International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 5, October 2015, pp. 992~1002
ISSN: 2088-8708

Networking Heterogeneous Microcontroller based Systems through Universal Serial Bus

Sastry Kodanda Rama Jammalamadaka[*], Valluru Sai Kumar Reddy[*], Smt J Sasi Bhanu[**]
[*]Department of Electronics and Computer Science Engineering, KL University
[**] Department of computer Science and Engineering, KL University

Article Info

Article history:

Received May 13, 2015
Revised Jul 1, 2015
Accepted Jul 20, 2015

Keyword:

Distributed Embedded Systems
Heterogeneous ES systems
Networking ES Systems
Serial Communication
USB

ABSTRACT

Networking heterogeneous embedded systems is a challenge. Every distributed embedded systems requires that the network is designed specifically considering the heterogeneity that exits among different Microcontroller based systems that are used in developing a distributed embedded system. Communication architecture, which considers the addressing of the individual systems, arbitration, synchronisation, error detection and control etc. needs to be designed considering a specific application. The issue of configuring the slaves has to be addressed. It is also important that the messages, flow of the messages across the individual ES systems must be designed. Every distributed embedded system is different and needs to be dealt with separately.
This paper presents an approach that addresses various issues related to networking distributed embedded systems through use of universal serial bus communication protocol (USB). The approach has been applied to design a distributed embedded that monitors and controls temperatures within a Nuclear reactor system.

Corresponding Author:

Sastry KR Jammalamadaka,
Department of Electronics and Computer Engineering,
KL University,
Vaddeswaram, Guntur District, Andhra Pradesh, INDIA 522502.
Email: drsastry@kluniversity.in

1. INTRODUCTION

1.1 Background

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 electronic gadgets (which include digital cameras, washing machines…etc.) being used today are 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 specialised applications such as automobile systems require interconnecting individual embedded systemsfor controlling brakes, doors, mirrors, rare and front object indicators, engine temperature, wheel speed, tyre pressure, DVD control…etc. and to provide information into a display unit which is fitted into a dash board. The individual embedded systems are generally heterogeneous in nature as they are built around different technologies. Sometimes the networking has to be achieved through connecting the individual embedded systems that are placed in different layers, each layer catering for a specific communication speed.

Networking of individual embedded systems is generally achieved through Serial bus based communication around I^2C, CAN, USB, RS485 etc., communication protocols. Each type of networking

leads to different communication characteristics such as baud rate, length of communication, bus termination etc.

Networking of heterogeneous embedded systems using any of the serial communication protocol requires hardware conversion and the kind of conversion required will be based on the type of technology used in building the individual embedded systems. Every networked embedded system, thus must be individually designed and as such there is nothing like general approach recommended for networking a set of heterogeneous embedded systems. Many issues such as master and salve addressing, designing the messages that flow across, the sequence of flow of messages have to be considered for building distributed embedded systems. These issues differ in many ways from one distributed embedded system to other.A system of working / Approach is needed that help developing a distributed embedded system for a specific application considering the issues which include heterogeneity, message flow, node addressing, and message design.

1.2 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 different microcontroller based systems requires addressing various Hardware and software related heterogeneous issues 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^2C, ETHERNET, PCI, and ESA etc. Among all, bus based serial communication protocols are used for establishing a network connecting all the individual microcontroller systems. USB 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 USB based system is due to lack of native support within some of the microcontroller systems. The USB implementations within some of the microcontroller systems defer a lot, as USB 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 several versions of the same protocol is used which also should be addressed.

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

The messages must flow in a sequence for effecting a distributed application. The USB based system does not support prioritisation of the slaves to respond even though all the slaves are allotted with an address. There must be a way of prioritising the requests and responses as per the application requirements initiated from the master and responded by slaves. The data packets must also be designed considering the way the data is exchanged among the masters and the salves.

1.3 Related Work

Many contributions have been presented related tothe problem area; most of them concentrated around implementation of USB based serial communication method to be used as a standard which is generally achieved through converting from one interface to the other. Some work has been presented that explain the way the USB protocol can be converted into other protocols. Most of the contributions are in the area of point-to-point communication using USB.

Ana Luiza de Almeida Pereira Zuquimet. al., [1] used a converter to transform RS232C output/input to USB equivalent. The design of the converter is presented based on Engetraon UPS Serial communication requirements and they have shown the implementation on Cypress microcontroller based system.

Every home, these days is being automated by using various electronic gadgets forming into a digital network. Even the mobile phones are being used for automating and communicating purposes. Homes are being connected both through wired and wireless broadband. Many standards have been developed for

affecting digitization and communication but not many applied for real life applications. Standard protocols are to be developed that suits completely the issue of home digitization.Yong-Seok Kim et. al., [2] have developed home network using USB as a standard. It is possible to expand the home network when the same is developed through USB protocol.While the devices installed in the home are interfaced through USB, the mobile communication is achieved through CDMA and the local communication is achieved through Wi-Fi. They primarily focussed in protocol conversion from USB to CDMA and vice versa.

Universal Serial Bus (USB) is being used as default industry standard for processing input data that get generated continuously. The use of USB and USB HUB causes certain amount of latencies making it unsuitable for accessing the data which is generated at rapid speeds. It has been proved that USB can be used when the real time requirements are soft. USB protocol can be used in respect of the devices that can tolerate the delay in the order of milliseconds [3]. This paper has limited its discussion with reference to the latencies with which one can work using USB.

Many communication protocols are being used for affecting communication between the devices, systems and control equipment. Standard protocols being used include such as USB, RS232C, CAN, and ETHERNET. In most of the advanced systems, the protocols are being used inter- mixingly for the development of integrated circuits. However it has become evident that a real-time operating system and advanced microcontroller based systems are to be used when such many protocols are to be used within the same board.

Taghi Mohamadi et. al., [4] have presented that implementing, controlling and data acquisition functions through embedded systems will help in achieving overall reliability and durability. There is a need to determine hardware architecture and real time multi-tasking processes when multiple protocols are to be used. Such a system can be conveniently employed when network interfaces with different protocol layers are to be used. The architecture can also be employed for constructing a smart gateway or a router. However an embedded board which is versatile having all the stated interfaces is required for establishing such kind of hybrid network which uses different kinds of protocols. In this case a middleware is required using which communication with all standard protocols can be carried. Here again protocol conversion based on the real-time operating system has been only discussed.

Different sensors are normally connected to a microcontroller based system either through using I^2C/SPI direct interface or using analog signals converted to digital signals using an A to D converter. Microcontroller based systems can be connected to both a local host or to a remote host through an internet connection established through USB ports [5]. USB's are thus being used as Hubs, connecting devices on one side and hosts on the other leading to establishment of both local area and wide area networks. A networking architecture in this manner can be implemented, though Bus based networking is not used.A microcontroller based system can be designed for communicating with many of the devices by implementing different protocols within the same board. However this kind of topology is limited considering the extendibility requirements of distributed embedded systems; only fixed number of embedded boards can be connected in this way. The networking in this case is elaborated considering all those devices that have been supported with USB interface.

Connectivity between two different devices that have two different protocol interfaces cannot be achieved without the support of a USB-Wi-Fi bridge. Communication is affected through converting one protocol to other. Converters of this kind can be used for developing hybrid networks [6]. TusharSawant et. al., [7] have presented a microcontroller based system that has built-in USB ports and the same is interfaced with touch screen which has buttons through which commands can be fed for effecting data transfer between the mass storage devices using USB ports supported on the same board. This, in a way is called as USB to USB bridge device.

A. Ying Huang et. al., [8] have designed a microcontroller based USB Host system which can be used for interfacing different kinds of USB based devices. They have used an 8-bit MCUAT89C55 and an interface chip SL811HS for achieving the USB implementation within a single embedded system. The MCU is used to make it behave like a USB host. The driver/controller function is implemented within the MCU. Communication between a USB device and an USB implemented MCU is undertaken as per the standard of USB and the data flow is used to direct the transfer across the mass storage devices as per the format and structure. This a is a kind of point to point communication system than just simply visualising as a network of embedded systems.

Universal serial bus storage devices are used for faster I//O handling and generally treated as peripheral devices which needs a host for communicating with it. Communication between two USB devices thus needs to be achieved through a host. To avoid the use of PC for effecting communication between two USB devices there is a need to implement a MCU based USB communication system. Such kind of a host can be used as a network hub. The protocol needed to sense existence of USB device is implemented by the MCU based host. Harpreetet al., [9] have presented a microcontroller based system to transfer data between

the two USB devices. In this system VDIP2 module is used along with the microcontroller. VDIP2 consists of a chip called VNC1L which has built in USB ports, LCD, and a keypad. Commands are issued through a keyboard effecting data transfer between the mass storage devices that are connected to two of the USB ports of VDIP2. Users can see the data flowing across both the USB devices. As such, modifications are required to this proposition so that more USB devices are supported and communication between any two devices is achieved. This approach is more or like implementing a USB based hub.

Yassine Bouterraet. al., [10] have presented a distributed architecture for implementing an industrial robot using multiple MCU's which are connected through a I^2C bus and controlling initiated through a PC connected to one of the MCU's through a USB interface. Dave [11] has presented implementation details of USB based hub for conducting either high speed/low speed transactions. Dogan Ibrahim [12] has presented complete description of the networking of distributed embedded systems through USB protocols. The architecture and design of a USB hub and the USB protocol descriptions/specifications have been extensively presented.

In literature, the issue of establishing an USB based network connecting a heterogeneous microcontroller based systems has not been quite addressed. The communication architecture as such has to be designed separately considering a specific distributed embedded application. The design of specific communication systems involves allocation of specific addresses to the slaves such that communication takes place as per the priorities required by the distributed embedded application.The configuration of the slaves is also very much dependent on a specific distributed system. There is a need to design special descriptors using which the slaves are configured as per the requirements of the distributed system.

1.4 Solution

Thus, this paper addressed the design of USB based network for connecting heterogeneous Microcontroller based system, design of specific communication system as required by the distributed embedded application, address allocation to the salves and configuring the slaves through descriptors for making them adaptable for the implementation of distributed embedded application. The designing of the messages and controlling the flow of messages across the distributed Microcontroller based system has been presented considering a distributed embedded system that monitors and controls temperatures within a Nuclear reactor system.

2. METHODS

2.1 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 tubes must be sensed and it is also necessary to maintain proper gradients across various points at which the temperatures are measured. When temperatures raises 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 controls the process of pumping is achieved through various embedded systems. The operators must be alerted when the temperature gradients 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, temperaturegradients, status of triggering buzzer etc., is created at a PC(HOST) 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 implement the distributed embedded systems. One of the major issue that must be addressed is heterogeneity that exists among different types of Microcontroller based systems which are used for developing and implementing a distributed embedded system.These requirements leads to implementation of distributed embedded system having several microcontroller based 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

individual heterogeneous embedded systems meant for either sensing or actuating with built-in interfaces along with an individual embedded system that provides centralised coordination.

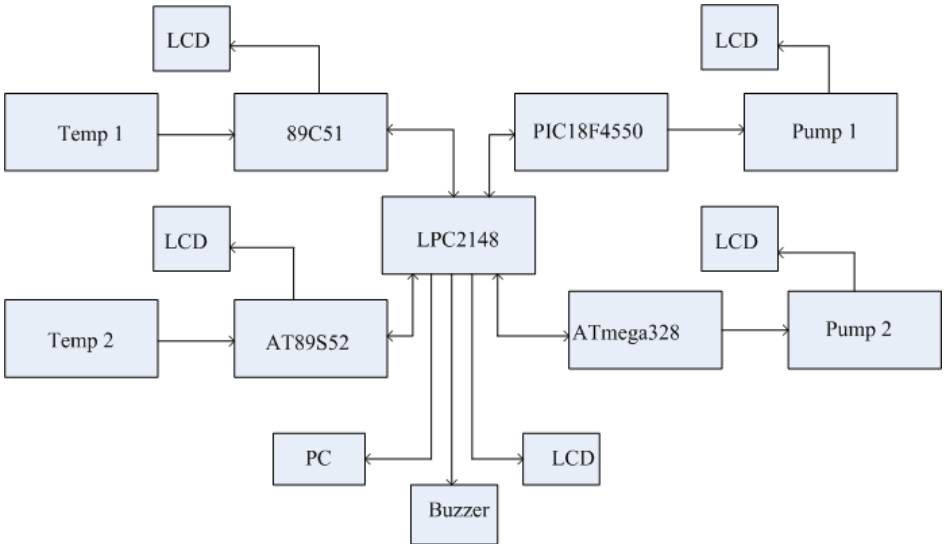

Figure 1. Top view of a Distributed embedded system

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

Table 1. Requirement specification of a distributed embedded application

Req. Num.	Requirement description
1.	Read Temp-1 and write to LCD
2.	EffectUSB 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
6.	Compare Temp-1 > temp-2 and if true assert buzzer on 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.	EffectUSB 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

Req. Num.	Requirement description
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

2.1. Designing USB Based Networking for Interconnecting Heterogeneous Individual Embedded Systems

USB based networking is one of the methods that exists today for achieving interconnection among different embedded systems. USB as such is being used universally and has become industry standard for effecting communication between the Computing stations and peripheral devices and now even being used for establishing the communication between many embedded systems.Many of the Microcontroller based systems have no native support for USB 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.

USB helps in establishing a network interconnecting many embedded systems. Every distributed embedded system is different and a dedicated network has to be designed and developed. The ES network design must address application specific requirements. The application specific requirements related to the Nuclear reactor application are shown in the Table 1. In the network, a specific embedded system must behave like a central microcontroller system which is typically situated at a remote location. As per the description of the functional requirements, the central microcontroller based system shallhave to act like a master and the rest as slaves. The communication between the master and the slave requires a speed of 40 Kbps which allows the signals to be driven to a distance of more than 1KM which is a sufficient requirement considering the application in view. If bus length has to be increased 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 USB as a backbone is shown in Figure 2. The USB based network contains a single master and 4 slaves. Conversion mechanisms are to be considered when the microcontroller based systems which are used as a part of network are heterogeneous due to the reason that they do not have native USB interface.

As per the functional requirements of an application, LPC2148 a 32 Bit Microcontrollers used as a master device for achieving communication between the slave devices and the master.It consists of native USB support. 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 USB support. For interconnecting these systems, conversion mechanisms are required. 89C51 and AT89S52have been designed with inbuilt RS232C communication interfaces, A device FT232R has been used for converting RS232C signals to USB and Vice Versa. The devices implements buffering techniques for converting a 19.2Kbps speed which is the maximum speed achievable through a RS232C serial communication system into to 1.5Mbps speed which is a low speed support on the USB side. This conversion is good enough as the amount of data to be transmitted from the salve side is not more that 18K bytes considering that the 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 USB network considering the interfacing issues is shown in the Figure 2.

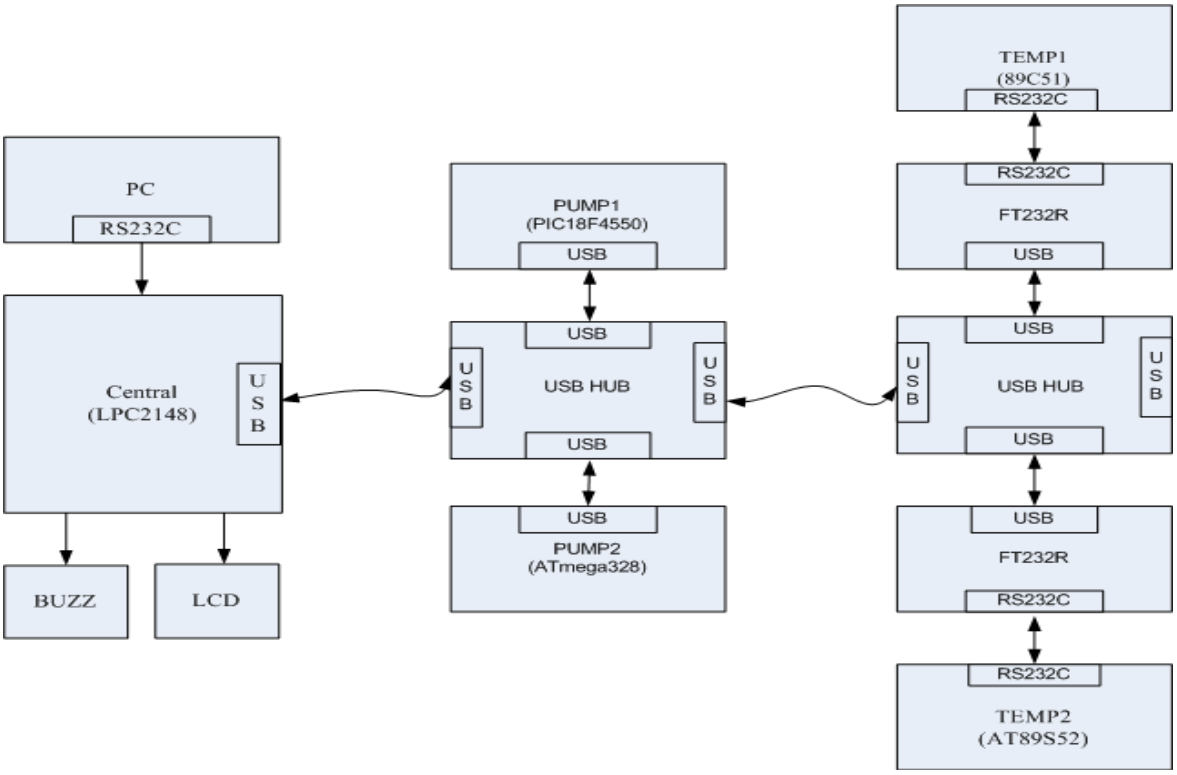

Figure 2. USB based Networking for Nuclear reactor system

2.3 Designing Communication System

The networking diagram shown in the Figure 2 shows the interfacing of the various heterogeneous microcontrollers based systems which are interconnected through a USB 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 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 USB 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 USB 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.

Two communication components that can communicate using USB with the master are provided within the applications resident on PIC18F4550 and ATmega328 that controls the running of the pumps for regulating the flow of coolants into the reactor tubes. The software architecture that depicts the application specific communication is shown in the Figure 3.

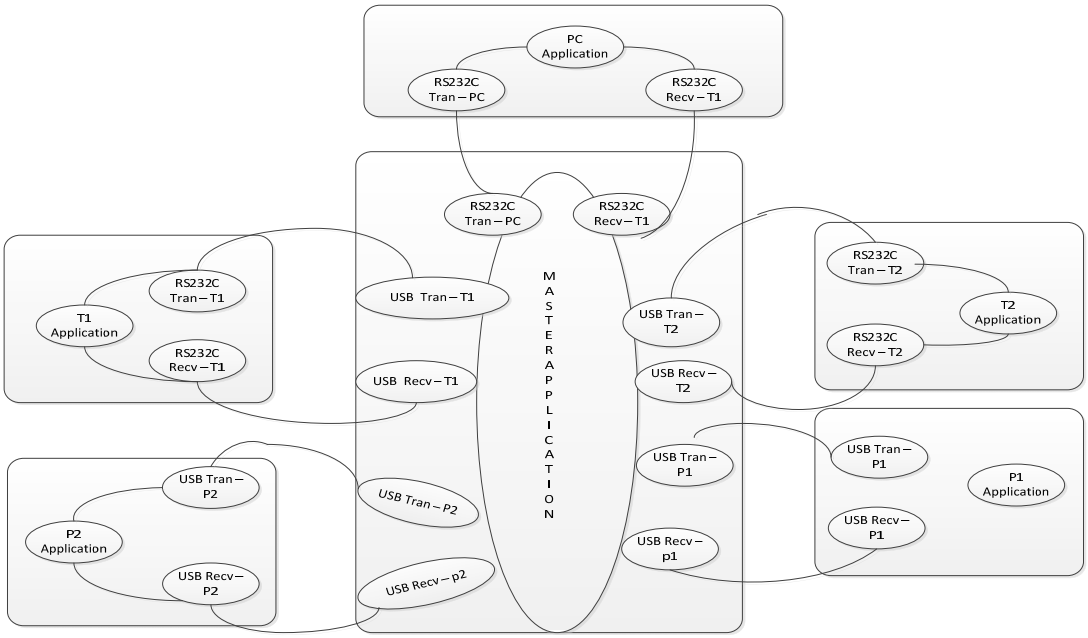

Figure 3. System Architecture for effecting communication among distributed embedded systems

2.4 ANovel Address Allocation Algorithm

In USB based communication, one of the connected devices will be the master and the others as slaves. Every communication is initiated from the master. Every slave is assigned with an address by the master at enumeration stage at the time when any device is interfaced with the bus.When a device is plugged into a USB bus, it becomes known to the host through a process called Enumeration. After the process of enumeration, the host sends a reset signal to the device through address0 for placing the device in a known state. The device will send its details to the host through address0. The host assigns a unique address to the device and sends a reset address request to the device. After the request is completed, the device assumes the new address.

In the case of USB communication only one slave device will respond when requested by the master. The response from the salve could be an acknowledgment followed by the actual data requested by the master through a data packet which contains the details of the data the master is expecting. The addresses allotted to the devices by the master could be random and the address as such does not dictate the priority of the salves to respond. However the master should have a mechanism using which it can prioritise the requests to the salves as per the message flow required by the distributed embedded system.The application running on the master can be dynamically fed the addresses and the sequence in which the messages should flow using the addresses that were allocated to the device by using an application on the PC which is interfaced with the master device. The addresses to the slave devices can be allocated as per the priority of the message flow. A typical address allocation scheme that can be initiated from PC is shown in the Table 2.

Table 2. Address allocation algorithm

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

However the messages from the slaves can be of different patterns and the same are to be handled as per the priorities attached to those messages. The communication software running on the master, will post a message along with its priority to a queue and a queue handler will despatch the messages as per the priorities attached to the messages. The working of the priority based dispatching system for effecting the flow of control of messages as required by the distributed embedded application is shown in the Figure 4.

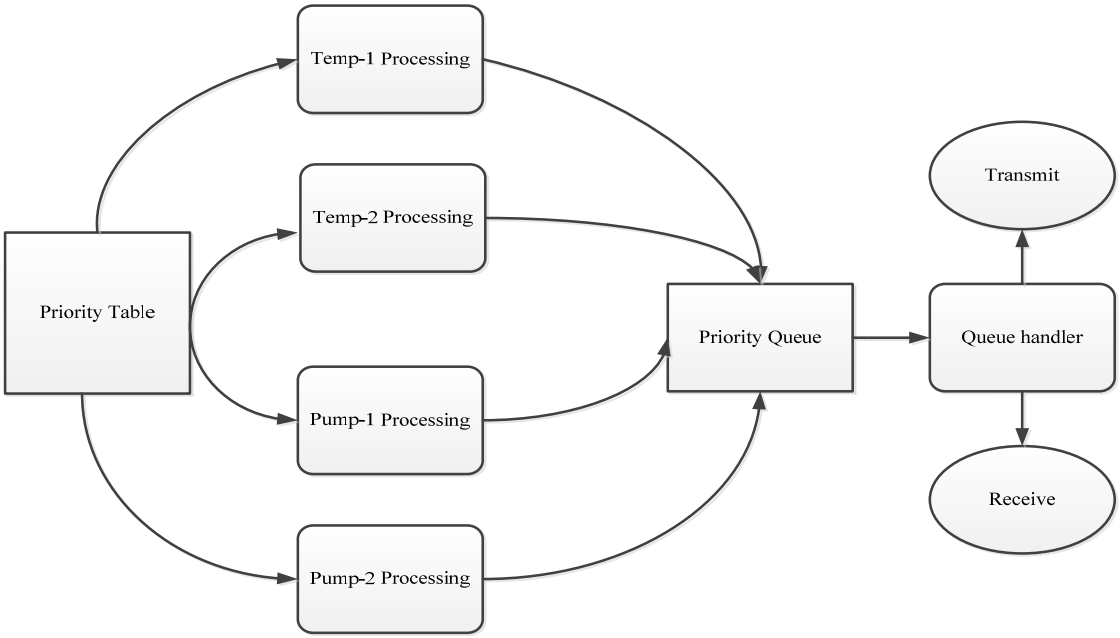

Figure 4. Priority based message dispatching method

2.5. Designing Descriptors for Configuring the Slaves

The salve based systems that are interfaced through USB can be configured through different types of descriptors. Different details of the salves can be made available to the master through descriptors. The descriptors describes manufacturer ID, the version of the device, the version of USB it supports, what the device is, its power requirements and the number of endpointsetc. The most commonly used USB descriptors include Device descriptor, Configuration descriptor, Interface descriptor and Endpoint descriptor

Device descriptor represents the entire device. It provides the general information such as manufacturer ID, serial number, product number, the class of the device and the number of configurations. Configuration descriptor provides the information about the power requirements of the device and how many different interfaces it supports. There may be more than one configuration for a device. The interface descriptor specifies the class of the interface and the number of endpoints it uses. There may be more than one interface.The Endpoint descriptor specifies the transfer type, direction, polling interval, and maximum packet size for each endpoint. Endpoint0 is the default endpoint, is always assumed to be a control endpoint and never has a descriptor.

In addition to above mentioned descriptors the USB protocol support the inclusion of more application specific descriptors. A new descriptor is designed for making available the priority of the device to the salve so that the same can be stored within it which can be used by the slave to check whether the required message flow is being affected and report the irregular sequence to the master when such event happens.

2.6. Designing Data Packets

In USB, the data is transferred in the form of packets. Normally it consists of three packets.

1. Token packet is the header defining the transaction type and direction, the device address, and the endpoint.
2. Data is transferred in a Data packet.
3. The status of the transaction is sent by the acknowledgement through Handshake packet.

In a transaction, data is transferred either from the USB host to an USB Device or vice-versa. The transfer direction is specified in the token packet that is sent from the USB Host. Then, the source sends a

data packet or indicates it has no data to transfer. In general, the destination responds with a handshake packet indicating whether the transfer was successful.Packets could be thought of as the smallest element of data transmission. Each packet transmits an integral number of bytes at the current transmission rate. Packets start with a synchronization pattern, followed by the data bytes of the packet, and concluded with an End-of-Packet (EOP) signal. All USB packet patterns are transmitted least significant bit first. Before and after the packet, the bus is in idle state.
The data packet design of temp1 processing system is shown in the Table 3.

Table 3. Data packet design of temp1 processing system

Token Packet	**Sync**	**IN**	**ADDR**	**ENDP**	**CRC**
	0000 0001	0110 1001	0111100	0	5 bits
Data Packet	**Sync**	**Data0**	**Data**	**CRC**	
	0000 0001	1100 0011	2 bytes	16 bits	
Handshake Packet	**Sync**	**ACK**			
	0000 0001	1101 0010			

The data packet design of temp2 processing system is shown in the Table 4.

Table 4. Datagram design of temp2 processing system

Token Packet	**Sync**	**IN**	**ADDR**	**ENDP**	**CRC**
	0000 0001	0110 1001	0110010	0	5 bits
Data Packet	**Sync**	**Data0**	**Data**	**CRC**	
	0000 0001	1100 0011	2 bytes	16 bits	
Handshake Packet	**Sync**	**ACK**			
	0000 0001	1101 0010			

The data packet design of pump1 processing system is shown in the Table 5.

Table 5. Data packet design of pump1 processing system

Token Packet	**Sync**	**OUT**	**ADDR**	**ENDP**	**CRC**
	0000 0001	1110 0001	0101000	0	5 bits
Data Packet	**Sync**	**Data0**	**Data**	**CRC**	
	0000 0001	1100 0011	2 bytes	16 bits	
Handshake Packet	**Sync**	**ACK**			
	0000 0001	1101 0010			

The data packet design of pump2 processing system is shown in the Table 6.

Table 6. data packet design of pump2 processing system

Token Packet	**Sync**	**OUT**	**ADDR**	**ENDP**	**CRC**
	0000 0001	1110 0001	0011110	0	5 bits
Data Packet	**Sync**	**Data0**	**Data**	**CRC**	
	0000 0001	1100 0011	2 bytes	16 bits	
Handshake Packet	**Sync**	**ACK**			
	00 1	1101 0			

3. RESULTS AND DISCUSSION

Experiments have been conducted using the USB network designed and the distributed embedded application system & the communication system implemented. Communication is effected by making the

data flow as per the communication system architecture. The results of the experiment conducted are shown in the Table 7. The experimental results have also been validated by using PROTEUS simulator.

Table 7. Experimental results

Transaction ID	From			To			
	Microcontroller system	Microcontroller system Address	Number of bytes sent	Microcontroller system	Microcontroller system Address	Number of bytes Received	Whether Checksum error exists
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

4. CONCLUSIONS

USB communication system is an effective protocol for networking of heterogeneous microcontroller based systems. Reasonable speeds of communication can be achieved using the USB.A USB network must be designed specific to a distributed embedded system considering the type of microcontrollers 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 Application. USB protocol system does not support priority based message management system and therefore it became necessary to investigate a method and implement the same which is suitable for a particular distributed embedded application. The USB protocol system provides for an enumeration stage at which time the information related to the devices is sent to a master for its complete understanding of the device. A new descriptor has been added using which the priority of messaging is fed to the slave and to make the slave store the same within it and use the same for checking the correctness of the message flow so that slave can inform the master if wrong message flow for any reason has been initiated**.**

REFERENCES

[1] Ana Luiza de Almeida Pereira Zuquim Claudionor Jos C Nunes Coelho Jr. Antanio Ot6vioernandes. *An Embedded Converter from RS232 to Universal Serial Bus.* IEEE Conference Publications. 2001: 91-96.

[2] Yong-Seok Kim, Hee-Sun Kim, Chang-Goo Lee. *The Development of USB Home Control Network System.* IEEE Conference Publications. 2004; 289-293.

[3] Lalitha Ramadoss and John Y. Hung. *A Study on Universal Serial Bus Latency in a Real-Time Control System*. IEEE Conference Publications. 2008; 67-72.

[4] Taghi. Mohamadi. *Designing an Embedded System for Interfacing with Networks based on ARM*. IEEE Conference Publications. 2011; 407-410.

[5] J. Ducloux, P. Petrashin, W. Lancioni, L. Toledo. *An Embedded USB Dual-Role System Integrated for Mobile Devices*. IEEE Conference Publications. 2012; 66-72.

[6] Disha Juriasinghani, Tanay Krishna Dev. Embedded System for USB WiFi Bridge. *International Journal of Engineering Research and Applications (IJERA)*. 2013; 2(1).

[7] Tushar Sawant, Prof. Sanjay Deshmukh, Shilen Jhaveri, Siddarth Bhatt. Implementation of USB to USB Bridge for Computer Independent Data Transfer. *IJCET*. 2013; 4(2): 300-308.

[8] **A**. Ying Huang, Xiaoyong Fang. Study and Implement of an Embedded System Based on USB Host. *International Journal of Advanced Computer Technology (IJACT)*. 2013; 2(5): 14-20.

[9] Singh Harpeet, Kaur Kamaldeep. Flash Drive Communication Using Embedded System. *International Journal of Engineering and Computer Science*. 2014; 3(2): 3947-3950.

[10] Yassine Bouterra, AlaaChabir, Asma Ben Mansour. *Development and Implementation of a real time system for distributed control of laboratory robot*. IEEE Conference Publications. 2014; 1-5.

[11] D. Anderson, D. Dzatko. Universal Serial Bus System Architecture. Text Book.

[12] Dogan Ibrahim. Advanced PIC Microcontroller projects in C. Text Book. 409-461.