting the pump-1 must be 10μ Secs |
| | If Temp-2 > Reference Temp-2 Pump-2 On |
| 16 | Response between the Reading the Temp-2 and stopping the pump-2 must be 10μ Secs |
| | If Temp-2 > Reference Temp-2 Pump-2 Off |
| 17 | The response between the Reading the Temp-2 and starting the Buzzer must be 10μ Secs |
| 18 | If Temp-2 > Reference Temp-2 Buzzer on |
| 19 | The response between the Reading the Temp-1 and stopping the Buzzer must be 10μ Secs |
| | If Temp-2 > Reference Temp-2 Buzzer off |

[2] Have implemented CAN based protocol that includes both static time triggered and dynamic segment for high priority and low priority data transmission. D10 Bit of data segment is used to indicate the assignment of the messages for transmission either in low priority or high priority mode. The communication is designed to be effected in the asynchronous mode. This protocol ensures that the critical messages are sent out before their deadlines and one message hugs the bus. However this protocol loses one of the most important bits.

[3] Have implemented a CAN as an attractive alternative to automate various functions due to its ease, low cast and simplicity in implementation and as such reduces the complexity. The CAN system works effectively even in the presence of electrical and mechanical interferences with the signals used by CAN by adapting different kinds of terminating techniques. Has implemented a project based on PIC technology using CAN for monitoring various parameters such as temperature, CO level etc. No real time aspects as such have been considered in the design of the project.

CAN is making inroads in to every sector. CAN protocol suffers from inherent disadvantage that the low priority devices cannot transmit data for long periods due to low priority attached to the devices at the time of interfacing the devices to the network. [4] Have presented two priority schemes which include a round-robin scheme based priority which gives the access to the bus in round robin fashion. The schemes give an opportunity to low priority devices to transmit. The second scheme implements least recently used priority. The priority of the nodes is rotated based on the amount of usage of the BUS. The proposed schemes are applied at every node which avoids the use of any explicit master node or message scheduler to provide equal opportunity to all the nodes. The proposed schemes do not require changes at physical and data-link layers of CAN protocol. An emergency state has also been considered that allows particular nodes to have provision of emergency states for gaining access to the bus out of turn.

[5] Programmable logic controllers are quite frequently used in the process industries. Many faults have been recognized when PLCs are used using RS485 network. PLCs are being implemented using 32 Bit Microcontrollers and the interconnectivity between the PLCs is through CAN (TJA1050T) interface as CAN is found to be noise free. Each PLC built using the Microcontroller based system works like an intelligent node. Several anti-interference measures by designing appropriate terminating systems have been presented.

CAN based networking is in wide use and it spans across several domains. CAN however cannot be used for some specific domains such as Dependable systems. CAN enhanced Layer (CANELy) [6] architecture has been presented that supports highly dependable systems. This architecture provided the analytic models of CAN operation, based on which both the hardware and software mechanisms for dependable and timely CAN-based network service provisions can be implemented. The mapping of the error monitoring and fault-confinement mechanisms defined by CANELy into hardware has proven to be resource-effective, allowing their integration in an inexpensive Field Programmable Gate Array (FPGA) device. This opened room for cost-effective highly dependable CAN-based solutions in several domains, such as aerospace industry.

CAN based systems are being used quite frequently for home automation to help the elderly and disabled persons to make their own living. CAN helps in establishing a network of home appliances through use of three sub-systems namely Central Controller Subsystem(CCS) [7], CAN module sub-system and relay module sub-system. The central Controller Sub-System is a simple server that provides Graphical user interface. The CAN module connected to CCS acts as master and the CAN modules connected to the home appliances acts as slaves. Master CAN and Slave CAN modules are connected through a CAN bus. The user input signal that contains a CAN data frame broadcasts from master CAN node to all slave CAN nodes via CAN bus. Each data frame consists of an identifier that is compared by each slave CAN node. If the identifier is not matched, the signal will be discarded by the slave CAN node. Otherwise, the slave CAN node will decode the data frame and execute the given command. The relay module of this slave CAN node will either switch ‘ON’ or ‘OFF’ the corresponding home appliances.

The nodes that are interfaced to a CAN bus implements FIFO or priority queues for stacking the messages that are transmitted periodically or sporadically. Few high level protocols such as CANopen, Hagglunds which are termed as HCAN [8] supports the transmission of mixed messages. Using HCAN the existing response-time analysis of mixed type CAN messages can be extended. The extended analysis can compute the response-times of mixed (periodic/ sporadic) messages in the CAN network where some nodes use FIFO queues while others use priority queues.

Using CAN a, a multi-motor control circuit can be implemented. The breakdown of the motor systems is controlled through a Microcontroller based system and DAC [9] which is interfaced to a CAN bus. The Multi-control motor system is interfaced to a CAN bus. The overall structure of the system of hardware design, software design process, and CAN bus controller structure together when implemented will help in implementing multi Motor control system.

Most of the distributed embedded systems have been implemented using bus topology and sharing the Clock. The shared clock algorithms uses CAN base microcontrollers to implement time-triggered network architectures [10]. The bus based topology is generally used for implementing time triggered processing. Star topology is considered to be fault tolerant more than the bus topology. The comparative analysis of implementation of time triggered systems using star and bus topology has been done and it is proved that the star topology is efficient when implemented through a Modified S-C algorithm.

Shared Clock (S-C) algorithms are used in conjunction with off-the-shelf microcontrollers based systems in the development of distributed embedded systems. All the SC designs are based on BUS topology. A novel SC algorithm [11] has been presented which is intended for establishing a distributed embedded system using star topology. Through a fault-injection case study and quantitative comparisons, it is demonstrated that the star-based design has number of advantages when compared with a bus-based equivalent.

The Controller Area Network (CAN) protocol is a real-time, serial, broadcast protocol with a very high level of security. The AT90CANl28 CAN controller is fully compatible with the CAN specification but its internal structure is different from independent CAN controllers. A method [12] has been presented to set the baud rate and other parameters within AT90CAN128 CAN controller so that the communication can be properly effected through the CAN Module.

CAN bus are being used increasingly in wide range of applications for its superiority, but CAN cannot be used directly to communicate with a PC. Conversion of RS232 to CAN bus protocol will help connecting a PC to a CAN network. A converter is presented [13] using a PIC18F2580 Microcontroller and SJA1000 CAN controller which can be used for establishing communication between a PC and a Microcontroller based system.

A controller area network (CAN) is ideally suited to the many high-level industrial protocols embracing CAN and ISO 11898 as their physical layer. Tremendous flexibility in system design can be implemented considering the cost, performance, and upgradeability. A brief introduction to the CAN operating principles, the implementation of a basic CAN bus using Texas instrument’s, CAN transceivers and DSPs, has been presented [14]. A discussion of the robust error detection and fault confinement mechanisms has also been presented. The CAN has also been extended called eCAN and the properties of the same have also been presented.

[15] A new system design which uses CAN2.0B specification is presented. The system design focusses on hardware and software design of intelligent node built using a Microcontroller S3C44B0, MCP2510 CAN controller, which is a full CAN protocol controller implementing CAN specification V2.0 A/B and receive standard and extended messages. In addition, the TJA1050 is used as the interface between the Controller Area Network (CAN) protocol controller and the physical bus.

VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

A fixed priority DM algorithm is used to assign the priority for each message and get a better result. In software, UC/OS-II multitask operation system is used which is a real time operating system suited for industrial control and better than anyone else.

In all the literature references, developing a CAN based network using heterogeneous microcontroller based systems have not been presented. The message flow system specific to an embedded distributed application has not been discussed much. None have dealt the issue of arbitration considering the sequence of messages as required by the distributed application. The design of datagrams that must be transmitted commensuration to the expected message flow has not been discussed.

## INVESTIGATIONS AND FINDINGS

### Networking heterogeneous embedded systems

A CAN based networking system has to be established for connecting several distributed embedded systems into a network. The central micro controller system is expected to be in a remote location. As per the description of the functional requirements the central micro controller shall have to act like a single master and the rest as slaves.

The communication between the master and the slave requires a speed of 40kbps which allows the signals to be driven to a distance of more than 1KM which a sufficient considering the application in view. Due to the increase in bus length biased-split termination method has to be followed while establishing the network. The higher level network showing the connectivity between the distributed embedded systems using CAN as a backbone is shown in Figure-2.

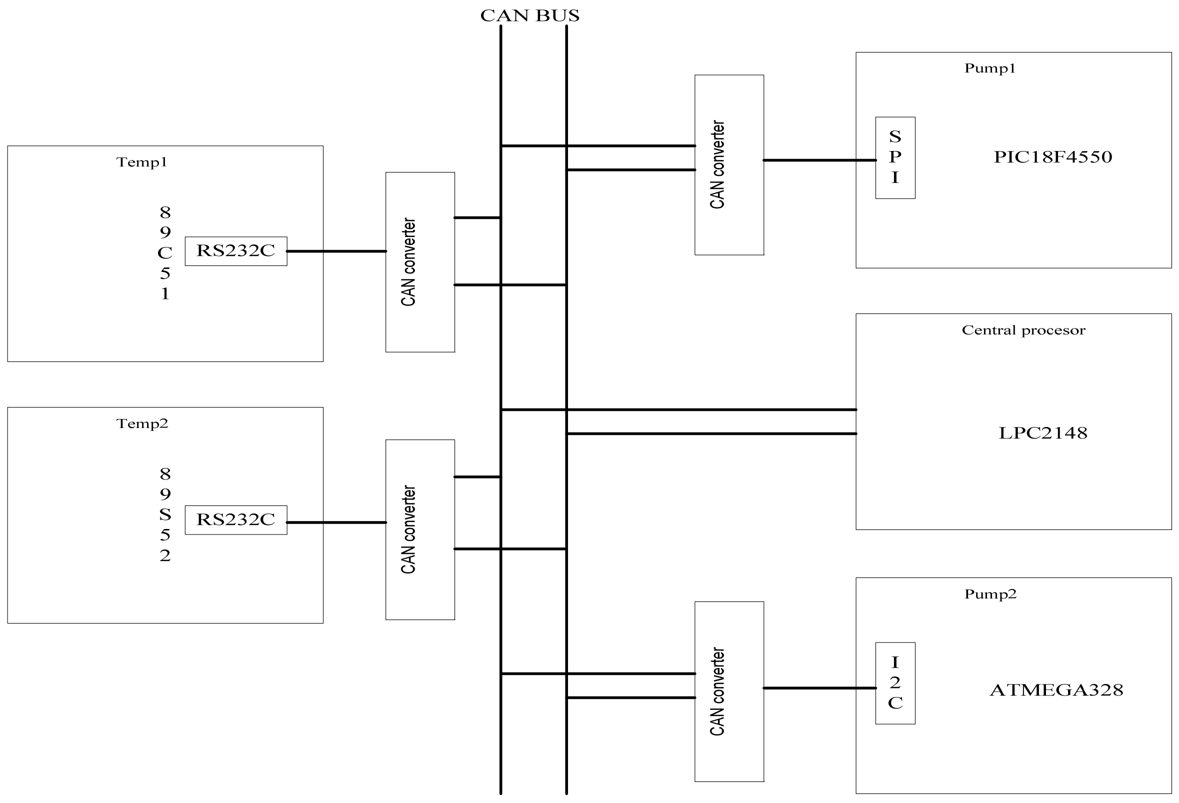

**Figure-2.** CAN based networking of heterogeneous embedded systems.

The CAN based network chosen for the application contains a single master and 4 slaves. Conversion mechanisms are to be considered when the micro controller based systems which are used as a part of network is heterogeneous due to the reason that they do not have native CAN interface.

As per the functional requirements of an application, LPC2148 a 32 Bit Microcontroller is used as a master device for achieving communication between the slave devices and the master. The master device must also be designed to alert local user through triggering a Buzzer about the variations taking place with the temperature gradients. The master system must also be designed for interfacing with a PC for communicating with it for obtaining the reference temperatures and transmitting the process data to be stored in a database.

4 slave microcontroller based devices which include 89C51, AT89S52, PIC18F4550 and ATmega328 have been considered for implementing various functions that are projected as requirements which include sensing

temperature-1, sensing temperature-2, starting and stopping pump-1, starting and stopping pump-2. In those slave systems 89C51 and AT89S52 don't have the CAN support.

For interconnecting these systems, conversion mechanisms are required. 89C51 and AT89S52 have been designed with inbuilt RS232C communication interfaces, A device CAN / CAN open / RS232 - Converter has been used for converting RS232C signals to CAN and Vice Versa.

The other two microcontroller based systems which include PIC18F4550 has inbuilt SPI interface which is converted into CAN using the converter SPI-CAN and Microcontroller ATmega328, is used by converting built-in I$^2$C to CAN using a converter.

Thus the heterogeneity that exists due to varying interfaces for communication is tackled through choice of appropriate converters. The application specific communication requirement requires transfer of data at the rate of 19K which is supported by the interfaces and the converters used in the designing of the distributed embedded system.

## Architectural design for message flow across the distributed embedded systems (pilot project)

The networking diagram shown in the Figure-2 shows the interfacing of the various heterogeneous microcontrollers based systems which are interconnected through a CAN based protocol system. However communication software resident in different microcontroller based system is required for achieving application specific messaging requirements using the network designed for the purpose.

The communication has to be initiated by the master by using RTR (Remote transmission request) for want of Temperature-1 and Temperature-2 to be transmitted by 89c51 and AT89s52 in that sequence. The throughput, sequencing and timing of receipt of the temperatures are designed and developed into master device.

The applications on 89C51 and AT89S52 will have software components to receive the master requests and transmit the data to the master device. The communication components implements RS232C serial communication system for transmitting and receiving the temperature data.

The master device at the start-up receives the reference temperatures from PC which is connected to the master through RS232C serial communication system. The sensed temperatures are compared with the reference temperatures and in the event that the sensed temperatures are more than the reference temperature, a message is sent to the Microcontroller based systems that operate the pumps to be on or off.

On the master side, two individual software components for each of the pump controller system shall have to be in place for transmission of the commands and reception of acknowledgement that the intended pump operation has been achieved successfully or otherwise. The communication in this case is achieved through use of CAN interface.

The software components that are designed for effecting the communication between the master and the pump control slave devices is achieved through implementation of the CAN protocol.

The master also is provided with a component that computes the temperature gradient and asserts a buzzer or otherwise if the temperature gradient is beyond the prescribed limits. This function as such requires no communication as the entire functioning is implemented within the master device.

The software architecture that depicts the application specific communication is shown in the Figure-3.

It can be seen from the Figure that the central microcontroller based system has been provided with necessary interfaces for effecting CAN based communication with other microcontrollers which use a different interface all together.

## Designing the device identifiers for effecting optimized message flow

CAN protocol do not support specific addresses assigned to the Microcontroller based systems. CAN supports many masters and a master must arbitrate to gain control of the bus. Each of the device on the CAN bus can be provided with identifier bits through toggle switches and the identifier bits more or less identify the priority with which a device can access the bus.

During the phase of arbitration, each transmitting device transmits its identifier and compares it with the level on the bus. If the levels are equal, the device continues to transmit.

If the device detects a dominant level on the bus while it is trying to transmit a recessive level, it quits transmitting and becomes a receiving device. After arbitration only one transmitter is left on the bus, and this transmitter continues to send its control field, data field, and other data.

Aprori identifiers are allocated to the devices based on the transmission priority assigned to a device. The device has the highest priority to transmit the message based on the message flow requirement are assigned with a dominant identifier sequence.

The identifiers thus are allocated based on the flow of messages as required by the application system. Table-2 shows the details of the identifiers allocated to the devices which included into the CAN based network.

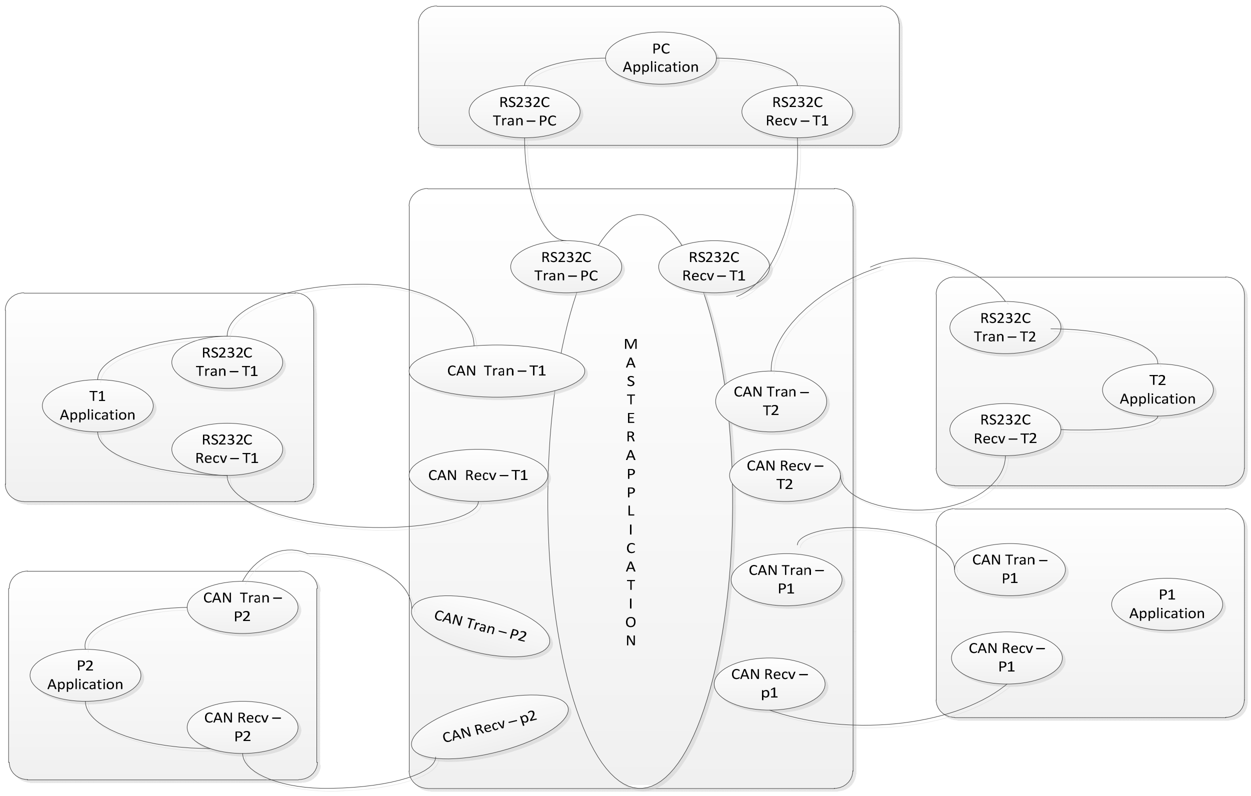

**Figure-3.** Communication system architecture.

**Table-2.** Identifiers allocation.

| Serial Number of device | Type of device | Device Model Number | Allocated identifiers | Transmission /reception priority | Reason for assigning the priority |
|---|---|---|---|---|---|
| 1 | **Master** | **LPC2148** | **11010101000** | **1** | Master has the priority over the salves |
| 2 | Slave-1 | 89C51 | 11010101001 | 2 | Temp-1 flows before other messages |
| 3 | Slave-2 | AT89S52 | 11010101011 | 3 | Temp-2 must follow temp-1 in a fraction of 10µsec |
| 4 | Slave-3 | PIC18F4550 | 11010101100 | 4 | Message to pump-1 must follow temp-2 within 20µsec |
| 5 | Slave-4 | ATmega328 | 11010101110 | 5 | Message to pump-2 must follow the message to pump-1 within 10µsec |

The arbitration between all the devices considering that all of them as masters are shown in the Figure-4.

VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

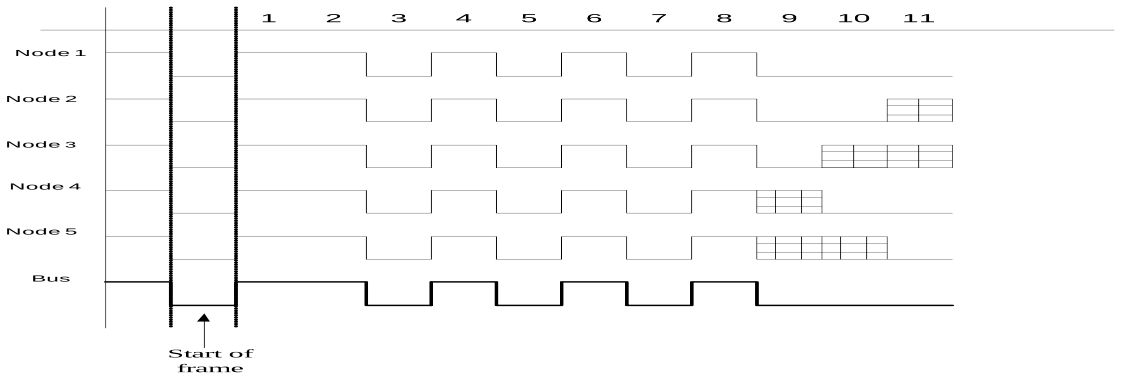

**Figure-4.** Identifier sequences for arbitration.

At bit 9 Node-4 and Node-5 are in receiving mode and therefore shall move on to receiving state. At bit 10 Node-3 is in the recessive mode and therefore move on to receiving mode. At the 11th bit Node-2 will go on to recessive mode and therefore move on to recessive mode. Thus finally it is the node-1 that wins to gain the access to the bus. Node-1 is the central processor and therefore it has the access to the bus to start the communication.

In the absence of Node-1, Node-2 will have the access to the bus. Node-2 in this case is the Microcontroller that senses the temperature and transmits the temperature to the central server.

In absence of Node-1 and Node-2, Node-3 that controls the Pump-1 will have the access to the bus and similarity the access further is gained by the Pump-2. This design of the identifiers will regulate the flow of the messages to the bus.

**Design data packets for effecting communication**

To effect communication between the server and the slave nodes, data packets are to be designed which has been included in it, the identifier that transmits the data on the bus. The protocol implemented on the server side starts the communication by sending a data frame as the central server's identifier is dominant and therefore will have the access to the bus.

The slave that must respond with the data will have the next prioritized identifier and therefore will have the access to the bus. The slave will transmit the data required by the master. The requests for the data and responses to the requests are channelled as per the message flow required and as stated in the distributed application. The data field will have data which is either the command initiated from the master or the data corresponding to the command requested when the data packet is initiated by the slave. Figure-5 shows the sequence in which the data packers are transmitted which depicts the actual messages that must be flown according to the application requirement.

Data frame design of Temp-1 processing system.

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 000 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 001 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

Data frame design of Temp-2 processing system.

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 000 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 011 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

Data frame design of Pump-1 processing system.

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 000 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 100 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

Data frame design of Pump-2 processing system.

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 000 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

| SOF | Identifier | RTR | Control | Data | CRC | ACK | EOF |
|---|---|---|---|---|---|---|---|
| 0 | 11010101 110 | 1 | 000011 | 3 bytes | 15-bit | 11 | 1111111 |

**Figure-5.** Design of data frames as per message flow.

## EXPERIMENTAL RESULTS

Experiments have been conducted using the CAN network designed and the distributed embedded application system and the communication system implemented. Communication is effected by making the data flow as per the communication system architecture. The results of the experimenting conducted are shown in the Table-3. The experimental results have also been validated by using PROTEUS simulator.

**Table-3.** Experimental results.

| Transaction ID | From | | | To | | | Whether Checksum error exists |
|---|---|---|---|---|---|---|---|
| | Microcontroller system | Microcontroller system identifier | Number of bytes sent | Microcontroller system | Microcontroller system Identifier | Number of bytes Received | |
| 1 | 89C51 | 11010101001 | 2 | LPC2148 | 11010101000 | 2 | No |
| 2 | AT89S52 | 11010101011 | 2 | LPC2148 | 11010101000 | 2 | No |
| 3 | LPC2148 | 11010101000 | 1 | PIC18F4550 | 11010101100 | 1 | No |
| 4 | LPC2148 | 11010101000= | 1 | ATmega328 | 11010101110 | 1 | No |

## CONCLUSIONS

Networking has be undertaken for each of the distributed embedded system individually even though the same standard protocol like CAN is used due to lack of native support for CAN within the heterogeneous microcontroller based system which are used for networking. The flow of messages across the distributed embedded system also varies from application to application and therefore needs to be designed separately. Flow of messages must be sequenced and the sequence of flow of messages can be controlled through setting appropriate identifiers that are used by the devices for arbitrating the bus. Data packets are also to be signed considering the sequence of flow of data and there needs to be designed accordingly.

## REFERENCES

[1] Mansi Desai, Rahul Shetty, Viraj Padte, Mayur Parulekar, Samuel Ramrajkar. 2013. Controller Area Network for Intelligent Vehicular Systems. IEEE conference publications. pp. 1-6.

[2] Chin-Long Wey, Chung-Hsien Hsu, Kun-Chun Chang, and Ping-Chang Jui. 2013. Enhancement of Controller Area Network (CAN) Bus Arbitration Mechanism. International Conference on Connected Vehicles and Expo (ICCVE). pp. 898-902.

[3] Presi. T. P. 2013. Design And Development Of PIC Microcontroller Based Vehicle Monitoring System Using CAN. IEEE Conference Publications. pp. 1070-1076.

[4] Pranjal Vyas, Swaroop A. Hangal, Leena Vachhani. 2012. Methods for equitable distribution of bus access among nodes of Controller Area Network. IEEE Conference Publications. pp. 1-6.

[5] Jinshui Shi. 2012. The Implementation of CAN Bus Network of PLC Based on ARM. 4th International Conference on Intelligent Human-Machine Systems and Cybernetic. 1: 268-270.

[6] Ricardo Pinto, Jose Rufino, Carlos Almeida. 2011. High Availability in Controller Area Networks. IEEE Conference Publications. pp. 1-6.

[7] Kent How Teh, Wei Lun Ng, Chee Kyun Ng and Nor Kamariah Noordin. 2011. Home Appliances Management System using Controller Area Network (CAN). 17th Asia-Pacific Conference on Communications (APCC). pp. 899-904.

[8] Saad Mubeen, Jukka Maki-Turja and Mikael Sjodin. 2011. Extending Response -Time Analysis of Controller Area Network (CAN) with FIFO Queues for Mixed Messages. IEEE Conference Publications. pp. 1-4.

[9] Shanshan Wang, Bo Yin, He Li, Qingshu Yang. 2011. Multi-motor Control System Design Based on Can Bus. International Conference on Electronic and Mechanical Engineering and Information Technology. pp. 4910-4913.

[10] Amir Muhammad, Michael J. Pont. 2010. A Time-Triggered Communication Protocol for CAN-based Networks with a Fault-Tolerant Star Topology. IEEE Conference Publications. pp. 2347-2354.

[11] Amir Muhammad, Devaraj Ayavoo, Michael J. Pont. 2010. A Novel Shared-Clock Scheduling Protocol for Fault-Confinement in CAN-based Distributed Systems 5th International Conference on System of Systems Engineering. pp. 1-6.

[12] MinLi Suo-liang Zhang, Cheng-hao Han, Zhi Ca. 2010. The CAN Bus Baud Rate Setting Method Based on AT90CAN128. International Conference on Computer, Mechatronics, Control and Electronic Engineering. 3: 542-545.

[13] Xianjun Wang and Wencheng Guo. 2009. The Design of RS232 and CAN Protocol Converter Based on PIC MCU. Computer and international science journal. 2: 3: 176-181.

[14] Hanxing Chen, Jun Tian. 2009. Research on the Controller Area Network. IEEE Conference Publications. 2: 251-254.

[15] Yanxia Liu, Shufen Li, Yuqiu Song. 2009. Application of can-bus in embedded vehicle body control system. IEEE Conference Publications. 2: 261-264.