Indian Journal of Science and Technology, *Vol 8(15), 68739, July 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

I^2C based Networking for Implementing Heterogeneous Microcontroller based Distributed Embedded Systems

J. K. R. Sastry*, J. Viswanadh Ganesh and J. Sasi Bhanu

Department of Electronics and Computer Science Engineering, KL University, Vaddeswaram, Guntur District – 522502, Andhra Pradesh, India; drjkrsastry@gmail.com, jjvganesh@gmail.com, sasibhanu@kluniversity.in

Abstract

Realizing a distributed heterogeneous embedded system using I^2C requires investigations and findings related to designing of networking, architecture, message design and flow for prioritization, and datagram design. The existing methods for effecting communication among the heterogeneous embedded systems interconnected through I^2C communication system have not addressed the heterogeneity issues. To suffice, new methods for designing Heterogeneous Embedded networks, communication systems, message flow systems and the message design have presented. The methods have been applied to an existing distributed embedded system and the results obtained proved to be exact as required. Every distributed embedded system is different and the kind of Hardware and protocol conversions required is dependent on the type of distributed embedded system. It has been shown in the paper the kind of conversions that must be undertaken considering I^2C as the communication method and a set of selected heterogeneous embedded systems that are used for developing an application to monitor and control the temperatures within a Nuclear reactor system. The way the communication must be effected is dependent on the number of masters and the slaves that must be supported on the network. A communication system architecture that suits the pilot project has been presented. A design flow method which uses priority queues has been presented to effect the communication according to the flow required. The design of datagrams required for effecting the communication has also been presented in the paper.

Keywords: Architecture, Distributed, Heterogeneous, Embedded, I^2C, Message Flow

1. Introduction

Distributed computing architectures offer numerous advantages in the development of complex devices and systems. These advantages include well-defined interfaces, flexible composition, streamlined integration, straightforward function-structure mappings, standardized components, incremental testing, and other benefits.

Embedded systems are being extensively used in monitoring and controlling various physical parameters. Embedded systems are reactive that they respond to changes taking place in the external environment. Almost all electronic gadgets (which include digital cameras, washing machines etc) are being operated using an embedded system built into it. Embedded systems are also being 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 require interconnecting of many individual embedded systems. An automobile system as such has in it many embedded systems which individually deals with controlling of breaks, doors, mirrors, rare and front object indicators, engine temperature, wheel speed, tyre pressure, DVD control etc. These individual

*Author for correspondence

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 ECU's that are used for monitoring and controlling any one of the aforesaid 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 protocols which include I²C, CAN, USB, and RS485 etc., are being used for interconnecting various embedded systems. Each type of networking leading 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 some microcontroller based systems. However, many Microcontroller based systems exists that do not provide support for a native protocol, making it difficult to establish a seamless network. In a distributed embedded network many heterogeneous embedded systems are used and to connect them on to standard network, protocol conversion would be required quite frequently which generally through a problem of speed matching and synchronisation.

The standard protocols do not address any 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 requirements and these needs to be addressed separately through specific designs.

Controlling the flow of messages across the microcontroller based system is most important and the standard protocols as such do not provide any support for that. The message flow system must consider the sequence in which the messages flow across the network connecting the distributed systems. It is also necessary that transmission system that includes data communication must be designed specifically for each of the distributed embedded system individually.

1.1 Problem Definition

A distributed embedded system involves use of individual microcontroller based systems. Each microcontroller 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 based 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 issue that must be considered to achieve the networking of embedded systems is 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, FireWire, USB, CAN, I²C, ETHERNET, PCI, and ESA etc. Among all, I²C bus based serial communication protocols are used for establishing a network connecting all the individual microcontroller based systems. I²C is as 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 I²C based system is due to lack of native support within some of the microcontroller based systems. This requires frequent protocol conversions. Every distributed embedded system requires different communication system architecture and the very communication systems must be customized for implementation of specific distributed embedded system. 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 USB 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.

Design of message flow across various microcontroller based systems is equally important and the same is to be achieved independently and specifically customised for each of the distributed application.

Even the data communication system must be designed independently.

2. Specification Description of Distributed Embedded Application

Monitoring the temperatures within the nuclear reactor tubes is one of the most important issues when it comes to uranium enrichment. Sensors are mounted on to the nuclear reactor tubes which are distantly situated. Many temperatures at various points within each of the Nuclear reactor tube must be sensed and it is also necessary to maintain proper gradients across various points at which the temperatures are measured. When temperature rises above some pre-defined levels, coolants have to be injected into the tubes to bring the temperature down. Pumps are used for injecting the coolants into the tubes. The temperature sensing and implementing the actuating mechanisms that control the process of pumping is achieved through various embedded systems. The operators must be alerted when the temperature gradient goes beyond uncontrollable levels through asserting a buzzer and lighting a pattern of LEDs as the case may be.

A historical database of temperatures sensed, pumping levels implemented, temperature gradients, status of triggering buzzer etc., are written on to a PC into a Database for providing the historical evidences. Each part of sensing and actuating requires a kind of response time and therefore needs to be sensed, monitored and controlled individually through a separate embedded system. There is a need for coordinating the functions between the individual embedded systems for achieving the sensing and actuating in real time. This leads to the need for interconnecting the individual embedded systems that help in establishing the communication between the embedded systems which are individually responsible for either sensing, actuating or monitoring the process taking place within the Nuclear reactor system.

Designing, development and implementing the Networking of embedded systems becomes one of the most crucial issues when it comes to designing, development and implementing the distributed embedded systems. One of the major issues that must be addressed is heterogeneity that exists among different types of Microcontroller based systems which are used for developing and implementing different parts of a distributed embedded system.

These requirements leads to implementation of distributed embedded systems, each 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 is the block diagram that shows various decentralised embedded systems with built-in respective interfaces along with an individual embedded system that provides centralised coordination.

Some of the major requirements that must be met by the distributed embedded applications are cited in the Table 1.

3. Related Work

Researchers at Santa Clara University (SCU) have proposed[1] a distributed computing architecture for small or multi-spacecraft missions. This architecture extended existing I²C, Dallas 1-wire and RS232 data protocols and was adaptable to a number of microcontrollers. Since then, that architecture has been implemented on six university-class space missions at three different universities. Many types of architectures have been recommended by the universities all aimed at developing I²C communication network connecting various types of Microcontroller based systems. Each architecture is different and have considered different kinds of challenges related to project size, scope and infrastructure. The details of three architectures Akoya-A/Bandit-A and Akoya-B/Bandit-C, EMERALD and ONYX at SCU and FASTRAC and ARTEMIS have been presented. These architectures have shown the way standard distributed embedded systems can be used to develop custom specific I²C based networking systems.

I²C protocol has been an Industry standard being used for implementing communication between I²C enabled devices and the system. Many Microcontroller based systems have implemented native support for I²C. However lot programming and knowledge of internal systems of

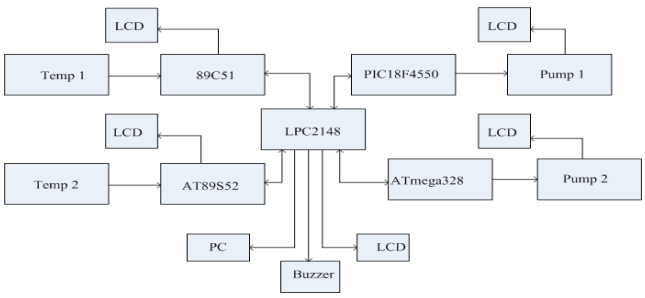

Figure 1. Top level view of a distributed embedded system.

Table 1. Requirement specification of distributed embedded application

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

(Continued)

microcontrollers which have provided internal support is needed[2]. FPGA based implementation of networking of the embedded systems is simple and straight forward. Any Microcontroller based system can be configured as slave or master on an I²C bus, by using FPGA based networking. The solution however is not extendable and has not considered many important issues related I²C based communication.

A three level protocol design has been considered which include protocol level, signal level, and interface level. The reusability of all the three level for each of the new device that can be brought on to a network could be undertaken[3]. Protocol level can be reused without the need for any modification and the signal level re-usability can be reused by setting the number of bytes to be transferred and the interface level can be achieved through change of operating modes of the interface. The reusability at the interface level is more complicated as it involves change of operating modes. FPGA based implementation of the interface level protocol makes it easier to implement reusability at this level. This design approach makes the process of networking of embedded systems more complicated in the name of reusability.

The embedded systems are highly optimized to perform limited duties of particular needs. They can be control, Process, medical, signal, and image processing applications. The challenges faced by embedded systems are security, real-time, scalability, high availability and also performance based interoperability as more and more different devices are added to the systems. These complex ubiquitous systems are glued together with layers of protocols. Networking of these is a task to look for with minimum flaws in manageability, synchronization and consistency. A gateway has been presented[4] to interconnect various systems that support UART with SPI, I²C and CAN Protocols. The gateway approach included another level of complexity and the protocol conversion related logics have to be implemented whether needed or not.

Another important research is in progress to interconnect two different networks say USB based and I²C based networks. The mapping between I²C protocol and the USB based protocol has been undertaken[5]. The hardware design structure of the schematic mapping has been presented. The gate way has been developed and presented considering the hardware design and the software architecture.

Many applications that include flight control, banking, medical, and other high assurance systems have a strict requirement on correct operation. Fundamental to this is the enforcement of non-interference between subsystems that are connected on to a network. In an effort to help guarantee this policy, recent work has emerged with tracking information flows at the hardware level. A specific method known as Gate-Level Information Flow Tracking (GLIFT) has been presented that tests information flows in two common bus protocols, I²C and USB. It has been shown that the protocols do elicit unintended information flows and a solution based on Time Division Multiple Accesses (TDMA) that provably isolates devices on the bus from these flows[6].

Several distributed architectures have been implemented in the field of robotics[7]. A closed loop, real time multi tasked controller has been implemented through an asset of networked microcontroller based embedded to be used in a robotic has been presented. The control architecture is distributed in five microcontrollers with master slave scheme. The master unit is dedicated for the network management and the communication with the user-interface.

The design of embedded board focuses on the position control of the corresponding joint. An implementing embedded control for brushless motor is established. I²C protocol is implemented for the network management. A user-interface for monitoring and control is developed. The control is done through an USB communication assured with compatible drivers on the three most popular platforms (windows, Linux and Mac OS). The approach is completely application specific and cannot be used for any other purpose. The model did not discuss about the information flow across the network.

In embedded system designing and managing communication among various bus interfaces and attaching multiple systems with different interfacing protocols to a main processor has been one of the challenging tasks. Popular serial interfacing protocols include: USB, I²C, SPIISSP, CAN and UART is used for communication between integrated circuits for low/medium data transfer speed with on board peripherals.

A platform has been presented for implementing some of the serial protocols which are presented by a low power 32-bit ARM RISC processor (LPC2148)[8]. This approach literally leads to a centralized star based topology and therefore is non extendable. Increasing the length of the network is not quite possible.

I²C protocol provides easy communication without data loss. It also gives excellent speed compared to other protocols. I²C uses only two wires for communication. It is light weight, economical and omnipresent. It also increases data transfer rate. A new protocol has been developed that considers high speed communication through use of control registers inside the devices as well as the data that can be saved into the internal registers. Using this process various control parameters can be monitored and controlled[9]. The design method has been implemented on a FPGA. This approach does not consider any heterogeneous issues.

Distributed computing architectures offer numerous advantages in the development of complex devices and systems. The design, implementation and testing of a distributed computing architecture for low-cost small satellite and multi-spacecraft missions which is composed of a network of PIC microcontrollers linked together by an I²C serial data communication bus has been presented[10]. The system also supports sensor and component integration via Dallas 1-wire and RS232 standards. A configuration control processor serves as the external gateway for communication to the ground and other satellites in the network. The processor runs a multitasking real-time operating system and an advanced production rule system

for on-board autonomy. The data handling system allows for direct command and data routing between distinct hardware components and software tasks. This capability naturally extends to distributed control between spacecraft subsystems, between constellation satellites, and between the space and ground segments. A technical design incorporating various features has been presented. This approach has used a single protocol system and has not addressed any of the issues related to heterogeneous implementation.

From the literature it can be seen that none have attempted to establish distributed embedded networks that takes into account the heterogeneous aspects of various types of microcontroller based systems. Even the design of message flow that is needed with a specific distributed system has not been attempted. None have discussed the communication architectures that consider heterogeneous embedded system have not been discussed in the literature.

4. Investigations and Findings

4.1 Designing I²C based Networking for a Heterogeneous Embedded System

I²C based networking is one of the methods that exists today for establishing interconnecting various embedded systems. I²C is frequently used protocol for effecting communication between the Computing stations and peripheral devices and now even being used for establishing a communication network that connects various Microcontroller based systems. Many of the Microcontroller based systems have no native support for I²C while some have. Most of the Microcontroller based system differs in many ways (word boundary, endian, byte addressing, parity, word length, number of registers etc). Networking such heterogeneous embedded systems through a challenge and many innovative approaches are required for establishing the networking of the same.

The generic way of connecting the heterogeneous embedded systems is shown in the Figure 2.

An I²C based communication system helps in achieving a network interconnecting a set of heterogeneous distributed embedded systems. Every distributed embedded system is different and a dedicated network has to be designed and developed. The designing of the networking needs application specific requirements. The application specific requirements related to the Nuclear reactor application are shown in the Table 1.

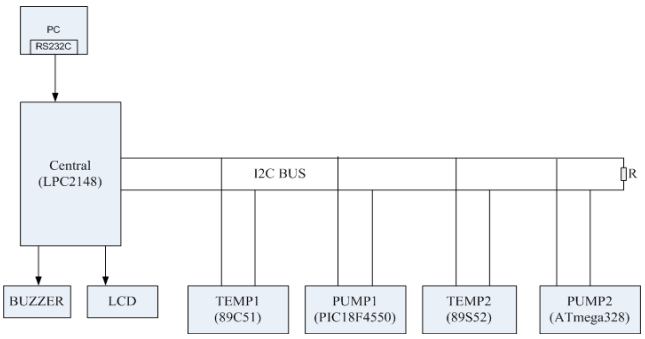

Figure 2. I²C based Networking.

Table 2. Address allocation to the devices

Serial Number of Device	Type of Device	Device Model Number	Allocated Address	Transmission Reception Priority	Reason for Assigning the Priority
1.	Master	LPC2148	70	1	Master has the priority over the salves
2.	Slave-1	89C51	60	2	Temp-1 flow before other messages
3.	Slave-2	AT89S52	50	3	Temp-2 must follow temp-1 in a fraction of 10µsec
4.	Slave-3	PIC18F4550	40	4	Message to pump-1 must follow temp-2 within 20µsec
5.	Slave-4	ATmega328	30	5	Message to pump-2 must follow the message to pump-1 within 10µsec

The central microcontroller system is expected to be in a remote location. As per the description of the functional requirements, the central microcontroller 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 100Kbps to 1Mbps which allows the signals to be driven to a distance of more than 1kms which is a sufficient requirement of the distributed embedded application.

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. It consists of native I²C support. The master device must also be designed to alert local user through triggering a Buzzer about the variations taking place within 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.

Three of the Microcontroller based systems (LPC2148, ATmega328 and PIC18F4550) have in-built I²C communication interface and the remaining two Microcontroller based systems (89c51 and 89S52) have in-built RS232C interface for effecting communication. These RS232C interfaces are to be converted into I²C and vice versa using the converter SP16IX752. The devices implements buffering techniques for converting a 19.2 Kbps speed which is the maximum speed achievable through a RS232C serial communication system into to 1Mbps speed. This conversion is good enough as the amount of data to be transmitted from a Slave to Master and Vice Versa is not more than 18K bytes considering throughput for sensing and transmission is not more than 9K temperatures/second which is more than sufficient for the application to be implemented. The designing of the I²C network considering the heterogeneous interfaces has been shown in the Figure 3.

4.2 Designing Communication System

The networking diagram shown in the Figure 3 shows the interfacing various heterogeneous microcontrollers based systems which are interconnected through an I²C 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

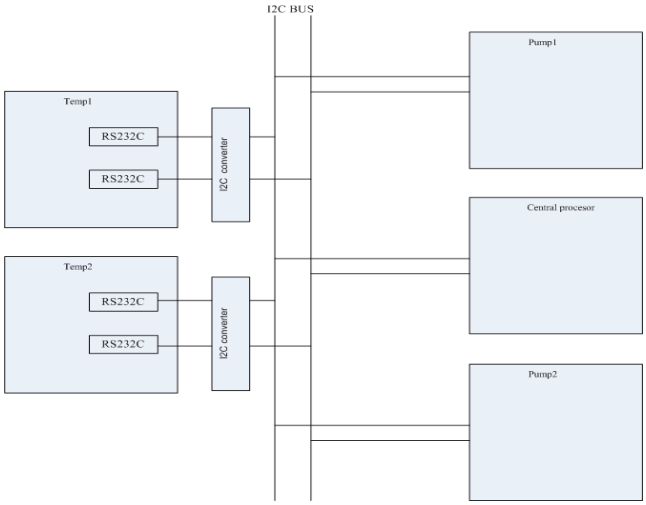

Figure 3. USB based Networking for Nuclear reactor system.

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 system that operates 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 I²C 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 I²C 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 4.

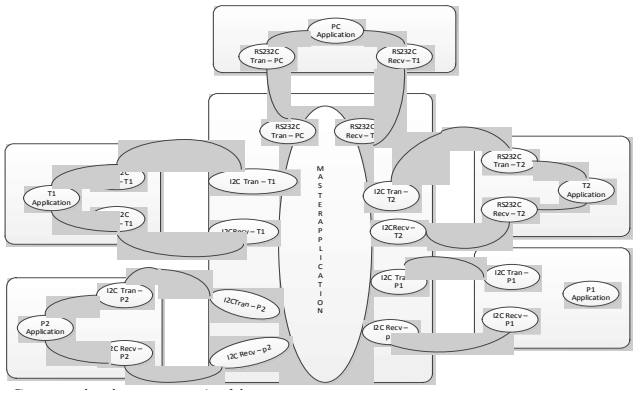

Figure 4. Communication system Architecture.

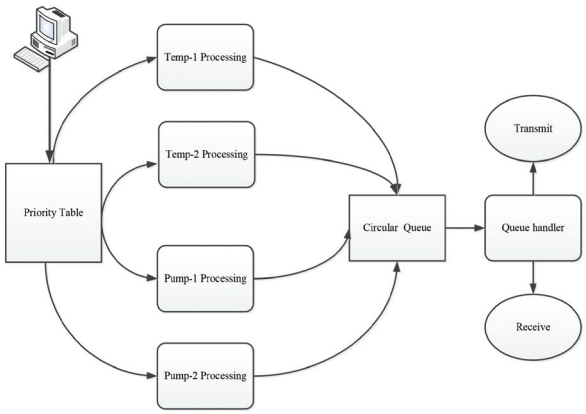

Figure 5. Priority based message dispatching method.

4.3 A Novel Message Flow Design Method

In I²C based communication, one of the connected devices has been fixed as the master even though I²C supports multiple masters. Every communication is initiated from the master.

Every slave is assigned with an address right in the beginning through assigning address values to a memory location through an application running at the slave side. The address as such has no priority assigned for having preference to communicate over the network. The addresses assigned do dictate the flow of control of the data on the network as per the distributed application requirement. The addresses assigned to the slaves are predecided and coded into the individual slave applications. The design of the addresses that can be allocated to the salves is shown in the Table 2.

However the messages between the master and the slaves need to flow in a fixed pattern meaning as per prefixed priorities. The Table 2 shows the priorities to the message flow across the master and the salves. The communication software running on the master, will post a message as per the priority to a circular queue and the circular queue handler will despatch the messages as per the queue in circular fashion. The working of the circular queue based dispatching system for effecting the flow of control of messages as required by the distributed embedded application is shown in the Figure 5.

Table 3. Flow of data from Master Microcontroller (LPC2148) to Temparature-1 Microcontroller (89C51)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0111100	0	1 bit	2 bytes	1 bit

4.4 Design of Data Flow for Communication

For the TMCNRS system which connects 5 Microcontroller based systems through I²C, Only one Microcontroller acts as master and the rest as slaves. Communication takes place both ways between the master and the slaves. The flow of data between the master and the salve have been designed considering the requirements as stated in the Table 1 related to the pilot project cited in this paper**.** The data flows have been shown in the Tables 3 to 10.

Table 4. Flow of data from Temparature-1 Microcontroller (89C51) to Master Microcontroller (LPC2148)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0111100	1	1 bit	2 bytes	1 bit

Table 5. Flow of data from Master Microcontroller (LPC2148) to Temperature-2 Microcontroller (89S52)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0110010	0	1 bit	2 bytes	1 bit

Table 6. Flow of data from Temperature-2 Microcontroller (89S52) to Master Microcontroller (LPC2148)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0110010	1	1 bit	2 bytes	1 bit

Table 7. Flow of data from Master Microcontroller (LPC2148) to Pump-1 Microcontroller (PIC18F4550)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0101000	0	1 bit	2 bytes	1 bit

Table 8. Flow of data from Pump-1 Microcontroller (PIC18F4550) to Master Microcontroller (LPC2148)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0101000	1	1 bit	2 bytes	1 bit

Table 9. Flow of data from Master Microcontroller (LPC2148) to Pump-2 Microcontroller (ATMEGA328)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0011110	0	1 bit	2 bytes	Bit

Table 10. Flow of data from Pump-2 Microcontroller (ATMEGA328) to Master Microcontroller (LPC2148)

Start bit	Slave Address	R/W	Acknowledgement	Data bits	Stop bit
1bit	0011110	1	1 bit	2 bytes	Bit

5. Experimental Results

Experiments have been conducted making data flow from one Microcontroller to the other and the results obtained are shown in the Table 11. It could be seen from the results that communication has taken place exactly as required considering the I²C network which is connected by using the heterogeneous interfaces.

6. Conclusion

I²C communication system is an effective protocol for networking of heterogeneous microcontroller based systems. Reasonable speeds of communication can be achieved using the I²C. A I²C network must be designed specific to a distributed embedded system considering the type of microcontroller that must be used for implementing the distributed embedded system. A specific architecture must also be determined for implementing a communication system that is suitable to a distributed embedded system. I²C protocol system does not support priority based message management system and therefore it became necessary to investigate a method and implement the same for a specific distributed embedded system.

The I²C protocol provides standard communication system. It does not prescribe any message flow system as required by a specific distributed embedded system. The messages must flow in a sequence as per a defined order. Addresses assigned to the device do not provide for any kind of preference for the device. The message flow system has been designed and implemented within a master designated microcontroller based system. Data must flow from one system to other as per the standard data structures as described by I²C communication protocol. The data structures that have been specifically designed for the application has been implemented in this paper.

Table 11. Experimental results

Tran ID	From			To			Whether Checksum error exists
	Micro-Controller System	Micro-Controller System Address	Number of bytes Sent	Micro-Controller System	Micro-Controller System Address	Number of bytes Received	
1	89C51	0111100	2	LPC2148	1000110	2	No
2	AT89S52	0110010	2	LPC2148	1000110	2	No
3	LPC2148	1000110	1	PIC18F4550	0101000	1	No
4	LPC2148	1000110	1	ATmega328	0011110	1	No

7. References

1. Swartwout M, Kitts C, Stang P, Lightsey EG. A standardized, distributed computing architecture results from three universities. 19th Annual AIAA/USU Conference on Small Satellite; 2005. p. 1–16.
2. Venkateswaran P, Mukherjee M, Sanyal A, Das S, Nandi R. Design and implementation of fpga based interface model for scale-free network using I²C bus protocol on quartus II 6.0. International Conference on Computers and Devices for Communication; 2009. p. 1–4.
3. Hu ZW. I²C protocol for reusability. 3rd International Symposium on Information Processing; 2010. p. 83–6.
4. Rahim BA, Rajan KS. Multi-protocol gateway for embedded systems. Int J Adv Eng Tech. 2011; 1(4):86–93.
5. Zhao F, Deng D, Wang Z, Liu H. Design of schematic mapping system based on I²C and usb bus. IEEE Conference Publication; 2011. p. 180–3.
6. Oberg J, Hu W, Irturk A, Tiwari M, Sherwood T, Kastner R. Information Flow Isolation in I²C and Usb. ACM. 2011; 254–9.
7. Bouterra Y, Chabir A, Mansour AB, Ghommam L. Development and implementation of a real time system for distributed control of laboratory robot. IEEE Conference Publication. 2014; 1–5.
8. Kommu A, Kanchi R. Designing a learning platform for the implementation of serial standards using arm microcontroller lpc2148. IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE); 2014. p. 1–6.
9. Mankar J, Darode C, Trivedi K, Kanoje M, Shahare P. Review of I²C protocol. International Journal of Research in Advent Technology. 2014; 2(1): 474–9.
10. Palmintier B, Kitts C, Stang P, Swartwout M. A distributed computing architecture for small satellite and multi-spacecraft missions. 16th Annual AIAA/USU Conference on Small Satellites; 2015. p. 1–11.