B.Tech Project

Development of Sensor Network Testbed

By

Udayan Kumar

200101227

Dhirubhai Ambani Institute of Information and

Communication Technology

Gandhinagar, GUJARAT

April 23, 2005

Dhirubhai Ambani Institute of Information and

Communication Technology

Gandhinagar, GUJARAT

CERTIFICATE

This is to certify that the Project Report titled *"Development of Sensor Network Testbed"* submitted by Udayan Kumar ID 200101227 for the partial fulfillment of the requirements of B.Tech (ICT) degree of the institute embodies the work done by him on campus under my supervision.

Date: _____ Signature: _____

(Prof. Prabhat Ranjan)

Abstract

The goal of this report is to provide information about CENSE (A Century of Sensor Nodes) testbed, which we have designed, to facilitate faster deployment of Sensor Networks. CENSE would provide testing facility for 100 nodes. We have tried to give CENSE a modular design, which would allow greater flexibility in choosing the components for different modules of a sensor node. The information is compilation of my experience and knowledge, so that it can guide others who want to design such a system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To motivate the objective of my thesis I would like to present a sightly modified scenario created by Jeremy Elson and Deborah Estrin [6]. Imagine a Thursday morning in the year 2005. A great earthquake hits Bhuj again. Shear waves of magnitude greater that 8.0 on Richter scale start spreading towards Gandhinagar. Since Bhuj was known for its seismic activity, scientist monitor this zone by heavily instrumenting it. The technological advancements have made it possible to setup a pervasive sensory system without requirement of pervasive support system.

Desert is made home to millions of tiny sensors. City too have a sensory skin for its protection. Each of the sensors have a small processor, memory, sensors and radio that can be used to communicate and cooperate. A few dozen variety of sensor nodes are deployed. Atmospheric, chemical, organic and seismic sensors are most common. Each node has its own special observation skill.

Some active sensors close to the epicenter detect unusual earth movement. Since individual sensors could not be trusted. A alert (most of the nodes remain switched off to conserve power) is sent in the network which put all the nodes in active mode. For first few milliseconds the information was a little sketchy but with in a second, network reached to the consensus that it is an earthquake of a frightening magnitude. Nodes with highest power and longest range transmit the information to the nearest wired communication access point nearly 50-60 Km away.

Alarm spread to all the cities in the area. Smart building, bridges, underground pipelines, freeways and even some private houses received the detailsalmost 30 seconds before the quake. Buildings change their resonant frequency to have least damage. Sirens blare at the crossings. The city seems to be waitingfor the quake. Finally the earth rolls and shakes violently. Most of the buildings survive with occupants unharmed.

Some old buildings collapse. Rescue team releases survivor locating self propelled sensors in the rubble. These sensors penetrate deep into the structure and using radar and acoustic signals map the whole structure and then locate the survivors using thermal imaging techniques.

Meanwhile the sensors in the city's water system detect abnormal increase in the toxicity. Areas with newer pipe installations had sensors to detect leaks. They send a command to the nearest valve to shut. In one of the areas shutdown didnot take place. So a message is send to warn the resident and to dispatch a crew to the site. From Monday life goes on normally [6].

This idea may look improbable today but a billion computers would have been as fancy full in 1955, when IBM declared the total number of computers required world over would remain less than 10.

Detection of world's physical parameters makes sensors most suitable technology for disaster management and sensing. Sensors, though are not just limited to environment sensing. Any application involving sensing of physical parameters like sound, light, pressure, temperature, acceleration, magnetic field etc, may use sensor network. They show a very close relationship with the physical world which is in contrast with traditional computing which often exists in virtual world. Since by design these sensor nodes are independent and self containted they do not have any infrastructure requirement. With the support of AD-HOC networking they can be deployed in any environment.

As my BTech project, I have tried to develop a testing platform for the Sensor Network - CENSE. This work was done in collaboration with Ashish Ranjan, Priyank Mundra and Vishal Jalan. We have put in

our efforts to create a platform where faster testing of Sensor Network applications could be done so as to reduce the time and cost involved in developing sensor applications.

This document explains the design challenges we faced in designing CENSE system. Beginning from background where a introduction is given to the different scenarios in which sensor networks work, we move on to the design consideration and then finally explain how we implemented CENSE.

Chapter 2

Background

2.1 General

Sensor Networks are very versatile. Almost all the time they use Ad-Hoc networks which are distributed and their sensors continuously gather information from the environment. The sensor network nodes communicate with each other, process data received from other nodes, aggregate it and then send it again to other nodes. Here we have an intelligent component which with the collaboration of other nodes in the network provides higher level of interpretation of spatial data and not just raw data which has to be interpreted at some other location. For example imagine a temperature sensor installed in a forest to monitor forest fires. It records the temperature continuously. And if we examine how temperature is changing with time we may draw some conclusions. During very sharp increase of temperature, can we safely conclude that fire has spread in the forest? Or some animal is very close to the sensor, shooting up the temperature? Can we surely say that it was a false alarm, we don't need to worry? Now suppose we also add a smoke sensor in the forest. If the sensor detects smoke, can we assume that its fire in the forest? This would provide more real scenario. Now can we add some more sensors in the forest. The sensor set may include temperature and smoke detector. Suppose now from a particular region of the forest many sensors report high temperature anddetection of smoke. We are more certain of the fire along with its location and magnitude. This example illustrates how multiple nodes and sensor data can aid in giving high level understanding of the activity, the inclusion of spatial data and multiple sensors can lead to a fairly accurate conclusion.

Now we will examine how processing is done.In order to interpret spatialsensor data, the data must to be collected and processed in an organized fashion. One method is to gather all the data and transfer it out of the network where the data can be analyzed. However, a connection to the network may be only accessible temporally or may have limited throughput, making it difficult to continuously monitor and analyze large amounts of sensor data. The other possibility lies in doing the analysis of the data within the network itself. As opposed to doing the processing on a single computer, the processing could be distributed among all the nodes, each of which acts as a minicomputer. After the completion of the analysis, the results could then be transferred out of the network. Distributed processing is a research area in itself. Depending on the communication constraints of the system, algorithms must be developed that would allow individual nodes to share and process data efficiently.

2.2 Capabilities

Looking at the functionality provided by the sensor nodes one can realize that capabilities required in each node. I have tried to list down the various capabilities required by the Node.

1. **Processing** : A sensor node is expected to communicate and process sensor data, therefore it should have some computing power. Though the computational power of the node may vary from an 8 bit micro controller to a full blown 32 bit microprocessor.

2. **Sensor** : The interface between the environment and the node is the sensor.The node should have basic

sensors suitable for the environment and the requirement of sensing, which may include temperature, pressure, humidity, light, sound, acceleration, magneticfields.

3. **Communication** : Communication must be possible in standard outside environments. Node must have the ability to communicate with one another at ranges from a few meters to kilometers.
4. **Power** : In order to monitor an environment for any length of time, the node must have enough energy to survive anywhere from a few hours to months at a time.
5. **Localization** : This capability is not very essential but may be required in scenarios where exact location of a node is required.
6. **Mobility** : For node whose position remains fixed may not require the mobility, but environments which are hazardous to humans, the nodes may themselves go there and get the information. In these scenarios adding mobility to the nodes would be a desired feature.

2.3 Related Work

2.3.1 UCLA

University of California at Los Angeles can be considered as the pioneers in the research field of Sensor Networks. They had started a project called WINS (Wireless Integrated Network Sensors)[20] in 1993. By 1996 they had deployed the first Sensor Network. Afterwards they have developed, Em* [7] a software tool for programming and simulating sensor network. They have also attempted to used 32bit microprocessors in sensor nodes.

2.3.2 UCB

University of California at Berkeley has been very active in the area of Sensor network. They made their first node in 1999 named it WeC. WeC provided temperature and light data to over 15 feet using radio [14]. Then Rene Node was developed, which had more data and memory space. Later Dot node, Mica node and Spec nodes have also been developed. Spec node is 2.5×2.5mm of size, with capabilities of communication, sensing and computation. They have been successful in producing a very popular Operating System for these nodes TinyOS [8].

2.3.3 Work in India

Sensor network is a new area research in India. Most of the institute have been working on the algorithms and simulation of sensor nodes, very few have gone to the hardware level.

IITD has been working on Sensor network since long. They have spawned a company "Elfsys"[5], which provides complete solution for creating sensor network applications. They also developed some softwares for collecting and interpreting the data.

The Sensor Network research group[9] at IITB have set up a 9 node test bench for recording temperature data. The temperatures are displayed on the internet. Exact details are not clear, since I could get data only from their web site which is not providing enough information about their implementation. They have established a multihop network, collecting temperature from all the sensors. As far as I could get no in-node processing being done.

2.4 Need of a test bed

The applications of Sensor Network Vary from Habitat monitoring(GDI) [11] to weather monitoring to military espionage [12]. New applications are coming up every day. This rate faster than the development of the sensor network for that application. Direction of our effort is to minimize the time for the development

of sensor network nodes by providing necessary hardware architecture for faster testing of sensor nodes. Currently we are concentrating at the hardware level.

Sensor network node's features are mostly and heavily depended on the requirement of the application In most of the cases a design for one application would not work for other application. Almost allapplication require a custom nodes. For example some application may require heavy processing capabilities on the nodes (Image processing application) and another may require the nodes to have a very long lifetime.Yet another may not require existing temperature sensor since it needs more precision, so the currentsensor needs to be changed. Likewise communication needs also depend on the appplication. It can be observed that most of the changes needed in the node, to meet the demands of an application require modification or customization in Computational unit,Sensors, power or communications.

So if we can develop a platform or Testbed where we could try out various configuration of a node, the final node would not only have required features, also the development would faster and of course after checking everything from software to hardware, we would be reducing the chances of failures of the node and network. Testbed is actually the device broken into independent parts. If you change one part of it the other remains unaffected. The obvious advantage is that it allows for design testing, it is flexible and debugging become very easy. But these are actually not the direct advantages which are the motivation for going for the modular test bed. By building the modular test bed we can save on the total development cost which is essentially one of the major constraints in the deployment of wide scale sensor networks.

Chapter 3

Design

3.1 Constraints

We would like to design a test bed which is flexible, cost efficient, easy to use, power efficient, provides excellent debugging facilities, covers all the requirements of sensors. For designing such a system we would have to take an assumption that all the components we require are easily available,there is no constraint on time required for designing such a system, so one can consider all the possible scenarios and conditions. This kind of design may lead to very optimized and close to ideal design of a Testbed. However the system we have designed was subjected to time constraint (only four months), resource constraint (since most of the components have to be acquired from abroad) and level of experience (people were not very familiar with the kind of work we have done, so lot of work went into learning simple things).

In spite of all these constraints we have made our sincere effort to keep our design flexible, such that if someone gets the required components in future, he may be able to make use of our design. We have also flouted the efficiency constraints in terms of cost and power. As our main aim is to prove that our design can be used for testing sensor network specific hardware in form of nodes.

3.2 Drivers

We wanted to complete our design and implement the design in hardware. So we fully focused our design which would be implementable, though we did not wanted to loose our flexibility. We looked at the components which are easily available in the local markets and can be used for the implementation. We proposed two designs (refer below), later however we chose one over the other on the grounds that components for one design were available locally and were cheap. Moreover it gave us more flexibility in designing the modules.

From the above discusion (Sec.2.2), one can infer that a sensor node is having 4 major modules in terms of functionality. Every application requires changes in one or more modules. Our design of CENSE should be such that the modules on a test bench are independent of each other or have least interdependence. This would allow us to change any of the module without affecting the other. Design of such an interface is the main challenge. To appreciate this point I will raise an example. Suppose we have a node on the test bed for image processing application. Now we want to make the node for weather sensing. The only parts that should be changed are the sensors, instead of image sensors we need temperature, humidity, pressure etc sensors. If rest of the requirements on life time and communication remain same then we can just replace the sensor module and load the required software on the microcontroller. Thus we need to create a super set of rules and device, which would allow interconnection between the devices and yet maintain the independence of the modules..

Constraints	Value	Assumption
Wires or lines	40	leaving some extra line for future
Max Data rate	1Mbps	SPI has 1Mbps data rate and is the fastest bus
Max Current	100mA	worst case radio requirements are 300mW. Assume. Voltage =5V. I=P/V (Jason Hill)
Power Supply	5V	most of the devices are compatible with 5v

Table 3.1: Constraints which decide the quality of line required

3.3 Solution

One of the question that comes to mind, when we desire almost zero interdependence between modules is that how will we interconnect these modules. They may be operating on different voltage level, may use different output formats or they may have totally incompatible connectors. For example suppose our power supply is providing output voltage at 5V and all our current modules are compatible with it. Now we want to change the processor module with a different processor module which requires 1.8v of power supply. Our design is such that we want to keep the modules independent of the changes in other module. So the processor module will not be able to connect, since the power supply is not compatible. A solution would be to connect a voltage converter which would convert 5v to 1.8v on the processor module. This would allow it to connect to the same interface. Same type of problem may occur if Power supply is changed from 5V to 3V (say). As we can relate from the example that some constraints or rules would come in the system during the design of such a common interface. We have tried to make the constraints such that they don't become a hurdle in testing of node for most of the applications. Some constraints that have to tackled are listed here. We need to fix before hand how many connecting lines would be used for a particular task, what kind of signals would be used, what would be maximum data transfer speed, what would be the maximum current capacity, what would be the power supply voltage and how many lines are needed for the overall system. As this informations would allow us to make the interface without missing any required features. Before making any decision on any of the constraints, a literature survey was done to collect information from various vendors about microprocessor, sensors, powersupply and commincation devices. We found that a large set of microprocessor provide very similar interfaces like digital I/O, ADC, support for USART, I^2C,SPI and JTAG port. Next we found that almost all the sensors can be interfaced using these features. Same was the case with the communicating devices. It was also found that majority of the microprocessor and devices were compatible with TTL levels i.e. 5V or 3V. The data thus obtained pointed that a lot of

Type of lines	Number required
Power	2
JTAG	4
I^2C	2
SPI	4
USART	3
ADC	8
Interrupt	1
Total	24

Table 3.2: Number of lines required

devices are compatible with each other in terms of power supply, output and output signal, which helped us in selecting the power supply output, total number of lines required, different types of digital buses required, number of analog lines required, maximum current that may flow in the line, max data rate to be supported by the interface. (Tab. 3.1)

In the above discussion we have discussed bottom-up approach to get a solution. We have also tried to define

our requirements by analyzing the requirements of various application. For example most of the application would not require more that 8 analog sensors. Most of the communication would not require very large distance communication, so radios would not require much power. Example of application we refereed GDI habitat monitoring [11] , weather monitoring, countersniper system [12], object tracking [18].

Chapter 4

Implemention

4.1 Generic Interface or Bus

One of the basic requirement of a modular design would be an interface which allows connection between modules. In an electrical device such interface would be provided by wires. For the sake of maintaining uniformity and independence across the modules, we allocated some wires on the bus for a particular purpose and a condition was imposed on the modules to strictly use wire according to the purpose they have been allocated(Fig.4.4). This kind of allocation and grouping would increase the workload as one would have to make the module such that it complies with this order. But this process made the modules independent of others. The interface would provide some wire which could be used by multiple modules and wires which would connect only specific modules to other modules for example analog signal lines.

In our system the wires need to carry two kinds of signals analog and digital. Analog signals would be shared between two modules only. They would be mostly used by analog sensors on the sensor modules and may be by the power module to measure the power level.

However most of the time we would be using digital data lines, since digital sensors and other devices are as easily available as they analog counterparts. The advantage is that many devices can be connected to a single bus (unlike analog line where one dedicated line is required for each analog device). One more reason of having lesser number of analog line is that we can use Analog to Digital convertors on sensor board itself and then provide digital output.

Digital lines are nothing but some standard buses available for transfer of digital data. Many a times microprocessor supports few of the popular buses in hardware. if the desired one is not available we can make the protocol stack on software. Some of the commonly used buses are SPI, I^2C (TWI) and CAN.

4.2 CENSE

One of the ways in which actual hardware can be built is by taking a Flexible Ribbon Cable (FRC) and connect the modules using the FRC connectors. Modules are made on a PCB (Printed Circuit Board) which can be hooked to FRC (Fig.4.2).A second solution is to make plug and socket kind of a connection. Each module would have a plug and a socket on its PCB, using which the modules can be interconnected (Fig.4.2). Both the solutions can serve our purpose equally well. But the easy availability of FRC made it our preferred choice.

Both solutions may suffer from the electrical signal interference due to the high voltage or high frequency signal on the neighboring line. To measure the kind of interference present, we conducted a small test using a digital oscilloscope, FRC, frequency generator. Analog signal was produced using function generator. Digital signal was produced using microprocessor, which was continuously switching output between 0 and 5 volts. Length of the FRC was 6 inches. We assumed that signal voltage would never increase above 5V, the maximum bandwidth would be 1Mbps since the fastest bus is SPI which works at this speed.(Tab.4.1)

The maximum interference found on the neighboring lines was due to an analog signal line carrying 5V peak

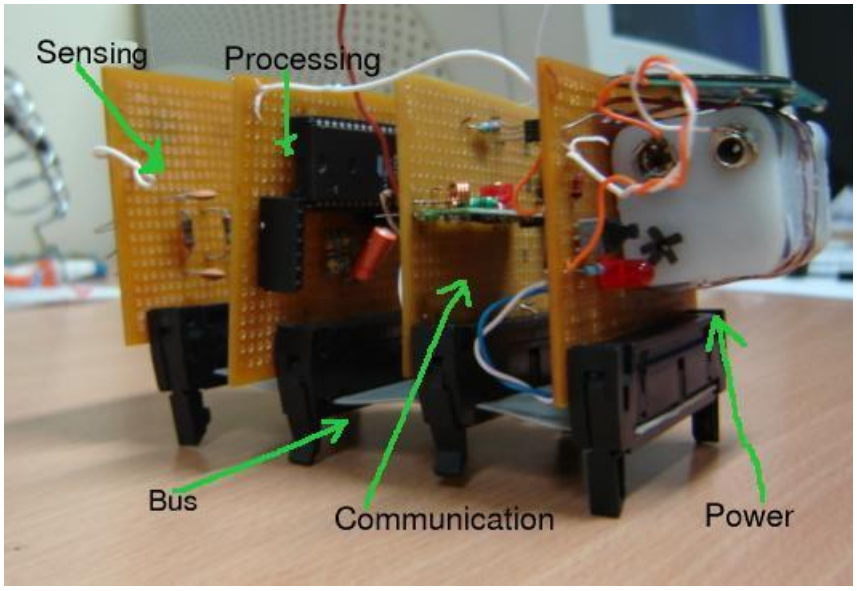

Figure 4.1: A node On CENSE

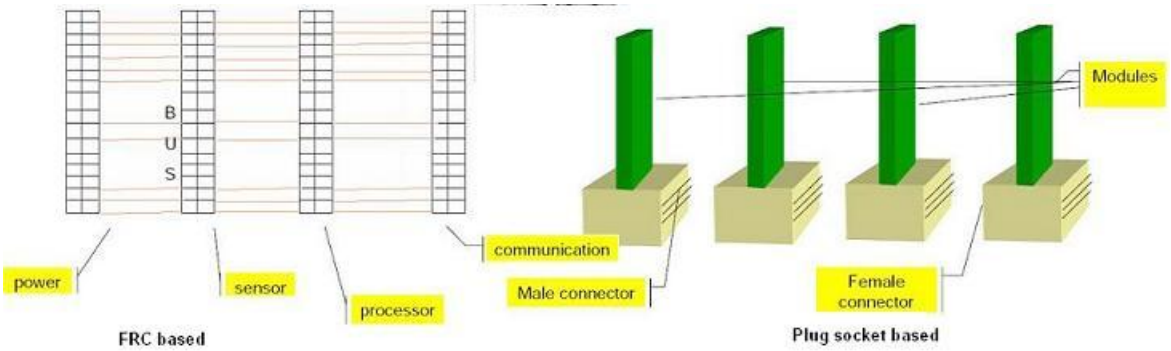

Figure 4.2: Possible solutions For the Testbed

Method	Signal Type	Pk-pk(V)	frequency	Effect on Neighbor
One line carrying digital signal, effect on neighboring	Square	5	1MHz	640mV square wave
One Analog Signal effect on neighbor	Sinusoidal	5	2KHz	same shape 372mV pk-pk
	Triangular	5	2KHz	same shape 324mV pk-pk
	Sinusoidal	5	200KHz	same shape 496mV pk-pk
	Triangular	5	200KHz	same shape 424mV pk-pk
	Sinusoidal	5	2MHz	same shape 496mV pk-pk
	Triangular	5	2MHz	same shape 424mV pk-pk
Effect on one digital line between two analog line.	Sinusoidal	5	20KHz	none
	Sinusoidal	5	200KHz	sine wave super impose on square waveform
	Sinusoidal	5	2MHz	freq. of sine wave increase on square waveform
Insertion of GND lines	Sinusoidal	5	2MHz	sine wave amplitude decreases
Effect on one analog line between two digital line	Sinusoidal	5	20KHz	none
	Sinusoidal	5	200KHz	sine wave superimpose on square waveform
	Sinusoidal	5	2MHz	freq. of sine wave increases on square waveform

Table 4.1: Interference pattern in FRC

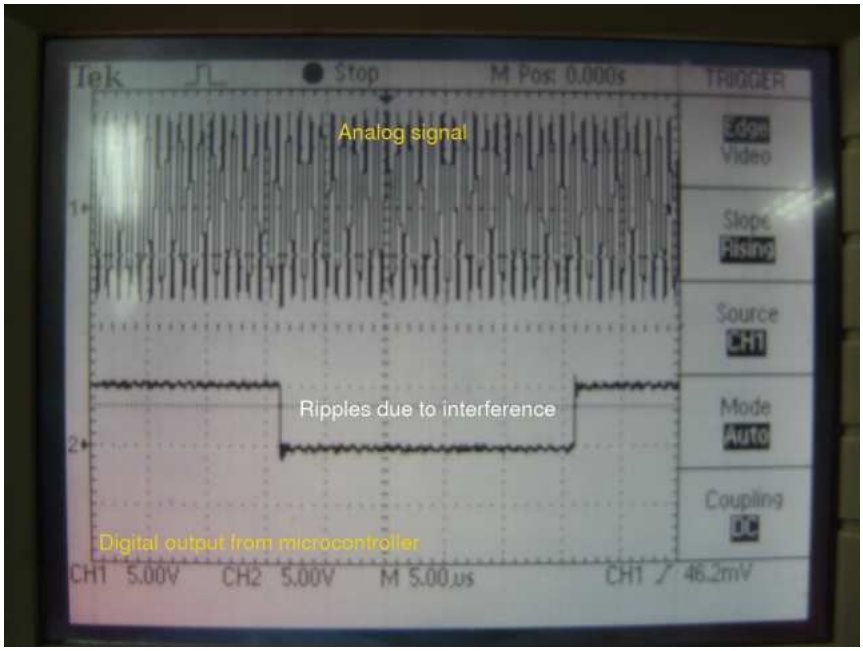

Figure 4.3: Interference pattern found when a digital line was next to analog line

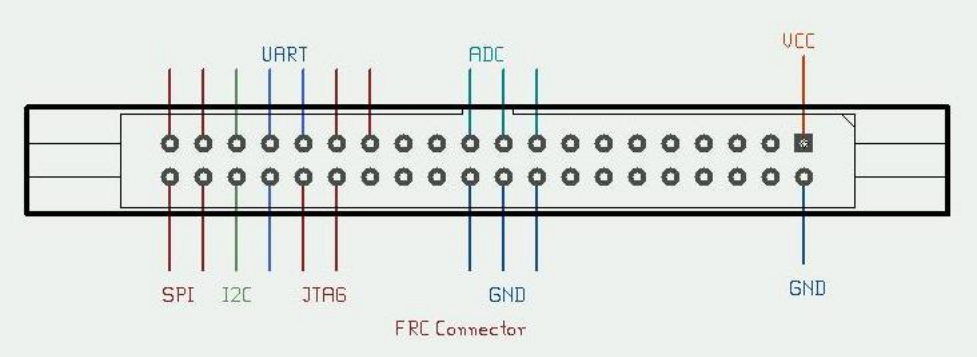

Figure 4.4: Bus settings as used in CENSE

to peak sinusoidal of the value 495mV ($\sim$ 0.5V). This kind of interference has little affect on the digital lines (digital high 5 - 3.5V and digital low 0 - 2.5V). However this may have a great impact on the analog signals. So we placed alternate ground lines between the analog signals to reduce the interference as they were found to reduce the interference considerably.(Fig. 4.4)

We have currently taken 40 line FRC, even though we require on 24 lines (Tab.3.1) because we require 7 lines for grounding lines between Analog line. And considering that in future requirements may change,so we have left 9 lines for future (looking at the Zigbee Chip interfacing we found that programing of that chip would require some extra lines[17]). The FRC has four FRC female connectors on it, which allow the modules to interconnect.

4.3 Selection of Module Component

4.3.1 Processor

The microprocessor acts as the overall controller of the node. It schedules all the functions of the node. Therefore we need to have a processor suitable for our need. Sensor node do not require very high computational power, it does fairly well even with 8 bit microprocessor, yet it is required to have following features

1. Support for power saving modes.

   Sensor nodes are always power constraint. Power consumption is one the strong basis for preferring one microprocessor to another. Power saving methods implemented in the processor are more important that other devices because it is a major power consuming device. We try to select processors with multiple power saving modes

   The power consumption of ATmega32 under various power saving modes is shown in Tab 4.2. We measured the current consumption using multimeter. The program running in the active mode was of UART transmission.

Mode	current(mA)
Active	4.02
Idle	3.66
Power Down	172 μA
Power Saving	174 μA
Standby	174μA

Table 4.2: Power characteristics of ATmega32

2. Suitable memory size

	ATmega32[1]	MSP430F157[10]	DS89C450[19]
Flash(K)	32	32	64
Permanent	1K	256B	0
ADC channels	8(10bits)	1(12bit)	0
programming	In System Programming	In Application programing	In Application programing
Interface support	JTAG, SPI, I^2C, USART	JTAG, SPI, I^2C, USART	USART
Operating Voltage	2.7-5.5V	1.8-3.6V	4.5-5.5V
I/O lines	32	48	32
Power down current	1μA	0.2μA	1μA

Table 4.3: Comparison of Microprocessors

Memory (flash) available on the processor dictates the size of program code that can be run from it. Complex routing algorithms and bigger protocol implementations produce bigger code, thus we need to have a fair idea of the code size before we choose a processor. This memory can be internal (on microprocessor itself) or external (interfaced by microprocessor buses). Another kind of memory EEPROM is also present on some microprocessor which can store some data on the microprocessor permanently like device ID, Seed number for Random number generation. The energy required for accessing EEPROM is much higher than Flash or RAM access.

3. Easy and fast reprogramming and even self programming

   If reprogramming a microprocessor requires it to be removed from the PCB, attached to some special programming module, then the programming becomes very tedious and time taking. These days microprocessor are available which allow In-Application programming which means that one can reprogram them without taking them out of the circuit. Some microprocessor also allow self reprogramming i.e. using the current program the microprocessor will generate a new code for itself which can be loaded on it by the current program. This feature is highly useful in scenarios were we require to change the program remotely or in cases were the current program on checking the requirements of application wants to modify itself to suit the requirement. (circuit showing ATmega32 getting programmed)

4. A/D channels

   A/D channels are required to digitalized the data given by analog circuits like sensors and power supply. Most of the microprocessor provide at least one A/D channel. A/D channel are graded by the number of bit resolution they provide. More the resolution in bits more is the precision. For example a 10 bit ADC over a input range of 0-5V can provide accuracy up to $5/2^{10}= 5/1000=0.005$V which is slightly better than a common multimeter. an additional signal amplifier is also present, which may be required to amplify signals before digitalizing them. In this case precision is lost. Moreover if the processor is not providing A/D support one can use external ADC's, though this would increase the size of the device.

5. Some common bus interface.

   Most of the digital sensors and communication device use some kind of digital bus for data transfer. Most commonly used buses are I^2C (TWI), SPI, CAN and USART. So a microprocessor supporting these buses would save a lot of time and memory which would be otherwise spend on implementing these on software. A JTAG interface is also provided in some microprocessor which can be used to debug the microprocessor.

4.3.2 Sensors

BTech project report of Priyank Mundra and Vishal Jalan containes essential information regarding selection of sensors for various application[16].

4.3.3 Power Supply

The Power Supply which is mostly derived from batteries, should have a set of desired characteristics that limit the choices available. The main issues are:

1. Size of the batteries

   The size of the batteries can be an important characteristic as minimizing the size of sensor node is one of the important goals.

2. Durability

   We would like to be able to recharge the batteries many times in order save costs.

3. Capacity

   The capacity of the battery pack in milli amperes-hour is important. It determines how long the node will run until a new charge is needed.

4. Initial cost

   This is an important parameter, but a higher initial cost can be offset by a longer expected life.

5. Environmental factors

   Used batteries have to be disposed of and some of them contain toxic materials.

For futher details refer [16]

4.3.4 Communication

BTech project report of Ashish Ranjan containes essential information regarding selection of communication devices for various application[17].

4.4 Processor

The processor we used for testing CENSE platform was ATmega32. It is available in DIP packaging, which makes its usage simple without any requirement of some custom board, it can be easily used on a breadboard so the testing also becomes easy. Most of the microprocessors with capability equivalent to ATmega32 come in surface mount packaging. It also supports self programming. Moreover Atmel processors are widely used so peer support is also easily available[2]. In our work we were tremendously helped by the forums that discuss Atmel related issues. The compilers and debuggers are also available for C language programming. So we don't need to do programming in assembly. It also supports In-Application programming. So we can easily program it using the parallel port of the computer. In fact we don't need to remove the processor module from the CENSE testbed for reprogramming it. Last but the least this processor was easily available.

We tested all the required features of the processor on the breadboard before making a PCB. Our test included interfacing with I^2C, USART, ADC, Timer, Power modes and Interrupts. For debugging we used the UART port of the microprocessor to provide information to the computer terminal using the serial port interface.

While making the PCB for the processor module, we used a general purpose PCB board. The PCB module is having the support for external Oscillator, though we can also use the internal clock generator. A separate docking point is provided for connecting the programming cable. Since the pin configuration of Atmel ATmega8535[1] is equivalent to ATmega32, we can replace the processor module by ATmega8535 also. On the board a FRC connector is also present to attach the module to the CENSE testbed. VCC and Vref for ADC are provided from the same source.

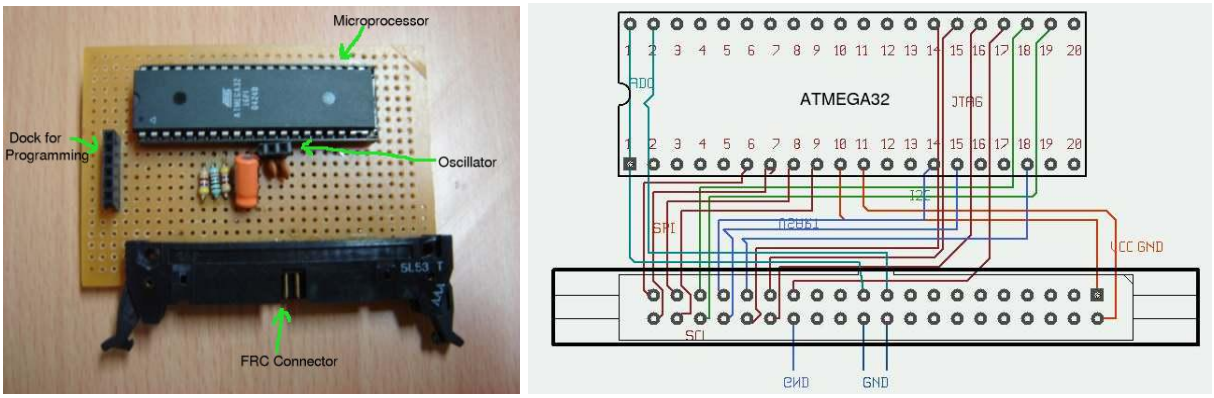

Figure 4.5: Processor module

4.5 Software

The increase in the Flash memory of the microprocessors has allowed, Real-Time operating systems with Tasking, Semaphores, Timer Management, Message Queues and preemptive scheduling support to be used on microprocessor. However most of the Operating Systems available are catering to one kind of applications like AVRX[3] is generally used in robots control. I have tested two open source operating system AVRX and FreeRTOS. Many paid operating system are available for ATmega32. I tested AVRX[3] and FreeRTOS[4]. AVRX was available only for ATmega8515, I have created a port for ATmega32[13].

TinyOS[8] was designed at University of California at Berkeley to cater to the specific needs of sensor network applications. This operating system is knit around the architecture of the nodes that UCB has produced namely mica, mica2 etc. We found that using this Operating System on CENSE would involve a lot of modification in the code, which also requires time for understanding the code. Hence we have left modification of TinyOS as operating system for CENSE as future work.

Currently we have developed some software libraries, which can be used to interface USART port, ADC and I^2C bus. In I^2C and ADC interface user has to give the name of device he wants to read data from, the libraries internally check the configurations of the device and access it. The data is returned back to the callee program.

Testing & debugging of software is very difficult in the microprocessor because we don't have a computer screen or LCD kind of utility. So we need to use various other debugging techniques like setting up output pins to produce a pattern when we reach a break point and then checking each pin if we are getting the desired output. If we are not using serial communications port of the microcontroller, we can use that port to send debugging information to the serial port of the computer. But if we are using the serial port then we can not use it to connect to computer, since output is not multidrop. In these cases we may have to use the oscilloscope, to monitor the output and decode it manually. Overall the debugging process is very tedious and time consuming.

Below is a sample of a code meant to read TMP175[16] I^2C sensor

```
i2c(int device)
{
...
...
TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN) ;//Comments Send START condition
while (!(TWCR & (1<<TWINT))) ; //Wait for TWINT Flag set
...
...
TWDR = 0x6A;//address
TWCR = (1<<TWINT) | (1<<TWEN);
...
```

```
...
TWDR =0x00 ;
TWCR = (1<<TWINT) | (1<<TWEN);
...
...
TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN) ;//Comments Send START condition
...
...
//Reading the temperature
TWDR = 0x6B; //address
TWCR = (1<<TWINT) | (1<<TWEN);
...
...
PORTB=data;
///now receiving second byte
TWCR = (1<<TWINT) | (1<<TWEN)| (1<<TWEA);
while (!(TWCR & (1<<TWINT))) ;
...
...
/// sending NAck

TWCR = 0|(1<<TWINT) | (1<<TWEN);
...
...
data=TWDR ;
//close the connection
TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);
}
```

4.6 Our implementation

Based on the architecture described in Sec. 4.2 we designed and built a system for weather sensing. We have built 5 nodes to test various protocols including static and multihop routing, which are used in Sensor networks. I will describe the configuration we put on the CENSE testbed (Fig. 4.1). For the computation I used ATmega32, as it provided many ADC channels, which are needed to interface analog weather sensors. On the sensing module we choose digital temperature sensor TMP175, LDR (Light Dependent Resistor), Humidity sensor, Accelerometer MMA2260D and Pressure sensor MPX4115[16]. On the communication we had Futuretechniks [15][17] Tx/Rx working 433.9MHz. The Power supply was given by 4 rechargeable Ni-Cd batteries[16].

4.6.1 Data Communication

We have tried to test some of the half duplex Data communication protocols on CENSE.

Two Way data transmission with dummy Data

1. Algorithm

   In two way data transmission we used only two nodes. Most important consideration in implementation of this protocol was synchronization. When one node is transmitting the second node should not start transmitting at the same time. Because in that case both the data would be lost.

   Our protocol is such that from the beginning one node ('A') would be in transmission mode and the other node ('B') would remain in the receiving mode. On reception of data at 'B', it will generate an ACK Ref. 4.6. If 'A' does not receive ACK it will wait for some time and then retransmit the

	A	B
1.	Check for synchronization character ('U' in our case) continuously	Transmit data
2.	Found 'U'	Wait for the Acknowledgment (ACK)
4.	Transmit ACK packet	If ACK received proceed
5.	Transmit DATA packet	Check for 'U' continuously
6.	Wait for ACK	Found 'U'
7.	if No ACK retransmit data	Read the whole packet
8.	if ACK received goto step 1	Transmit ACK and goto step 1

Table 4.4: Algorithms executed at the two node

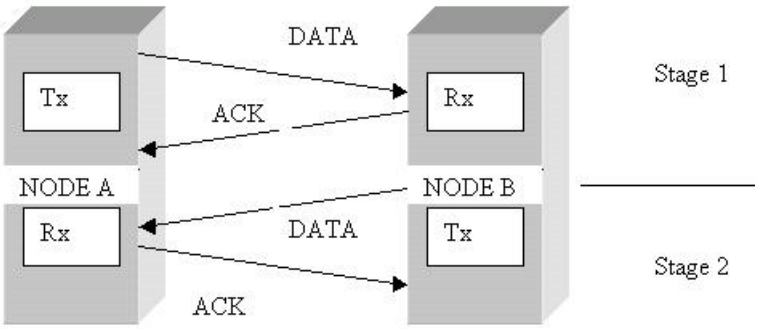

Figure 4.6: Arrow diagram showing the flow of Packets

data. 'A' on receiving the ACK would start listening to the channel waiting for 'B' to transmit. If 'A' receives the data from 'B' it will send back an ACK. This ACK would also mean that now 'B' will stop transmitting and 'A' will transmit its data. This process is repeated continuously.

2. Results

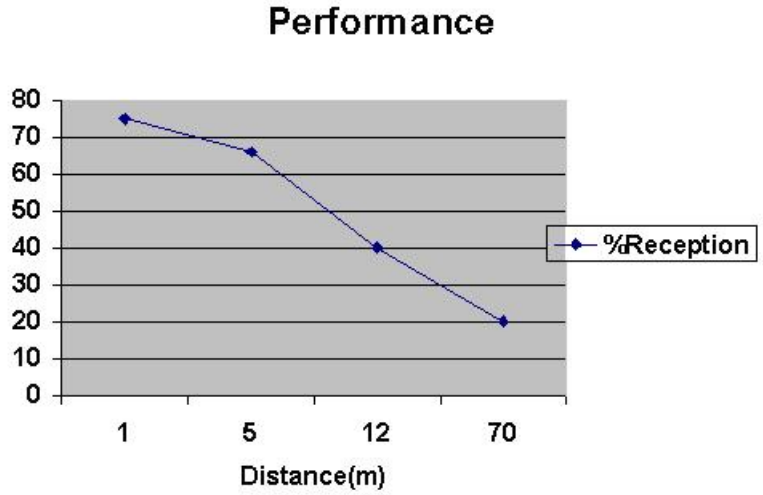

Figure 4.7: Success rate of Transmission

We tested this protocol in the Robotics lab, which is having many computers and some furniture. We tried to measure the success rate of transmission with the distance between the two nodes. We had

programmed the node in such a way that on reception of ACK packet they would change the output at one of the port (say from high to low) for some time, which could be easily detected on the oscilloscope. For this purpose we used two oscilloscopes each at one node. Success rate is defined as the number of ACK received to the number of packets send. Graph in the figure 4.7 plots the success rate to the distance between the two node.

Multiple nodes transferring data in single hop

Here we have taken 5 nodes, which are waiting for random time before transmitting the data. They also check if some other node is transmitting data before they start to transmit. In that case they again back-off by a random time interval. The transmission is initiated on reception of a message from the controller node. At the time of writing this report we had not fully finished the testing of the single hop.

We also hope to test a multihop protocol in near future.

Chapter 5

Conclusion and Future work

CENSE is an ongoing project. We have prepared a set of five nodes. After optimizing the design, a hundred such nodes would be prepared where actual testing could be done. Once an Application has been tested on CENSE all the components (modules) would be integrated on a single PCB. Thus the total size of the node would reduce considerably. The nodes would become more efficient and optimized because the conditions imposed by the bus would be removed.

As far as learning are concerned, this project offered us a extensive opportunities to learn how to diagnose causes of errors. Simple problems like soldiering the wrong pins have kept us engaged for hours. What may be working individually, may fail to work when integrated together. The mysteries of RF communication and the necessity time synchronization between sender and receiver were felt practically. We had chance to work on the MAC layer of communication and have a hands on experience. Most of us had never done soldiering before but now we have acquired skills to soldier very fine components even of SMD kind. The concepts of electronics were seen in practice.

Though we have all put in our best efforts, the work still remains unfinished. The future work required would be the development of an operating system for CENSE systems. We also need to test CENSE by choosing different components for a module, which we could not do because of time constraints.

Bibliography

[1] ATMEL. Website. *http://www.atmel.com/*.

[2] Avrfreaks. Website. *http://www.avrfreaks.net/*.

[3] Larry Barello. Website. *http://www.barello.net/*.

[4] Richard Barry. Website. *http://www.freertos.org/*.

[5] Elfsys. Website. *http://www.elfsys.net/*.

[6] Jeremy Elson and Deborah Estrin. Wireless sensor networks. *unpublished*, 2004.

[7] L.Girod T.Stathopoulos N.Ramanathan J.Elson D.Estrin E.Osterweil and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. *Proc. of SenSys*, 2004.

[8] Jason Lester Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[9] IITB. Website. *http://www.ee.iitb.ac.in/uma/ sensor/*.

[10] Texas Instrument. Website. *http://www.ti.com/*.

[11] Mainwaring Alan. Polastre Joseph. Szewczyk Robert. Culler David. Anderson John. Wireless sensor networks for habitat monitoring. *First ACM Workshop on Wireless Sensor Networks and Applications*, 2002.

[12] G.Simon M.Maroti A.Ledeczi G.Balogh B.Kusy A.Nadas G.Pap J.Sallai and K.Frampton. Sensor network-based countersniper system. *Proc. ACM SenSys*, 2004.

[13] Udayan Kumar. Website. *http://udayankumar.tripod.com/os.html/*.

[14] JHL Labs. Hardware profiles. *http://www.jlhlabs.com/*.

[15] Future Techniks Private Limited. Website. *http://www.futuretechniks.co.in/*.

[16] Priyank Mundra and Vishal Jalan. Design and modelling power supply and sensor for sensor network. Technical report, 2005.

[17] Ashish Ranjan. Wireless communication in sensor networks. Technical report, 2005.

[18] Akbar M. Sayeed Richard R.Brooks, Parameshwaran Ramanathan. Distributed target classification and tracking in sensor networks. In *Proc. of the IEEE*.

[19] Dallas Semiconductor and Maxim. Website. *http://www.maxim-ic.com/*.

[20] UCLA. Website. *http://www.janet.ucla.edu/WINS/*.

Appendix A

Circuit diagram for using ATMEGA32 without STK500

Components Required :

Chip : Atmega32 (may work with others of atmel avr series)

Resistors:

1. 220 Ohm - 1 (MISO)
2. 470 Ohm - 2 (SCK, MOSI)
3. 10 KOhm - 1 (RESET)
4. 1 KOhm - 1 (min) (LED)

Capacitors:

1. 22pF - 2 (Oscillator)
2. 10uF - 1 (RESET)

Oscillator:

1. 4MHz - 1

LED: - 1 (min)

DB2 Female connector: - 1 (MISO, MOSI, SCK, RESET, GND)

Bread Board or PCB: - 1 I have used Avrdude for programming

Configuration of AVR dude Locate avrdude.conf file (mine was in /usr/local/etc/). Create one new entry in Programmer types.

```
programmer
id = "try";
desc = "Lets see if this works";
type = par;
reset = 16;
sck = 1;
mosi = 2;
miso = 11;
;
```

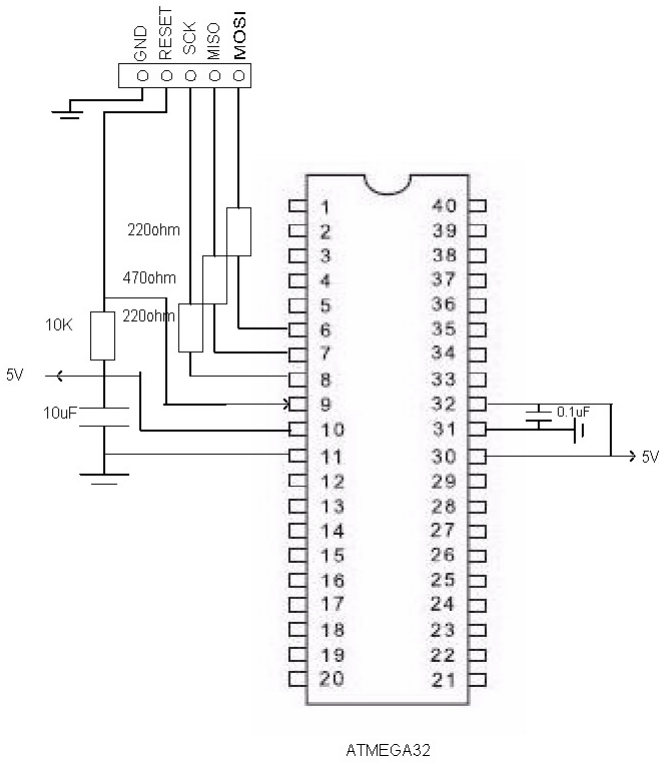

Figure A.1: Ciruit Diagram for In-Application programming of ATMEGA32