

**DATA CONTROL FOR SIGNAL SCAVENGING FOR A PERSONNEL  
DETECTION SYSTEM**

---

A Thesis

Presented to

The Faculty of the Graduate School  
At the University of Missouri-Columbia

---

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

---

by

UDAY GOVINDRAO SHRINIWAR

Dr. Harry Tyrer, Thesis Supervisor

JULY 2010

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

**DATA CONTROL FOR SIGNAL SCAVENGING FOR A PERSONNEL  
DETECTION SYSTEM**

Presented by

**Uday Govindrao Shriniwar,**

A candidate for the degree of

**Master of Science**

And hereby certify that, in their opinion, it is worthy of acceptance.

---

Dr. Harry Tyrer

---

Dr. Guilherme DeSouza

---

Dr. John Fresen

## **ACKNOWLEDGEMENTS**

I wish to express my sincere appreciation to Dr. Harry W. Tyrer, my advisor, for his invaluable guidance and encouragement in the completion of this study.

A special thanks to Richard Oberto, Jim Fischer and Anvari Reza for sharing their valuable knowledge about embedded engineering with me. I am also grateful to Rohan Neelgund and Krishna Devarkonda, my project mates, for suggesting ways to implement the protocols and for motivating me during the course of development. They have made available their support in a number of ways.

I owe my deepest gratitude to my parents. Special appreciation must be expressed to them for their belief in me, encouragement and financial support for my study in the United States. I am also thankful to my fiancé for being supportive and for encouraging me during my study of this project. Without her support, I could not have completed this project.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

## Table of Contents

ACKNOWLEDGEMENT.....	ii
LIST OF FIGURES .....	vii
LIST OF TABLES.....	ix
ABSTRACT.....	x
CHAPTER 1 : INTRODUCTION.....	1
CHAPTER 2 : LITERATURE REVIEW.....	3
CHAPTER 3 : METHODS.....	7
3.1. Hardware used.....	8
3.1.1. Microcontrollers used in all the systems.....	8
3.1.2. Programmer/Debugger.....	10
3.1.3. Radio Transceiver Module.....	11
3.2. Protocols/Standards/Bus used .....	13
3.2.1. Recommended Standard-232(RS-232).....	13
3.2.2. Serial Peripheral Interface (SPI) bus.....	13
3.2.3. MiWi P2P Protocol (IEEE 802.15.4).....	14
3.3. Integrated development environment (IDE) and compiler used .....	14
3.4. Software languages used .....	15
3.5. Window Hyper Terminal .....	15
3.6. Development System.....	15
3.6.1. Data acquisition and conditioning for the Development System.....	15
3.6.2. Microcontroller circuit for the Development system.....	16
3.6.3. Voltage regulator circuit (5V) used for the Development System .....	17

3.6.4.	Recommended Standard-232 (RS-232) Circuit used for the Development System .....	18
3.6.5.	Microcontroller program, the algorithm, and the frame format implemented for the Development System .....	18
3.6.6.	Experimental set up for the tests carried to assess the performance of the Development System .....	20
3.6.6.1.	Observation of sensor non-activation .....	21
3.6.6.2.	Observation of sensor activation.....	21
3.6.6.3.	Observation of data acquisition and transmission time .....	21
3.7.	Prototype System.....	22
3.7.1.	Data acquisition and conditioning for the Prototype System.....	22
3.7.2.	Microcontroller circuit used for the Prototype System.....	23
3.7.3.	Voltage regulator circuit (5V) used for the Prototype System .....	24
3.7.4.	Recommended Standard-232 (RS-232) Circuit used for the Prototype System.....	25
3.7.5.	Microcontroller program, the algorithm, and the frame format for the Prototype System .....	26
3.7.6.	Experimental set up for carrying out tests to assess the performance of the Prototype System .....	28
3.7.6.1.	Observation of sensor non-activation in the Prototype System .....	28
3.7.6.2.	Observation of sensor activation in the Prototype System .....	28
3.7.6.3.	Observation of the Prototype System data acquisition and transmission time .....	28
3.8.	Installed Floor .....	29
3.8.1.	Data acquisition and conditioning for the Installed Floor .....	30
3.8.2.	Microcontroller circuit used in the Installed Floor .....	31

3.8.3.	Voltage regulator circuit (3.3V) used in the Installed Floor .....	32
3.8.4.	Recommended Standard-232 (RS-232) Circuit used in the Installed Floor .....	33
3.8.5.	Installation Segment Electronics.....	34
3.8.6.	Installed Floor Networking.....	34
3.8.7.	Installed Floor Frame Formatting .....	36
3.8.8.	Experimental set up for the Installed Floor.....	41
3.8.8.1.	Observation of sensor non-activation in the Installed Floor.....	41
3.8.8.2.	Observation of sensor activation in the Installed Floor .....	41
3.8.8.3.	Observation of the Installed Floor data acquisition per second.....	41
CHAPTER 4 : RESULTS.....		42
4.1.	Development system .....	42
4.1.1.	Observation of sensor non-activation in the Development System .....	42
4.1.2.	Observation of sensor activation in the Development System .....	43
4.1.3.	Observation of the Development system data acquisition and transmission time .....	44
4.2.	Prototype system .....	46
4.2.1.	Observation of sensor non-activation in the Prototype System .....	46
4.2.2.	Observation of sensor activation in the Prototype System .....	47
4.2.3.	Observation of Prototype System data acquisition and transmission time .	48
4.3.	Installed floor .....	50
4.3.1.	Observation of sensor non-activation in the Installed floor.....	50
4.3.2.	Observation of sensor activation.....	52
4.3.3.	Observation of data acquisition per second .....	54

CHAPTER 5 : DISCUSSIONS .....	55
5.1. Microcontroller and software .....	55
5.2. Hardware used for the development, prototype and the installed floor .....	55
5.3. Protocols used in the development, prototype systems and the installed floor..	57
CHAPTER 6 : CONCLUSION .....	61
REFERENCES .....	63
APPENDICIES .....	65

# LIST OF FIGURES

<b>Figure 3-1:</b> The block diagram made up of three blocks,.....	8
<b>Figure 3-2:</b> The pin description of pins of PIC microcontroller involved in interfacing.....	11
<b>Figure 3-3:</b> The pin diagram of MRF24J40MA (8) .....	12
<b>Figure 3-4:</b> The connection between radio transceiver module.....	12
<b>Figure 3-5:</b> The block diagram of the Development System.....	16
<b>Figure 3-6:</b> The picture of microcontroller circuit used in the Development System. ....	16
<b>Figure 3-7:</b> The schematic of the circuit used for the Development System.....	17
<b>Figure 3-8:</b> The circuit diagram of regulator used to maintain 5V supply .....	17
<b>Figure 3-9:</b> The circuit diagram of Recommended Standard-232 (RS-232).....	18
<b>Figure 3-10:</b> The flow chart delineating the program.....	20
<b>Figure 3-11:</b> The block diagram of the Prototype System.....	22
<b>Figure 3-12:</b> The picture of the circuit boards used for the Prototype System. ....	23
<b>Figure 3-13:</b> The schematic of the microcontroller circuit.....	24
<b>Figure 3-14:</b> The circuit diagram of regulator used to maintain the supply voltage.....	25
<b>Figure 3-15:</b> The circuit diagram of Recommended Standard-232 (RS-232) .....	25
<b>Figure 3-16:</b> The flow chart used for writing the program for the Prototype System. ....	27
<b>Figure 3-17:</b> The picture of, the host node .....	30
<b>Figure 3-18:</b> The picture of peer node which transfer data to their master.....	30
<b>Figure 3-19:</b> The circuit diagram of the host node used in the Installed Floor.....	31
<b>Figure 3-20:</b> The schematic of microcontroller circuit implementation of peer nodes .....	32
<b>Figure 3-21:</b> The typical application circuit of 3.3V regulator.....	33
<b>Figure 3-22:</b> The circuit diagram of Recommended Standard-232 (RS-232) .....	33
<b>Figure 3-23:</b> The block diagram of the Installation Segment Electronics .....	34
<b>Figure 3-24:</b> The block diagram of the Installation Floor .....	35
<b>Figure 3-25:</b> The flow chart of program written for the host node.....	39



<b>Figure 3-26:</b> The flow chart of program written for the peer nodes .....	40
<b>Figure 4-1:</b> Data received on the Windows Hyper Terminal from microcontroller .....	42
<b>Figure 4-2:</b> Sensor display program when no sensor was activated (2) .....	43
<b>Figure 4-3:</b> Data received on Windows Hyper Terminal .....	43
<b>Figure 4-4:</b> Sensor display program when 2 sensors were activated (2) .....	44
<b>Figure 4-5:</b> Graph for Time in mS Vs No. of Active sensors .....	46
<b>Figure 4-6:</b> Data received on the Windows Hyper Terminal from microcontroller .....	47
<b>Figure 4-7:</b> Sensor display program when no sensor was activated (2) .....	47
<b>Figure 4-8:</b> Data received on the Windows Hyper Terminal .....	48
<b>Figure 4-9:</b> Sensor display program .....	48
<b>Figure 4-10:</b> Graph for Time in ms Vs No. of Active sensors .....	50
<b>Figure 4-11:</b> Data received on the Windows Hyper Terminal from microcontroller .....	51
<b>Figure 4-12:</b> The Sensor display program when no sensor was activated. ....	51
<b>Figure 4-13:</b> The data received on Windows Hyper Terminal via serial port. ....	52
<b>Figure 4-14:</b> Sensor display program .....	52

## LIST OF TABLES

<b>Table 2-1:</b> Table of all the microcontrollers available in the market.....	5
<b>Table 3-1:</b> List of all the microcontrollers used in this project.....	9
<b>Table 4-1:</b> The time taken by Development system to complete one polling cycle .....	45
<b>Table 4-2:</b> The time taken by each system to complete one polling cycle .....	49

## ABSTRACT

Injurious falls have been one of the major problems in elderly people and not providing medical assistance to those patients on time may increase complications. We developed a ‘Smart Carpet’, which detects the personal motion. It can be utilized to detect the falls and automatically call for assistance. This system has two major parts: the sensors, which sense the walking and the personal computer with internet connectivity, which displays the motion as well as can call automatically for assistance after detecting a fall. The lack of feasibility to maintain one computer per room raises the need of a smart electronic instrument to gather the sensor data of a room and forward it to a central computer. The use of a smart electronic device reduces the number of computers used and hence the cost. Along with this, increases the distance between sensors and the personal computer. Our aim was to develop a low cost and reliable electronics system, which can consistently accumulate data from sensors and transmit it to the personal computer. We used microcontrollers to achieve this electronics system. In the initial stages of development to gain the confidence on the system, we implemented systems using low numbers of sensors and after achieving success, built systems with larger numbers of sensors but using only one microcontroller. In the final stage of development, we created a wireless network in which 4 microcontrollers act as nodes and communicate with each other using wireless channel for transferring the sensor data of different areas to the personal computer. By achieving satisfactory results, we have gained a confidence that the electronics system developed in the concluding stage is highly reliable, accurate and can be extensively used for further development.

## CHAPTER 1 : INTRODUCTION

Falls are one of the major issues in Alzheimer's elderly patients. Study shows that 30% of elders fall each year and 5% of those falls result in fractures. An injury caused by falls might shorten the life anticipation and might increase the cost of care for the patient. Many researchers have conducted studies to prevent falls (16).

Temporary or permanent memory loss is the main cause of Alzheimer disease (7). After falling, patients might fail to remember to call for medical assistance, which result in late treatment, and hence more serious problems. On the other hand, monitoring requires manpower, and increased expenses.

Our motivation behind developing this system was to monitor elderly people and possibly detect their falls. Connectivity of the monitoring system to the internet helps distantly located family members to know the daily activities of their loved ones by visiting their respective web page. Achieving such a system will reduce the need of continuous monitoring and hence the cost.

We placed low cost sensors, aluminum foils, under a carpet to provide a sensor floor, amplified the signals generated from the sensors (1) and used that data to display on the computer monitor (2). A smart electronic device was implemented to process the sensor data, to format and to transfer it to the personal computer for display purpose. The use of the smart electronic device increased the scope for longer distance between the personal computer and the sensors, which in turn helped to cover a larger area. The

microcontrollers that are small, low cost and can be programmed to receive sensor data, process and transfer it to the personal computer were used to implement smart electronic device. This whole thesis discusses the role of microcontroller in this system.

Initially, to achieve confidence, we built faux floors with a low number of sensors i.e. the development system (4 sensors). This was followed by the prototype system (21 sensors); and after gaining confidence in these systems, we took a bigger step of implementing a floor with a larger number of sensors i.e. the installed floor (128 sensors).

The results achieved are reasonable in terms of efficiency, accuracy and repeatability. The results of one of the floors (development faux floor) have been published (3). A major concern was scalability of sensors; typically, a 12' X 10' room might require 120-240 sensors. For each circuit we designed, there was a sufficient margin to support such a scalability. This dissertation focuses on the evolution of the microcontroller circuitry which detects personnel walking on the carpet.

## CHAPTER 2 : LITERATURE REVIEW

Falls in elderly patients of Alzheimer's disease have been a major concern, since they may cause serious injuries, which may lead to broken bones (16). Timely delivery of medical treatment becomes of prime importance in the event of fall to avoid causing more serious problems. The application we developed is low cost, reliable, and efficient and in the event of falls can automatically call for medical assistance through the internet. Moreover, it can store sensor data and display it on a web page. Apart from monitoring the Alzheimer's patients, the application can also be used to monitor the elderly people's daily activities, mainly the people living alone, by accessing their web page.

We needed sensors and a visualizing program to sense and display the walking of a personal. For sensing the gait of a person we used aluminum foils as sensors. The aluminum foils generate an increased voltage when touched in some form compared to the untouched situation, Rohan Neelgund designed a circuit to amplify the voltage from mV to V (1). Furthermore, for displaying activated sensors on the personal computer Krishna Devarakonda wrote code to process the data frames sent from the microcontroller and use them for display purposes (2).

It is clear from the above discussion that there is a need of interface between the computer and the sensors. The main duty of the electronics we developed here is to create a communication path between the sensors and the personal computer by gathering data

from sensors and transferring it to the personal computer. So there are total two interfaces involved: one from sensors to the microcontroller and the second from the microcontroller to the personal computer.

Our major concern was to build a cost effective system by using satisfactory microcontrollers to cover a larger area i.e. to connect as many sensors as possible to one microcontroller, so we needed a microcontroller with an adequate number of input output pins. Furthermore, the Universal asynchronous receiver transmitter (UART)/Universal synchronous asynchronous receiver transmitter (USART) port is commonly available with all the personal computers and establishing communication between the computer and the microcontroller is easy and low cost.

We gained confidence by implementing systems with low numbers of sensors. Finally, for covering larger area i.e. with bigger numbers of sensors, we built a wireless network in which 4 microcontrollers associated with 4 groups of 32 sensors communicate with each other through wireless channel and transfer sensor data to a personal computer. For building the wireless network, we interfaced a radio transceiver module to the microcontroller using a Serial Peripheral Interface (SPI).

Because of the prior knowledge, we used PIC microcontrollers for developing our system. The Table 2-1 shows the list of some of the other microcontrollers in the market along with their manufacturers, which have all the features required for our application and can be used instead of PIC microcontroller.

**Table 2-1:** Table of all the microcontrollers available in the market along with their features useful for our application

<b>Microcontrollers</b>	ATmega16 A (Atmel) (4)	MC68HC908AB3 2 (Freescale Semiconductor) (5)	P87C52SBPN (NXP) (6)	PIC16F871 (Microchip)	PIC18F4455 (Microchip)
<b>Cost(\$)</b>	\$4.61	\$12.50	\$2.60	\$3.41	\$4.43
<b>Memory</b>	16K	32K	32K	3.5K	24K
<b>Input/Output Pins</b>	32	51	32	32	25
<b>UART</b>	Yes	Yes	Yes	Yes	Yes
<b>SPI</b>	Yes	Yes	Yes	No	Yes

Moreover, many wireless networking protocols are available in the market such as Zigbee, WiFi, MiWi, MiWi P2P, and Bluetooth. Out of these, we chose to use MiWi Peer to Peer (MiWi P2P) since it was compatible with PIC microcontroller and it did not require much change in the circuit we used for previous systems; however, we did use a different PIC microcontroller but other than that there were minor modifications.

Demo codes for MiWi Peer-to-Peer protocol are available in MiWi(TM) IEEE 802.15.4 Wireless Networking Protocol Stack, which is compatible to PIC18F87J11 (9). These demo codes had programs for two nodes, P2P Node 1 and P2P Node 2 in which the P2P Node 1 selects the least noise channel and waits for P2P Node 2 to get connected. After powering up, P2P Node 2 looks for P2P Node 1, after finding, gets connected and waits for user's actions on the hardware mentioned in the code. For initial studies, we preferred to use PDIP packaged microcontrollers over the surface mountable devices. The non-availability of PIC18F87J11 in PDIP package made us look for another PIC microcontroller. We chose PIC18F4455, studied the demo code and made changes in the



code as per our requirements and according to pin configurations of PIC18F4455 microcontroller.

## CHAPTER 3 : METHODS

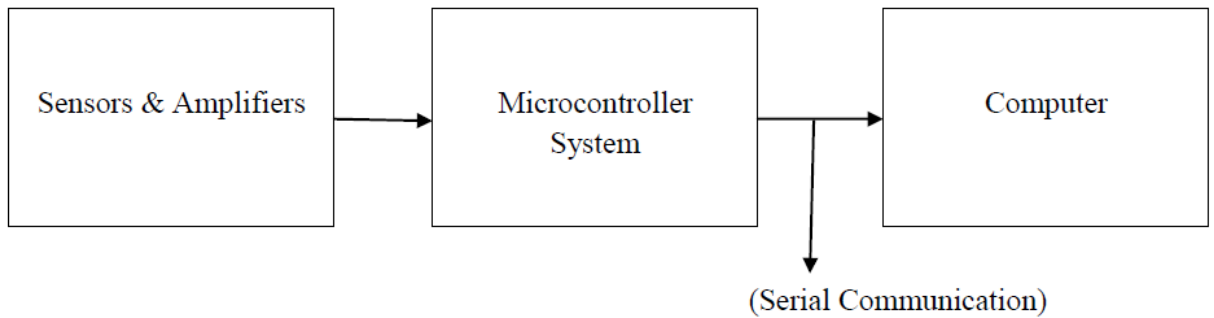
For sensing the location of a person we used aluminum foils as a sensor (1). We activated the foils by touching it in some form, the aluminum foils then generate an increased voltage in comparison to the untouched situation. Rohan Neelgund designed a circuit to amplify the voltage from mV to V (1).

The data gathered from the amplifiers can be interfaced directly to a personal computer but with unnecessary complexity. Furthermore, as the number of sensors increases the complexity increases so we avoid connecting the sensors directly to the personal computer. The microcontroller can provide a useful interface by taking the data from the amplifiers, processing it and forwarding it to the personal computer. While working on this project my focus was to build microcontroller circuits for each system we developed. Moreover, on the personal computer we needed a display program to show the activated sensors. Krishna Devarakonda wrote code to process the data frames received from the microcontroller and display them. We used serial communication to transfer the data from the microcontroller to the personal computer (2).

For testing purposes, we developed several different floors beginning with a low number of sensors and increasing the number of sensors after successful installation of each floor system. We built a development board using 4 (2 by 2) sensors. After successful accomplishment of this system, we moved on to 21 (7 by 3) sensors and named it the Prototype System. Then we built a 32 (8 by 4) sensors and called it as the

Installation Segment. Finally took a bigger step of development where we built a wireless network using 4 Installation Segments and referred it as the Installed Floor.

Figure 3-1 shows the block diagram made up of three blocks, where the first block consists of sensors and amplifiers, the second block consists of the microcontroller and the third consists of a personal computer.



**Figure 3-1:** The block diagram made up of three blocks, where the first block consists of sensors and amplifiers, the second block consists of microcontroller and the third consists of a personal computer. The microcontroller program polls all the connected sensors for activation, and sends a formatted frame to the personal computer using Recommended Standard-232 (RS-232).

### **3.1. Hardware used**

#### **3.1.1. Microcontrollers used in all the systems**

We required a low cost, fast processing microcontroller with an adequate number of input/output pins to connect variable numbers of sensors. Moreover, in order to establish the communication with the personal computer, we required a Universal asynchronous receiver transmitter (UART)/Universal synchronous asynchronous receiver transmitter (USART) port.

Table 3-1 shows list of all the microcontrollers along with some of the important features used in each system.

**Table 3-1:** List of all the microcontrollers used in this project along with some of the important features used in each system

<b>Applications</b>	<b>Development system, Prototype system</b>	<b>Installed floor</b>
<b>Microcontrollers Used</b>	PIC16F871	PIC18F4455
<b>Program Memory Type</b>	Flash	Flash
<b>Program Memory Size</b>	3.5K	24K
<b>Additional data storage(Data EEPROM)</b>	64 Bytes	256 Bytes
<b>CPU Speed</b>	5MIPS	12MIPS
<b>Digital Communication Peripherals</b>	1-A/E/USART,	1-A/E/USART, 1-MSSP(SPI/I2C)
<b>Voltage range</b>	2v to 5.5v	2v to 5.5v
<b>Input / Output Pins</b>	32	35
<b>Package</b>	Plastic Dual-In-line Package (PDIP)	Plastic Dual-In-line Package (PDIP)
<b>Pin Count</b>	40	40

There were only 4 and 21 sensors in the Development and the Prototype Systems so the size of the program written for them was smaller as in comparison to that of the Installed Floor. The 3.5Kbytes of program memory was sufficient for them, so we could continue using PIC16F871 for both the systems. Also, the program memory is of flash type, which enables us to erase and program it several thousand times. Furthermore, we utilized the Universal asynchronous receiver transmitter (UART)/Universal synchronous asynchronous receiver transmitter (USART) port to establish the communication with the personal computer.

In the Installed Floor, we created a Peer-to-Peer wireless network. A radio transceiver module was interfaced to the microcontroller using Serial Peripheral Interface (SPI). The program for a wireless network involves the implementation of all the layers

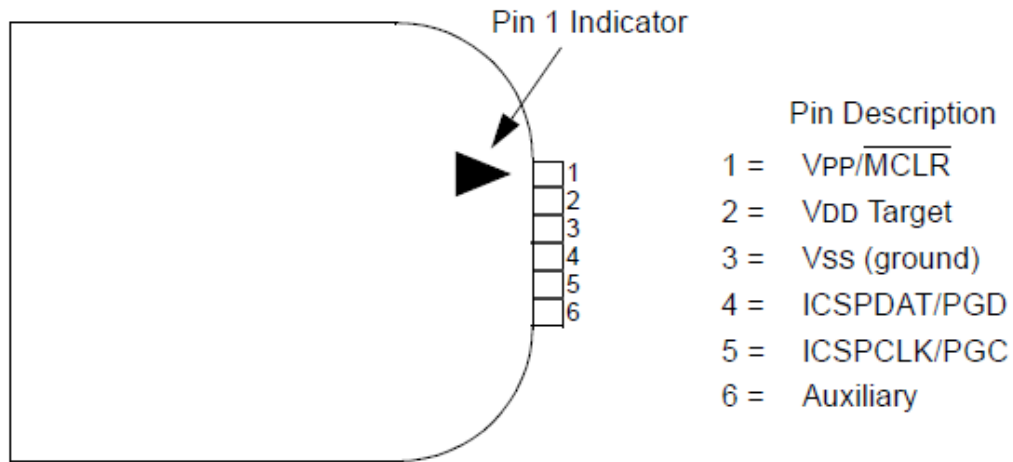
of the networking so the required Program Memory Size was much larger than 3.5Kbytes. We used PIC18F4455 since it has Serial Peripheral Interface (SPI) as well as 24Kbytes of flash type program memory. Additionally, the 35 input/output pins gave us scope to test our system by connecting more sensors to one microcontroller, a Universal asynchronous receiver transmitter (UART)/Universal synchronous asynchronous receiver transmitter (USART) port supports RS-232 so the use of different microcontroller did not require much of changes in the previous system's circuit.

We used PIC16F871 and PIC18F4455 microcontrollers with 40 pin count and Plastic Dual-In-line Package (PDIP) so we could use the same circuit designs for all the three systems i.e. the Development System, Prototype System and the Installed Segment.

### **3.1.2. Programmer/Debugger**

The requirements of programmer/Debugger are cost effectiveness, portability, and user friendly; it should be compatible with most of the PIC microcontrollers. We chose PICKit 2 Programmer/Debugger since it is portable, easy to use, easy to interface, compatible with almost all the PIC microcontrollers, and costs \$34.99 (12).

Figure 3-2 shows the pin description of pins of PIC microcontroller involved in interfacing the PIC microcontroller with PICKit 2. For interfacing PICKit 2 with the microcontroller we need to mount a 6 pin header on the circuit board (13).



**Figure 3-2:** The pin description of pins of PIC microcontroller involved in interfacing the PIC microcontroller with PICKit 2. For interfacing PICKit 2 with the microcontroller we need to mount a 6 pin header on the circuit board (13).

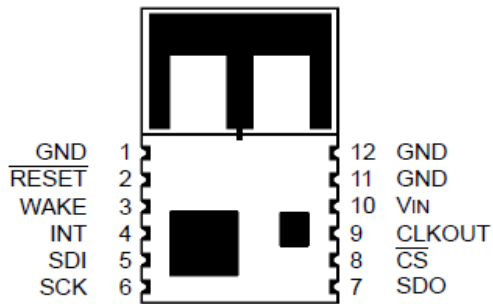
The PICKit 2 comes with a Universal Serial Bus (USB) cord whose one end is of type ‘A’ plug while its other end is ‘Mini-B’ type plug. The personal computer should have a Universal Serial Bus (USB) port in order to connect to the PICKit 2. The ‘A’ type plug of Universal Serial Bus (USB) goes to the personal computer and ‘Mini-B’ type plug to PICKit 2. In addition to programming the PICKit 2 can also be used as a debugger; to step through the program, find errors and debug (13).

### 3.1.3. Radio Transceiver Module

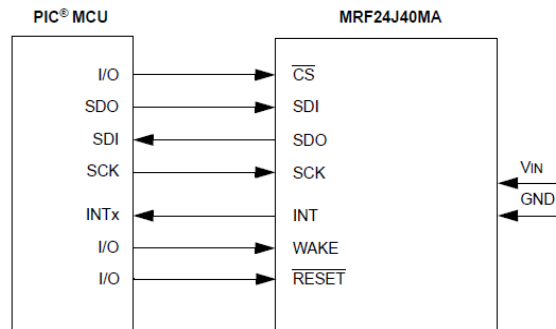
At the later stage of development, we created a wireless sensor network using MiWi Peer-to-Peer protocol (IEEE802.15.4). For creating a wireless sensor network, we selected MRF24J40MA, a transceiver supporting MiWi Peer-to-Peer protocol. This module is a certified 2.4 GHz IEEE 802.15.4 radio transceiver module. It has an integrated PCB antenna, matching circuitry, and it supports the ZigBee, MiWi and MiWi

Peer-to-Peer protocols. The MRF24J40MA module connects to most of the PIC microcontrollers via a 4-wire Serial Peripheral Interface (SPI) (8).

The Figure 3-3 shows the pin diagram of MRF24J40MA and the Figure 3-4 shows the connection between radio transceiver module and any PIC microcontroller with SPI capability. (8)



**Figure 3-3:** The pin diagram of MRF24J40MA (8)



**Figure 3-4:** The connection between radio transceiver module and any PIC microcontroller with SPI capability (8)

## **3.2. Protocols/Standards/Bus used**

### **3.2.1. Recommended Standard-232(RS-232)**

We chose to establish the connection between the microcontroller and the personal computer through Recommended Standard-232 (RS-232) because it takes only 4 pins of microcontroller to create the interface. Also it is common to have a serial port in all the personal computers. In our project, the microcontroller was the transmitter of data and the personal computer was the receiver of data, so for communication we used only 3 pins (TXD, GND, Vcc) out of 9 pins provided on the serial port.

### **3.2.2. Serial Peripheral Interface (SPI) bus**

The radio transceiver module required to be interfaced with PIC microcontroller using Serial Peripheral Interface. The Serial Peripheral Interface (SPI) enables us to use only 4 pins and establish communication between two electronic devices. In this interface, the devices communicate with each other in Master and Slave fashion, in which the master initiates the communication. This connection is synchronous and the signals carrying data go in both the direction concurrently. Following is the list of 4 pins used for establishing interface,

1. SCK Serial Clock (Clock from master for synchronizing purpose)
2. SDI Serial Data In (Data received from master)
3. SDO Serial Data Out (Data sent to master)
4. CS Chip Select (Since there can be many slaves connected to only one master, master has to select the chip before starting communication. This pin is active



low, so master has to give low on this pin to select the slave. Since we used only one transceiver module per node, this pin was always kept low).

### **3.2.3. MiWi P2P Protocol (IEEE 802.15.4)**

For increasing the number of sensors and cover a larger area, we used wireless technology. The wireless communication reduces the need of cables and other supporting hardware, and hence the cost and maintenance of the hardware. Our aim was to implement a simple wireless network which will satisfy our requirement of covering larger area. We chose to use MiWi Peer-to-Peer. It is a wireless networking protocol developed by Microchip, which works on 2.4GHz IEEE 802.15.4. It has modified Media Access Control (MAC) layer of IEEE 802.15.4. This protocol allows the transceiver to choose least noise channel among the 16 available channels and work on the same. It also has a unique feature in which in the event of collision of data or loss of data due to the noise in the channel the transceiver can switch to low noise channel (8).

### **3.3. Integrated development environment (IDE) and compiler used**

We utilized Microchip's MPLAB Integrated development environment (IDE) v8.50 to develop programs for PIC microcontrollers using assembly and Embedded C languages (11). The compiler used was MPLAB C18 v3.35 in LITE mode (10). The Integrated development environment (IDE) and C18 compilers are for free and can be downloaded from Microchip website (11) (10).

### **3.4. Software languages used**

The PIC microcontroller is a Reduced Instruction Set Computer (RISC). The assembly language of PIC is consisting of only 35 instructions, which simplified the code development. We developed the software for the Development System and the Prototype System, and the size of code increased. Also writing the code in assembly language slowed down the speed of coding. So we switched to Embedded C, which made writing the code easier, and hence easier to read and understand the code. Moreover, training in C language is wide spread and easier for the development.

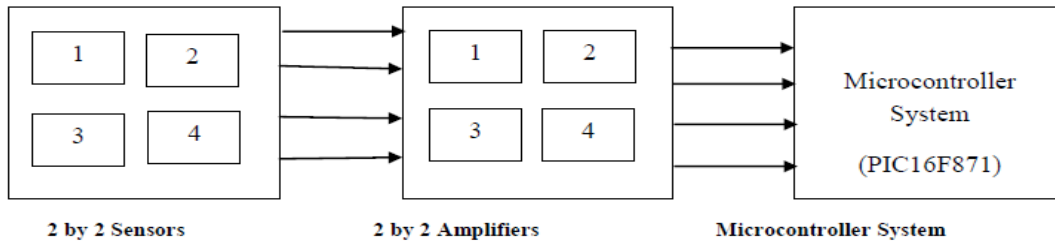
### **3.5. Window Hyper Terminal**

Use of the Windows Hyper Terminal helped us to check the proper initialization of the serial port and also the communication between the microcontroller and the personal computer. This application comes with all versions of Windows operating system except Windows 7.

### **3.6. Development System**

#### **3.6.1. Data acquisition and conditioning for the Development System**

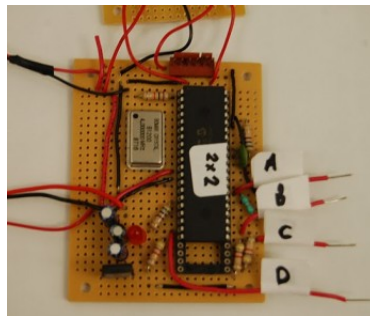
We constructed a 4 sensor faux floor and called it the Development System. The four sensors were connected to 4 amplifiers and the outputs of amplifiers were connected to the 4 input pins of the microcontroller. Figure 3-5 shows the block diagram of the development system. The sensors were arranged in 2 rows and 2 columns (1) (3).



**Figure 3-5:** The block diagram of the Development System.

The sensors were arranged in 2 rows and 2 columns.

Figure 3-6 shows the picture of microcontroller circuit used in the Development System. The wires with labels A, B, C, and D are input pins of the microcontroller and are connected to the outputs of the amplifiers (3).

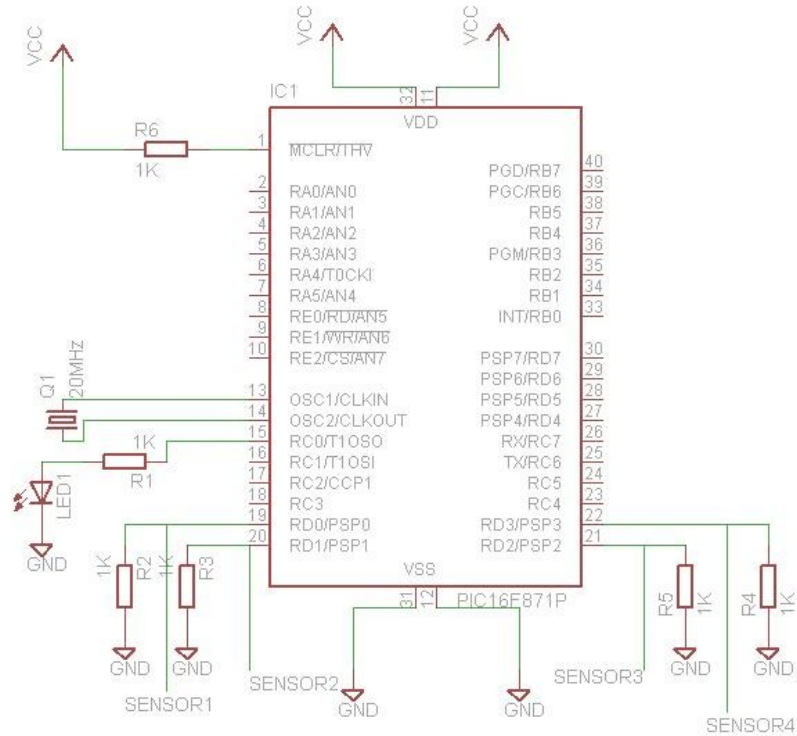


**Figure 3-6:** The picture of microcontroller circuit used in the Development System.

The wires with labels A, B, C, and D are input pins of the microcontroller and are connected to the outputs of the amplifiers.

### 3.6.2. Microcontroller circuit for the Development system

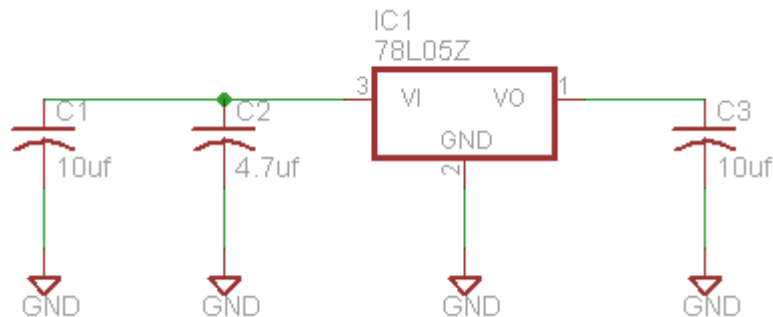
Figure 3-7 shows the schematic of the circuit used for the Development System. It includes a PIC16F871 microcontroller, a 20MHz crystal, 5V voltage regulator, a Light Emitting Diode, a pin header (Section 3.1.2) to get connected with PICKit 2 programmer/debugger (3).



**Figure 3-7:** The schematic of the circuit used for the Development System. It includes a PIC16F871 microcontroller, a 20MHz crystal, 5V voltage regulator, a Light Emitting Diode, a pin header (Section 3.1.2) to get connected with PICKit 2 programmer/debugger.

### 3.6.3. Voltage regulator circuit (5V) used for the Development System

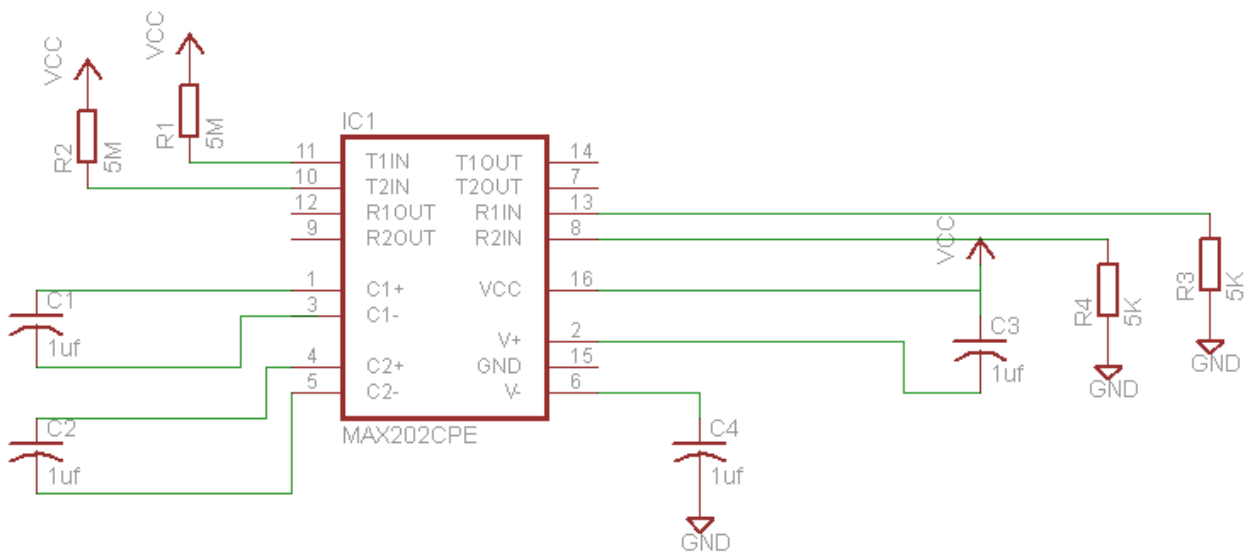
The Figure 3-8 shows the circuit diagram of regulator used to maintain 5V supply the microcontroller circuit.



**Figure 3-8:** The circuit diagram of regulator used to maintain 5V supply for the microcontroller circuit.

### 3.6.4. Recommended Standard-232 (RS-232) Circuit used for the Development System

Figure 3-9 shows the circuit diagram of Recommended Standard-232 (RS-232) used for establishing the connection between the microcontroller and the personal computer (17).



**Figure 3-9:** The circuit diagram of Recommended Standard-232 (RS-232) used for establishing the connection between the microcontroller and the personal computer (17).

### 3.6.5. Microcontroller program, the algorithm, and the frame format implemented for the Development System

The microcontroller program continuously polls the four input pins of the microcontroller and, sends a formatted frame to the personal computer through the serial port. We defined a frame for each polling cycle. The frame begins with ASCII character

'S' includes the polled data and ends with ASCII character 'E'. Each sensor is represented with ASCII character A, B, C, and D.

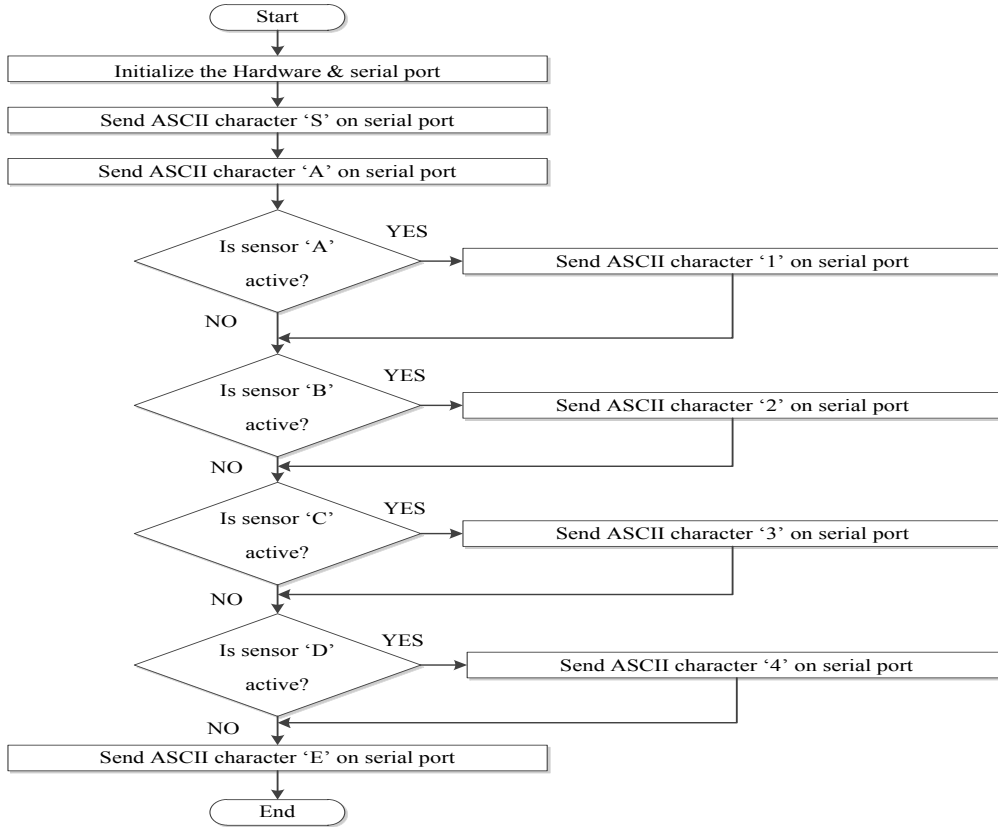
Following is a sequence of instructions for each polling cycle

1. The microcontroller sends ASCII character 'S' on the serial port which represents a start of a data frame.
2. The microcontroller sends the ASCII character 1, 2, 3, and 4, which represent sensors A, B, C, and D respectively (Shown in Figure 3-6), if any sensor is found active.
3. After polling all the sensors, it sends ASCII character 'E' which represents the end of a data frame.

Figure 3-10 a flow chart delineating the program written for the Development System. It also includes the details of the implementation of the list of instructions

mentioned

above.



**Figure 3-10:** The flow chart delineating the program written for the Development System. It also includes the details of the implementation of the list of instructions mentioned above.

### **3.6.6. Experimental set up for the tests carried to assess the performance of the Development System**

We describe below the experimental setup of the tests carried out to assess the performance of the Development System. The setup consisted of a power supply, microcontroller circuit, an RS-232 cable and a personal computer.

#### **3.6.6.1. Observation of sensor non-activation**

All the input pins of the microcontroller (shown in Figure 3-6) were connected to ground and output data was read on the Windows Hyper Terminal program. Using the same grounded input pins we displayed the sensor data on the sensor display program (2).

#### **3.6.6.2. Observation of sensor activation**

All the input pins from the microcontroller (Shown in Figure 3-6) except 'A' and 'B' were directly connected to ground, pins 'A' and 'B' were connected to 5v supply and output data was taken on the Windows Hyper Terminal. The sensor display program was observed after keeping the same activation on the input pins (2).

#### **3.6.6.3. Observation of data acquisition and transmission time**

For this test, we wrote a new program in which the microcontroller follows instructions as follows, Start the clock ticks, send ASCII character 'S', poll all the sensors, send data if any sensor is found active, complete the data frame by sending 'E', stop the timer, convert the count in timer into ASCII characters and send the number to the Windows Hyper Terminal, this number we converted into ms. We recorded time for 0 sensors activated, then 1 sensor activated and then 2, 3, and 4 sensors activated. The same tests were repeated using 3 baud rates (19200, 57600, and 115200). These tests were undertaken to determine the expected data acquisition time. Since the future implementation requires more sensors, it is essential to know the margin of time available for data acquisition.



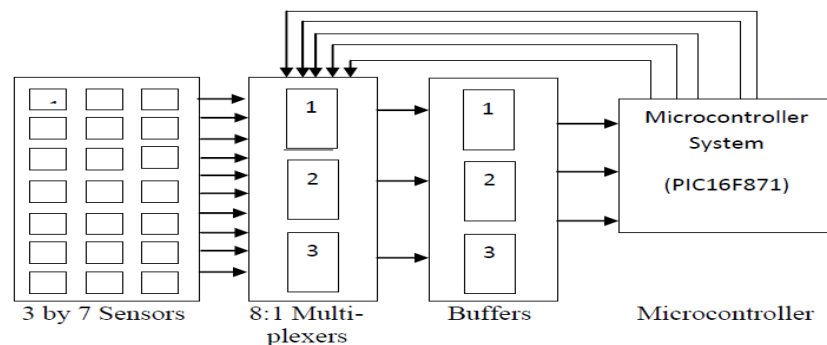
### 3.7. Prototype System

The successful demonstration of the Development S was followed by the Prototype System containing 21 sensors arranged in 3 columns referred as A, B, and C each column having 7 sensors named as A1-A7, B1-B7, and C1-C7 (1).

#### 3.7.1. Data acquisition and conditioning for the Prototype System

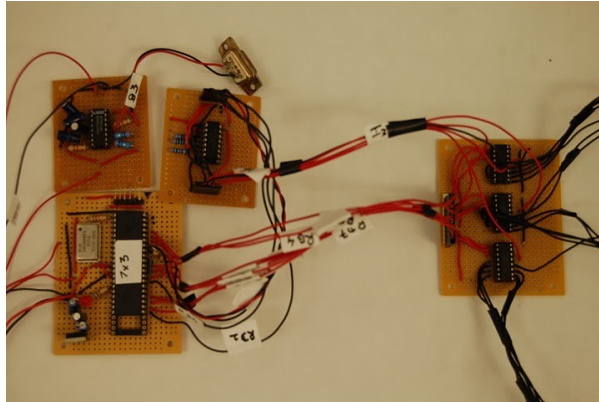
In this system, we used 21 amplifiers for 21 sensors. The outputs of 21 amplifiers were given to input pins of three 8:1 digital multiplexers. The select lines of multiplexers were connected to the output pins of the microcontroller, 3 output pins for each multiplexer's select lines, making a total of 9 pins for three 8:1 multiplexers. The outputs of multiplexers were given to the microcontroller through a buffer. The GND and Vcc of the microcontroller, the multiplexers and the buffer were connected together.

Figure 3-11 shows block diagram of the Prototype System consisting of the microcontroller, buffer, and three 8:1 multiplexers.



**Figure 3-11:** The block diagram of the Prototype System consisting of the microcontroller, buffer, and three 8:1 multiplexers.

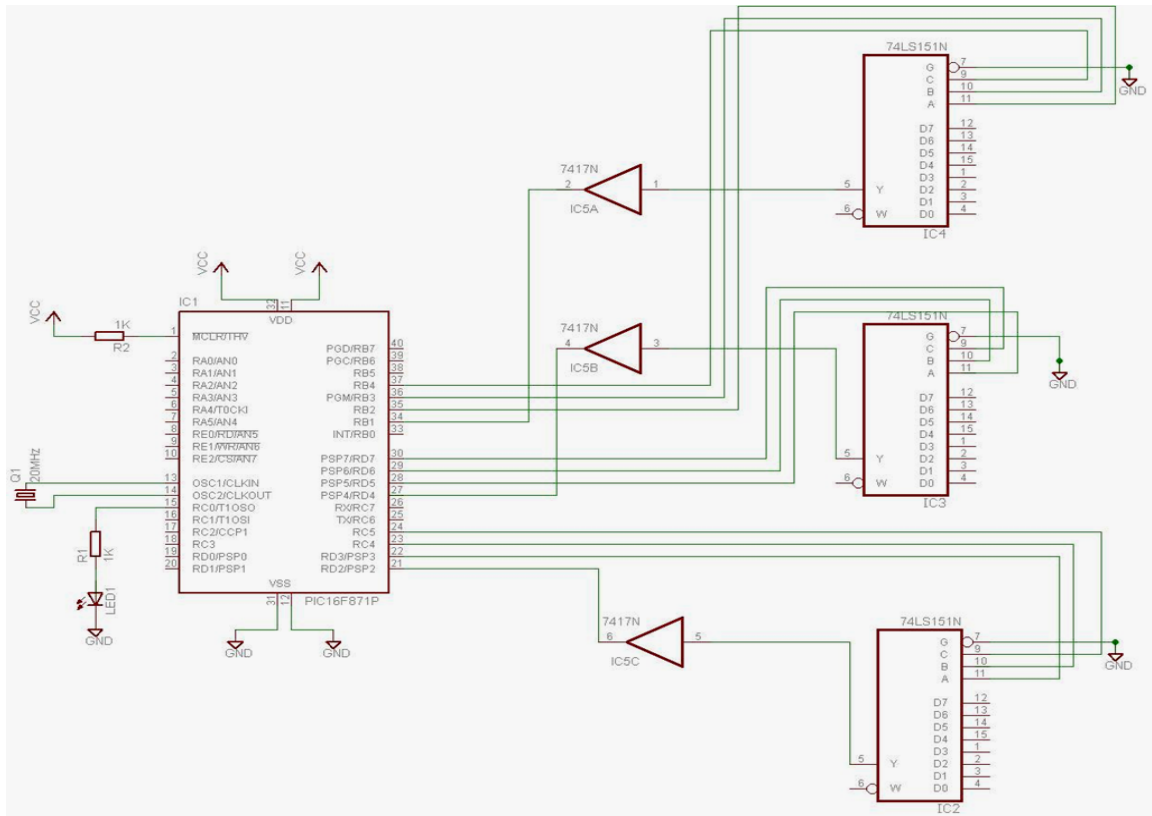
Figure 3-12 shows the picture of circuit boards used for the Prototype System. This includes boards of the PIC16F871 microcontroller circuit, a buffer circuit, three 8:1 multiplexers' circuit, and a Recommended Standard-232 (RS-232) circuit.



**Figure 3-12:** The picture of the circuit boards used for the Prototype System. This includes boards of the microcontroller circuit, a buffer circuit, three 8:1 multiplexers' circuit, and a Recommended Standard-232 (RS-232) circuit.

### **3.7.2. Microcontroller circuit used for the Prototype System**

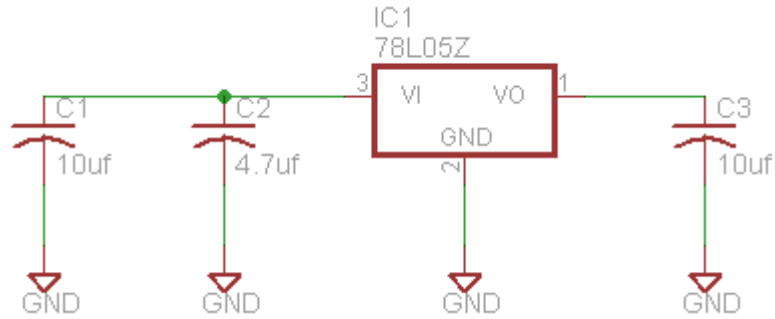
Figure 3-13 shows the schematic of the microcontroller circuit used for the Prototype System. The circuit used for the Prototype System is similar to that used in the Development System i.e. it also has the PIC16F871 microcontroller, a 20MHz crystal, a Light Emitting Diode, a pin header (**Section 3.1.2**) for connecting the microcontroller to the PICKit 2 programmer. In addition to these it has a buffer and multiplexer circuit. The microcontroller is connected to three 8:1 digital multiplexers and the output of multiplexer is given to the microcontroller through buffers.



**Figure 3-13:** The schematic of the microcontroller circuit used for the Prototype System. The circuit used for the Prototype System is similar to that used in the Development System i.e. it also has the PIC16F871 microcontroller, a 20MHz crystal, a Light Emitting Diode, a pin header (Section 3.1.2) for connecting the microcontroller to the PICKit 2 programmer. In addition to these it has a buffer and multiplexer circuit.

### 3.7.3. Voltage regulator circuit (5V) used for the Prototype System

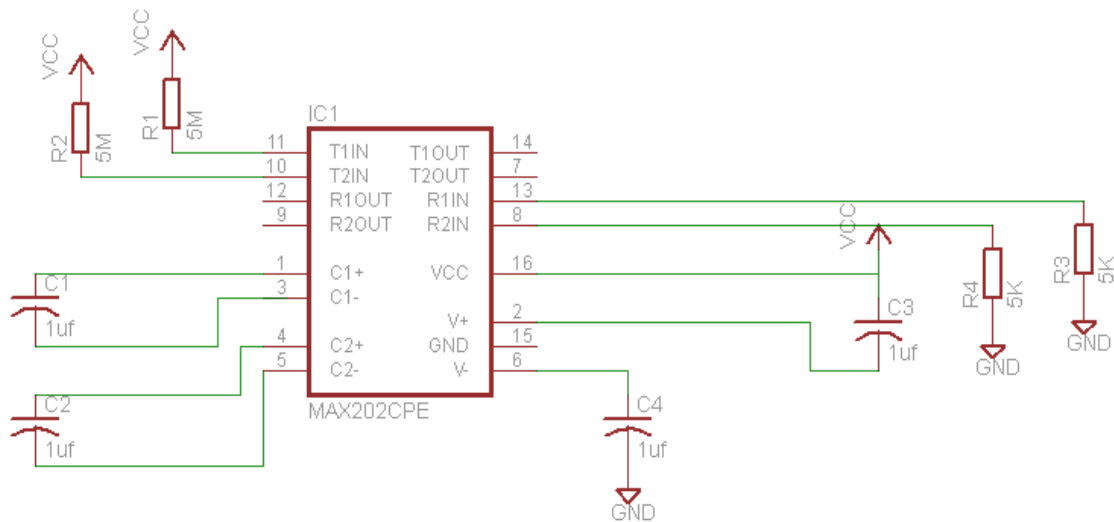
The Figure 3-14 shows the circuit diagram of regulator used to maintain the supply voltage of 5V to the microcontroller circuit. This regulator circuit is exactly similar to the one used in the Development System.



**Figure 3-14:** The circuit diagram of regulator used to maintain the supply voltage of 5V to the microcontroller circuit. This regulator circuit is exactly similar to the one used in the Development System.

### 3.7.4. Recommended Standard-232 (RS-232) Circuit used for the Prototype System

The Figure 3-15 shows the circuit diagram of Recommended Standard-232 (RS-232) used for establishing the connection between the microcontroller and the personal computer. This circuit is exactly similar to the one used in the Development System (17).



**Figure 3-15:** The circuit diagram of Recommended Standard-232 (RS-232) used for establishing the connection between the microcontroller and the personal computer. This circuit is exactly similar to the one used in the Development System (17).

### **3.7.5. Microcontroller program, the algorithm, and the frame format for the Prototype System**

The microcontroller program selects each of the 21 lines of multiplexer (for each sensor in each column) connected to amplifiers, polls the three inputs pins, and sends a formatted frame to the personal computer via the serial port. The 7 sensors in each column are referenced by ASCII characters from 1 to 7. We defined a frame with some addition to the one implemented for the Development System. The frame begins with the ASCII character 'S', adds character 'A', includes activated sensors' data in column 'A', adds the character 'B', includes the activated sensors' data in column 'B', adds the character 'C', includes the activated sensors' data in column 'C' and ends with character 'E'.

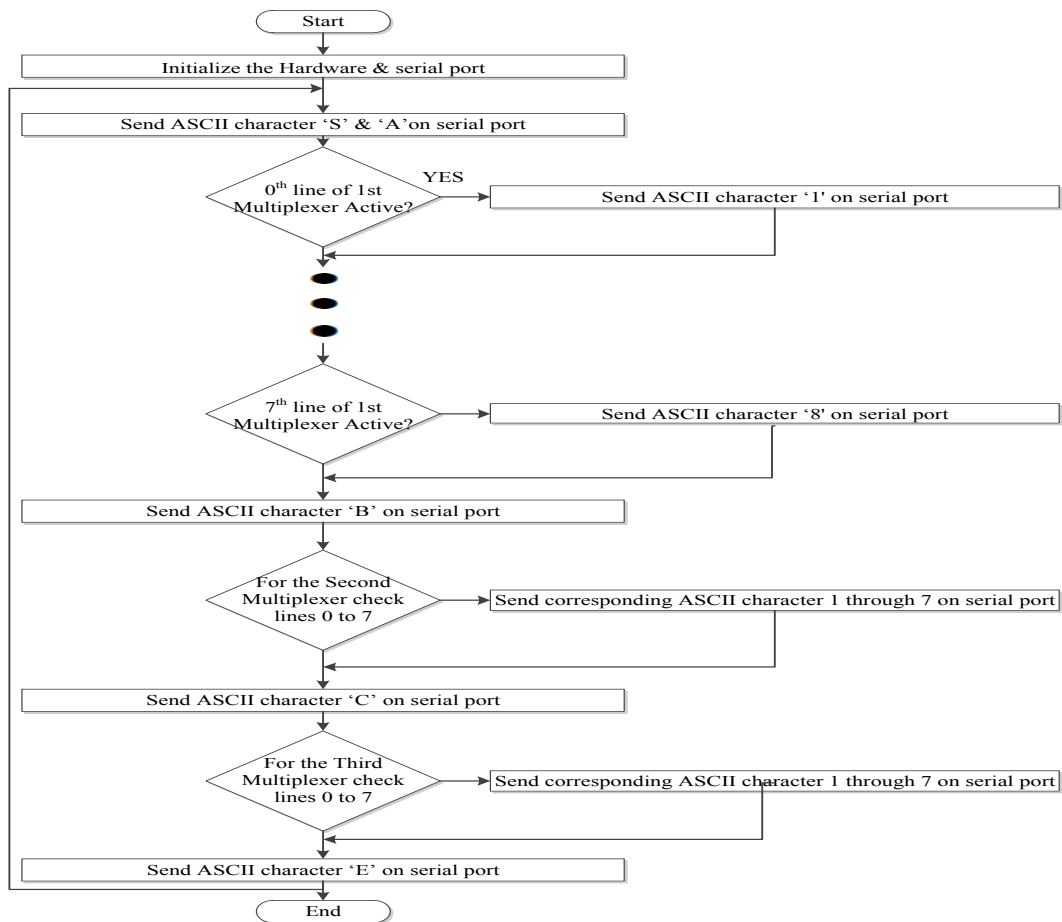
Following section discusses the instructions followed in the software to implement the data frame.

1. Before the polling starts the microcontroller sends ASCII character 'S' (Start of a frame) on the serial port,
2. It sends ASCII character 'A' to notify that column 'A' is being polled.
3. If any sensor is found active then ASCII character of respective sensor is sent on the serial port.
4. After polling column 'A' the microcontroller sends ASCII character 'B' indicating that column 'B' is being polled.
5. If any sensor is found active then ASCII character of respective sensor is sent on the serial port.

6. After polling column 'B' the microcontroller sends ASCII character 'C' indicating that column 'C' is being polled.
7. If any sensor is found active then ASCII character of respective sensor is sent on the serial port.
8. After polling all the sensors, it sends ASCII character 'E' (End of a frame).

Figure 3-16 shows the flow chart used for writing the program for the Prototype System.

It also includes the instructions mentioned above.



**Figure 3-16:** The flow chart used for writing the program for the Prototype System. It also includes the instructions mentioned above.

### **3.7.6. Experimental set up for carrying out tests to assess the performance of the Prototype System**

The three tests carried out were the duplicate of those described in section 2.8. As before, the setup consisted of a power supply, the microcontroller circuit, a RS-232 cable and the personal computer.

#### **3.7.6.1. Observation of sensor non-activation in the Prototype System**

All the input pins of 3 multiplexer's were connected to the ground and output data was read on Windows Hyper Terminal program. Using the same grounded input pins we displayed the sensor data on the sensor display program (2).

#### **3.7.6.2. Observation of sensor activation in the Prototype System**

The input pins of 3 multiplexers corresponding to the first sensors in columns 'A', 'B' and 'C' were connected to 5V, the remaining pins were connected to ground and output data was taken on Windows Hyper Terminal program. The sensor display program was observed after keeping the same activation on input pins (2).

#### **3.7.6.3. Observation of the Prototype System data acquisition and transmission time**

For this test, we wrote a new program in which the microcontroller follows the instructions as follows, Start the clock ticks, follow the steps described in Section 3.8.5 i.e. send ASCII character 'S' via the serial port, send characters A, B, and C and in between those characters sends polled data of all the sensors in each column, complete the data frame by sending ASCII character 'E', stop the timer, convert the number in timer into ASCII character and sent it to Windows Hyper Terminal. This number we converted into ms. The sensors were activated one after the other and time was recorded.

So we recorded time for 0 sensors activated, then 1 sensor from column A activated then 2, 3, 4, 5, 6, and 7 sensors activated. The same tests were repeated using 3 baud rates (19200, 57600, and 115200). The purpose of carrying out this test was similar to that for the Development System i.e. to know the expected data acquisition time as well as to gain an idea about the margin of time available for the data acquisition.

### **3.8. Installed Floor**

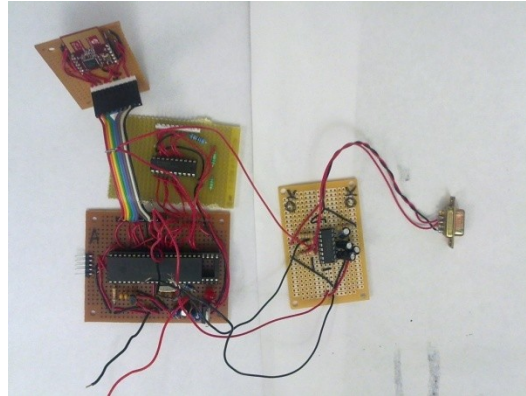
The successful demonstration of the Prototype System was followed by the Installed Floor consisting of 4 Installation Segments. Each Installation Segment contains 32 sensors arranged in 4 columns referred as A, B, C, and D, each having 8 sensors named as A1-A8, B1-B8, C1-C8, and D1-D8. A microcontroller and a wireless transceiver are associated with each Installation Segment making a total of 4 microcontrollers and 4 wireless transceivers in the Installed Floor (1).

The circuit consisting of a PIC18F4455 microcontroller, a MRF24J40MA wireless transceiver module and an Installation Segment is called a node. All the nodes communicate with each other through a wireless network. This wireless sensor network was built by connecting 4 nodes in linear fashion. The node connected to the computer is called the host node, while the node which initiates the transmission is called the end node, rest of the two nodes are called peer nodes. The end node connects to one of the unconnected peer nodes, which in turn connects to another unconnected peer node, which in turn connects to the host node.

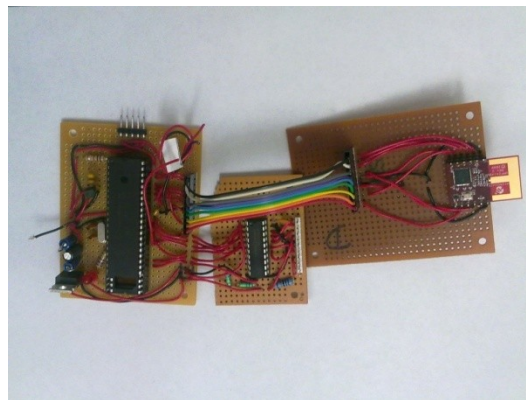


### 3.8.1. Data acquisition and conditioning for the Installed Floor

Figure 3-17 and Figure 3-18 show the pictures of circuit boards of the host node and peer/end node respectively. The host node processes the data and transfers the formatted data to the personal computer using RS-232. The hardware used for all other nodes (except the host) is exactly similar.



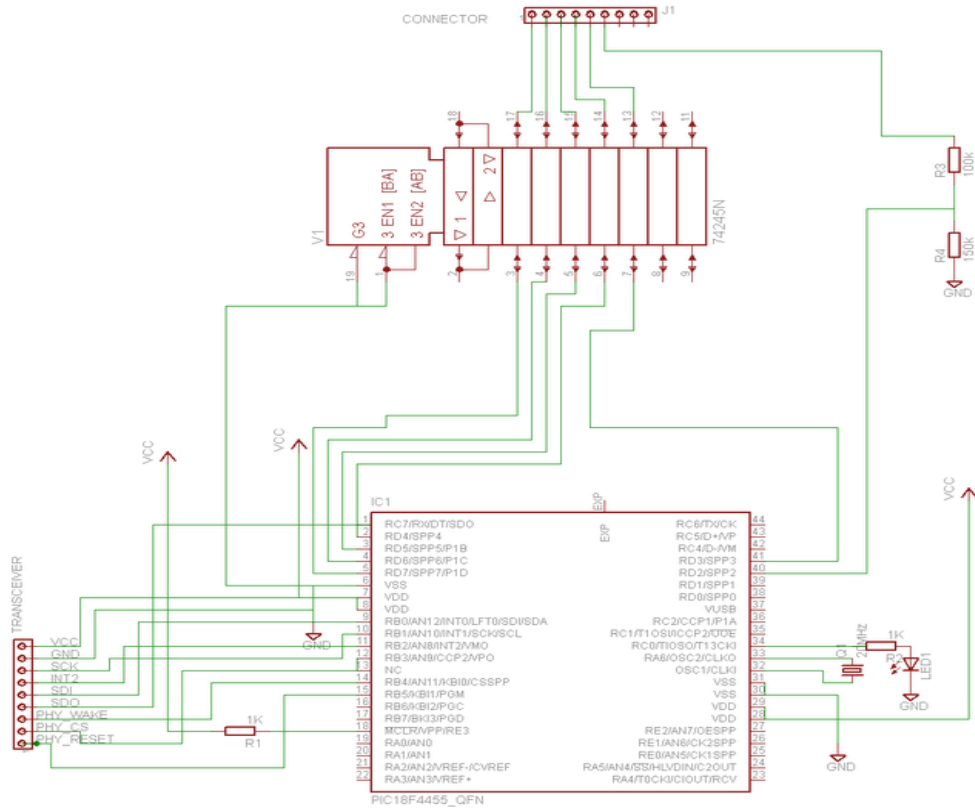
**Figure 3-17:** The picture of, the host node (the node connected to the personal computer) for the Installed Floor



**Figure 3-18:** The picture of peer node which transfer data to their master for the Installed Floor

The transceiver requires 3.3V for Vcc. To drive 5v select lines in the analog multiplexer we used a driver IC to convert 3.3v select lines to 5v (15). However, the 5v

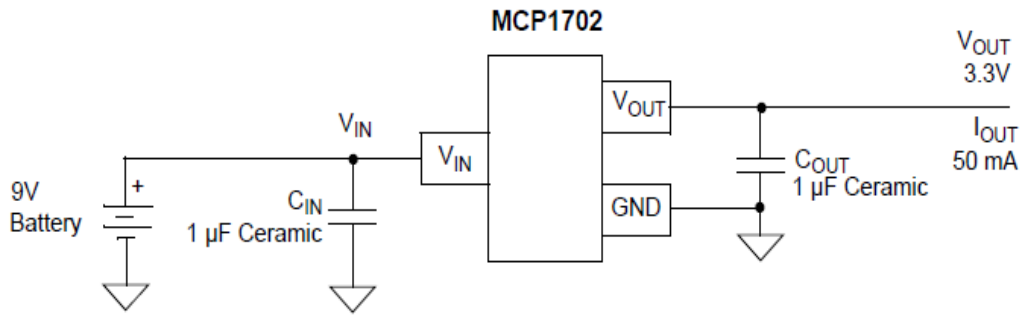




**Figure 3-20:** The schematic of microcontroller circuit implementation of peer nodes used in the Installed Floor

### 3.8.3. Voltage regulator circuit (3.3V) used in the Installed Floor

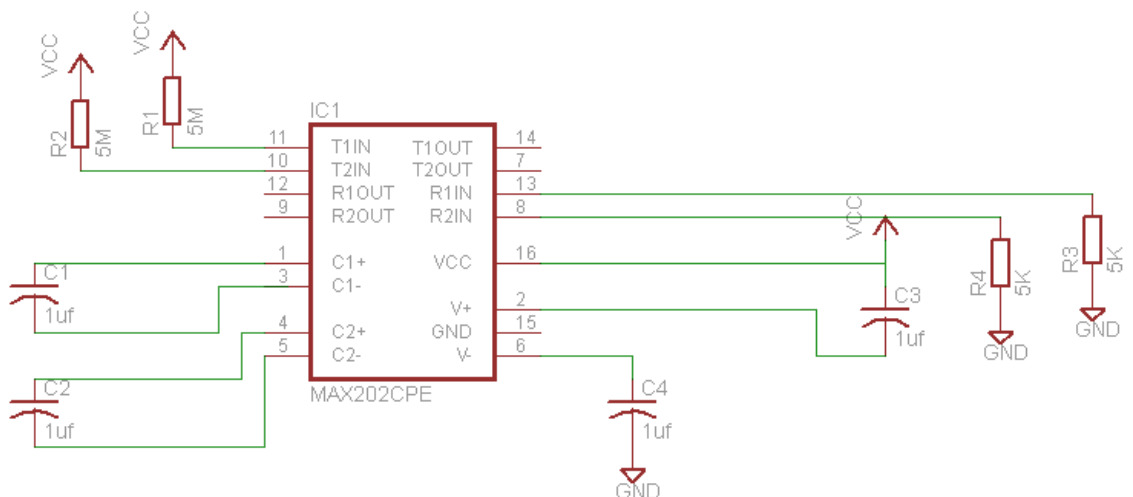
Figure 3-21 shows the typical application circuit of 3.3V regulator used for stepping down the voltage from 9V to 3.3V for supplying the microcontroller circuit as well as the radio transceiver module. The voltage regulator is different than that used in the Development System and the Prototype System, since they both use 5V Vcc (14).



**Figure 3-21:** The typical application circuit of 3.3V regulator used for stepping down the voltage to 3.3V for supplying the microcontroller circuit as well as the radio transceiver module (14).

### 3.8.4. Recommended Standard-232 (RS-232) Circuit used in the Installed Floor

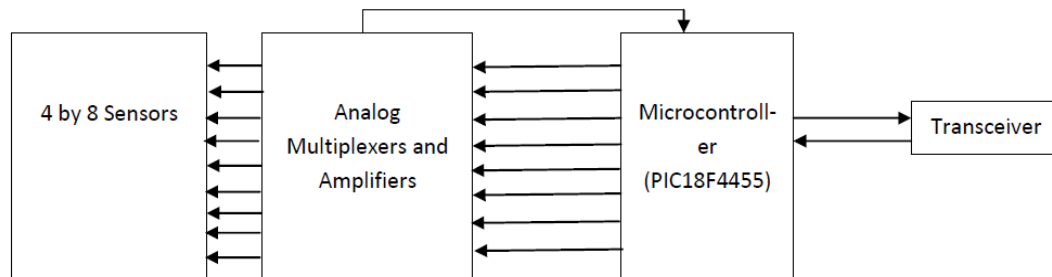
The Figure 3-22 shows the circuit diagram of Recommended Standard-232 (RS-232) which establishes a connection between the host node and the personal computer. This circuit is exactly similar to that used in the Development System and the Prototype System (17).



**Figure 3-22:** The circuit diagram of Recommended Standard-232 (RS-232) which establishes connection between the host node and the personal computer (17).

### 3.8.5. Installation Segment Electronics

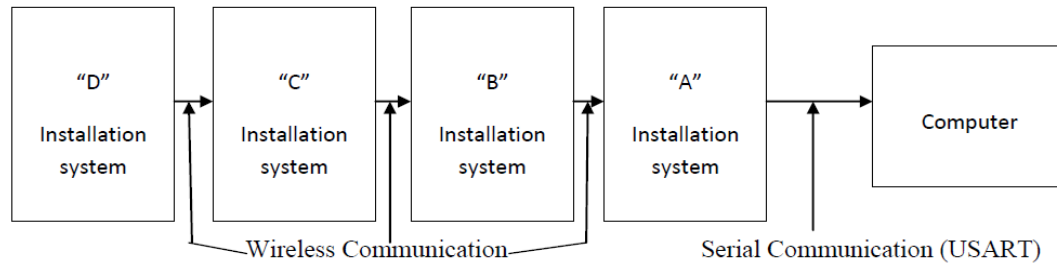
Figure 3-23 shows the block diagram of the electronic circuitry of an Installation Segment. Each Installation Segment Electronics consists of 32 sensors connected to input lines of four 8:1 analog multiplexers, while the outputs of the four 8:1 analog multiplexers are connected to the input lines of 4:1 analog multiplexer, and the output of 4:1 multiplexer is connected to the microcontroller input pins. Each select line of all the 8:1 multiplexer is connected to the microcontroller input pins. Each select line of all the 8:1 multiplexers are connected together and driven with 3 microcontroller signals and additionally 2 microcontroller signals drive select lines of 4:1 analog multiplexer. In summary, the microcontroller selects one of the four 8:1 analog multiplexers, and then selects one sensor at a time so that the software polls the output pin of a 4:1 multiplexer.



**Figure 3-23:** The block diagram of the Installation Segment Electronics used in the Installed Floor

### 3.8.6. Installed Floor Networking

Figure 3-24 shows the block diagram of installed floor in which the arrows show the direction of data flow in the network after establishing the network.



**Figure 3-24:** The block diagram of the Installation Floor consisting of 4 Installation Segment The arrows represent the direction of data flow in the network, from D to C, then C to B, then B to A and then from A to the computer.

In the Installed Floor, the node A is called the host node since it is connected to the computer; the remainders are called peer nodes. For establishing the wireless network steps below are followed,

1. The host node starts first, selects a least noise channel, and waits for another node to get connected on the selected channel.
2. Upon powering 'B' looks for another node in one of the 16 channels, 'B' finds and establishes a connection with 'A', saves it as a master, starts the timer, and waits for another node to get connected. After establishing connection with node B, the node A rejects further search requests from other nodes in the network.
3. After powering node C looks for a node in one of the 16 channels, finds node B, establishes a connection, saves it as a master, starts the timer and waits for another node to get connected. After establishing connection with C node B rejects further search requests from other nodes in the network.
4. After powering node D it looks for another node in one of the 16 channels, finds node C, establishes a connection, and saves it as a master, starts the timer and waits for another node to get connected. After establishing connection with D

node C rejects further search requests from other nodes in the network. After D is connected and timer has timed out it declares itself as an end node and starts the transmission of data to its master, node C.

Figure 3-24 also shows the sequence of data transmission. The node D sends data to node C, after receiving data from node D, node C transfers its own data and the data received from node D to, its master, node B. The node B sends its own data and the data received from C to its master, node A which is the host node. Then the host node processes the received data and sends it to the personal computer. The node D waits for 100ms before it again sends data to node C.

At the end of this waiting period the following has occurred.

1. The host node has finished formatting received data, has transferred data to the computer and is ready to receive data from its slave.
2. Every node has finished sending data to their respective master node.
3. Every node has finished polling all the sensors connected to it.
4. Every node is ready to receive data from previous node and also to transfer received data along with its own data.

### **3.8.7. Installed Floor Frame Formatting**

To transfer data received from the entire nodes the host node uses a frame format. At the end of the each polling cycle the host node receives 36 bytes of data from node B as follows. The data includes 8 address bytes and 4 data bytes from each node B, C, and D. After adding host node's address and data bytes it has a total of 48 bytes for sending to the personal computer. Furthermore, to differentiate data between different polling cycles

the host node adds ASCII character 'S' and 'E' at the beginning and end of each data frame so then it has 50 data bytes for the personal computer.

To reduce the transmission load, the host node replaces 8 address bytes of each node with one unique ASCII character, which reduces the transmission load from 50 to 22 bytes. Also the value of each data byte varies from 0x00h to 0xFFh but the computer works with only ASCII characters, which vary from 0x00h to 0x7Fh. The host node processes the data bytes and converts 16 data bytes into 32 ASCII characters.

In summary, after each polling cycle the host node sends 38 bytes consisting of 32 data bytes in ASCII form, 4 unique characters to differentiate the nodes and 2 characters to differentiate data from different polling cycles, to the computer.

The following section describes the steps followed by the program in the host node to processes the data received from its peer, put into a frame and sends it to the personal computer.

1. The program sends ASCII character 'S' to represent the start of a frame to buffer in the personal computer.
2. The program removes the 8 address bytes of the host node and sends ASCII character the installation segment (say the letter 'A') which causes the Installation Segment producing the data. For ease of programming the character A will always represent the host node.
3. Starting from 0<sup>th</sup> data byte, the program copies the 0<sup>th</sup> data byte to temp register, shifts temp register to the right by 4 bits to remove last four bits of the data byte, bitwise ORs it with hex value 0x30 to convert it to ASCII character and then send it to the computer via the serial port. Then again, the host program takes the 0<sup>th</sup>



data byte, ANDs it with hex value 0x0F to get rid of first 4 bits, ORs it with 0x30 to convert it to ASCII character, and then send it to the computer.

4. The program repeats the same steps mentioned above for 1, 2, and 3 data bytes to finish a segment.
5. Starting on the next segment, the host program sends ASCII character 'B' on the serial port to represent that following data will be of installed segment B, takes data received from its slave, installed segment B.
6. The program gets rid of the 8 address bytes of node B, takes 4 data bytes and again follows the same steps mentioned in 3 and 4 for sending data of installed segment B to the computer.
7. Then the host node program sends ASCII character 'C' on the serial port to represent that following data will be of installed segment C.
8. The program gets rid of the 8 address bytes of node C, takes 4 data bytes and again follows the same steps mentioned in 3 and 4 for sending installed segment C's data to the computer.
9. Then the host node program sends ASCII character 'D' on the serial port to represent that following data will be of installed segment D.
10. The program gets rid of 8 address bytes of node D, takes 4 data bytes and again follows the same steps mentioned in 3 and 4 for sending node D's data to the computer.
11. The host node sends ASCII character 'E' to represent the end of the data frame.

Figure 3-25 shows the flow chart of program written for the host node. It also includes the instructions mentioned above.

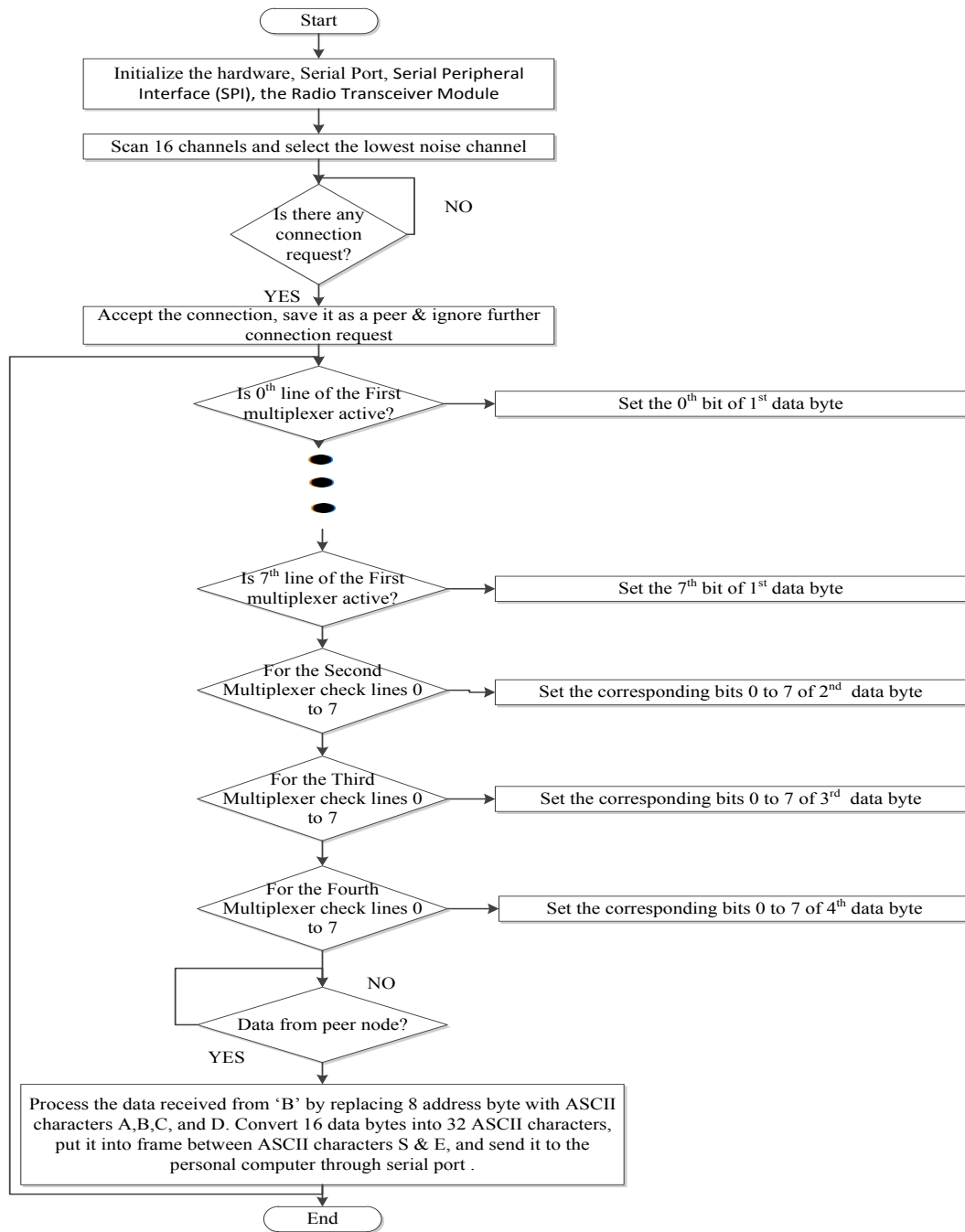
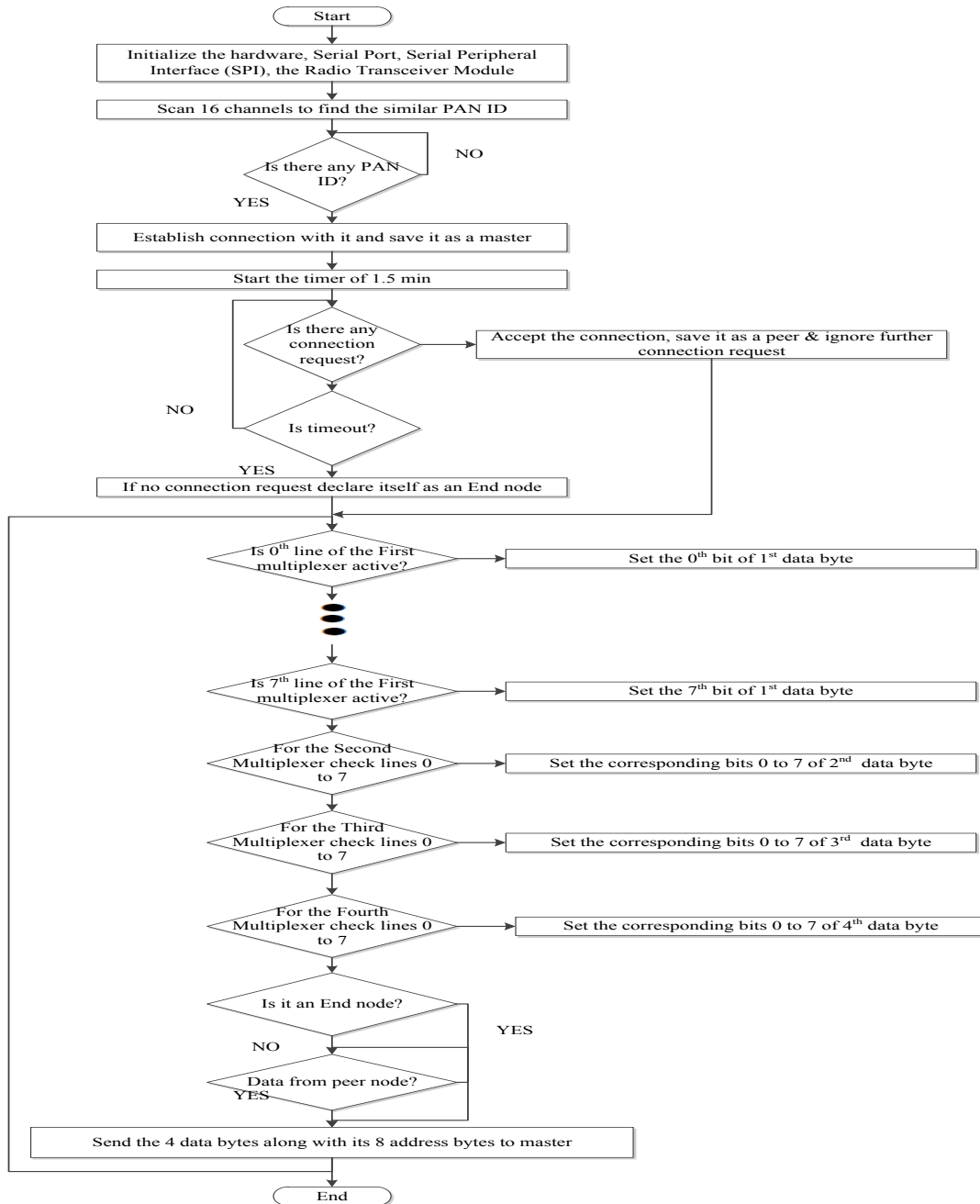


Figure 3-25: The flow chart of program written for the host node

Figure 3-26 shows the flow chart of the program written for the peer nodes which excludes the instructions followed for processing and transferring the data to the computer.



**Figure 3-26:** The flow chart of program written for the peer nodes which excludes the instructions followed for processing and transferring the data to the computer.

### **3.8.8. Experimental set up for the Installed Floor**

Three tests were carried out to assess the functioning of the system. The setup was similar to those for the Development System and the Prototype System i.e. a supply, microcontroller system and computer but with an addition of Oscilloscope.

#### **3.8.8.1. Observation of sensor non-activation in the Installed Floor**

All 32 input pins from all the Installation Segments were directly connected to ground and output data was taken on the Windows Hyper Terminal. The computer display was observed after keeping the same input signal on microcontroller's input pins after running the software (2).

#### **3.8.8.2. Observation of sensor activation in the Installed Floor**

All 32 input pins from all the Installation Segments except the one connected to 'A1' from each installation system were directly connected to ground, pins 'A1' from each I S was connected to 5v supply and output data was taken on the Windows Hyper Terminal. The computer display was observed after keeping the same input signal on microcontroller's input pins after running the software (2).

#### **3.8.8.3. Observation of the Installed Floor data acquisition per second**

The time to scan all the sensors and to send data to the computer was carried out by writing new code in host node to change the status of an output pin from high to low and vice versa every time it forwards data to the computer. The output of that pin was seen on an oscilloscope to calculate the time. This test was carried to know the number of times all the sensors are polled per second.







**Figure 4-4:** Sensor display program when 2 sensors were activated (2)

### **4.1.3. Observation of the Development system data acquisition and transmission time**

A test was carried out using 3 baud rates (19200, 57600, and 115200). We counted the number of pulses in a single poll to show the time taken by the microcontroller to perform a single complete polling cycle. The personal computer received the data via a serial port. Since the microcontroller timer increments after 4 clock Cycles, at 20 MHz frequency the timer increments, we get 5 million pulses per second.

For example, for activation of sensors ‘A’ and ‘B’ we get S12E10343 on the Windows Hyper Terminal. This means that the timer in the microcontroller produced 10343 pulses for one polling cycle which in turn means it took 2. 0686ms for the microcontroller to poll all the sensors and transfer the formatted data to the computer via

the serial port at 19200 baud. This same data is shown on 3<sup>rd</sup> row of 2<sup>nd</sup> column in Table 4-1

Similarly, the Table 4-1 shows the time to complete one polling cycle and transfer data to the personal computer. The first column of the Table 4-1 describes the total number of active sensors read by the microcontroller; the next three columns describe the time for data acquisition and transmission in ms, for using 3 baud rates (19200, 57600, and 115200).

**Table 4-1:** The time taken by Development system to complete one polling cycle and transfer data to the computer via serial port. First column of table describes the total number of active sensors read by the microcontroller while other three columns describe the data acquisition and transmission time in ms, for using 3 baud rates (19200, 57600, and 115200).

<b>Active Sensors</b> <sup>1</sup>	<b>For 19200 baud</b> <sup>2</sup>	<b>For 57600 baud</b> <sup>3</sup>	<b>For 115200 baud</b> <sup>4</sup>
0	1.0284	0.3408	0.1644
1	1.4288	0.5162	0.2522
2	2.0686	0.6928	0.3406
3	2.5884	0.8688	0.4284
4	3.1082	1.0448	0.5168

<sup>1</sup> Total number of active sensors read by the microcontroller system.

<sup>2</sup> Microcontroller data acquisition & transmission time in ms, using 19200 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

<sup>3</sup> Microcontroller data acquisition & transmission time in ms, using 57600 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

<sup>4</sup> Microcontroller data acquisition & transmission time in ms, using 115200 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

Figure 4-5 shows graphical representation of Similarly, the Table 4-1 shows the time to complete one polling cycle and transfer data to the personal computer. The first column of the Table 4-1 describes the total number of active sensors read by the microcontroller; the next three columns describe the time for data acquisition and transmission in ms, for using 3 baud rates (19200, 57600, and 115200).



Table 4-1. The 3 different baud rates are multiples of each other and show the reduced acquisition time as a result of increasing the baud rate. For our purpose, the increase in baud rate did not affect the low persistence display, neither was it apparent that the data was lost, indicating a substantial margin for the time to acquire and transmit data.

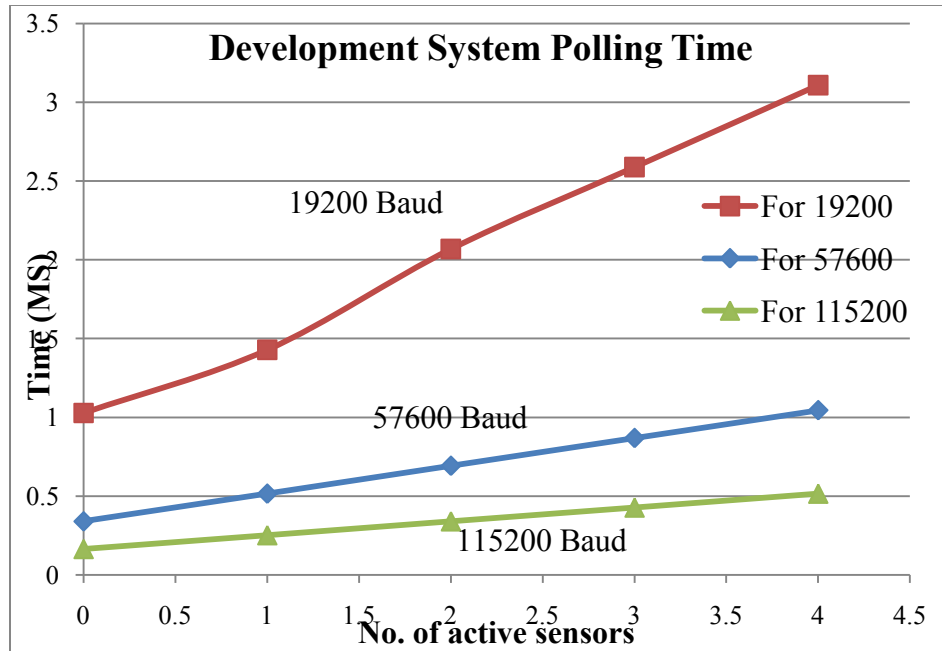


Figure 4-5: Graph for Time in mS Vs No. of Active sensors

## 4.2. Prototype system

### 4.2.1. Observation of sensor non-activation in the Prototype System

The tests taken were similar to test taken in case of the Development S The setup consisted of power supply, microcontroller system and computer, where for an operational purpose, we used the power supply to provide 5v on activated sensor. The Figure 4-6 shows the data received on Windows Hyper Terminal when no sensor was





means that the timer in the microcontroller produced 18150 pulses for one polling cycle which in turn means it took 3.63ms for the microcontroller to poll all the sensors and transfer the formatted data to the computer via the serial port at 19200 baud. This same data is shown on 3<sup>rd</sup> row of 2<sup>nd</sup> column in Table 4-2

Table 4-2 shows the time to complete one poll cycle and transfer data produced by the number of active sensors read by the microcontroller, as shown in 1<sup>st</sup> column, the other three columns describe the data acquisition and transmission time (ms), for 3 baud rates (19200, 57600, and 115200).

**Table 4-2:** The time taken by each system to complete one polling cycle and transfer data to the computer via serial port. First column of table describes the total number of active sensors read by the microcontroller while other three columns describe the data acquisition and transmission time ms, for using 3 baud rates (19200, 57600, and 115200).

<b>Active Sensors</b> <sup>1</sup>	<b>For 19200 baud</b> <sup>2</sup>	<b>For 57600 baud</b> <sup>3</sup>	<b>For 115200 baud</b> <sup>4</sup>
0	2.59	0.87	0.43
1	3.11	1.04	0.52
2	3.63	1.22	0.60
3	4.15	1.40	0.69
4	4.67	1.57	0.78
5	5.12	1.75	0.87
6	5.71	1.92	0.95
7	6.23	2.10	1.04

<sup>1</sup> Total number of active sensors read by the microcontroller system.

<sup>2</sup> Microcontroller data acquisition & transmission time in ms, using 19200 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

<sup>3</sup> Microcontroller data acquisition & transmission time in ms, using 57600 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

<sup>4</sup> Microcontroller data acquisition & transmission time in ms, using 115200 baud, for single polling of all the sensors and when numbers of sensors in <sup>1</sup> were active.

The Figure 4-10 shows graphical representation of Table 4-2. Similar to the Development System as expected the 3 baud rates show the reduced acquisition time as a

result of increasing the baud rate in multiple of each other. As before this demonstrated a substantial margin for a number of polled sensors.

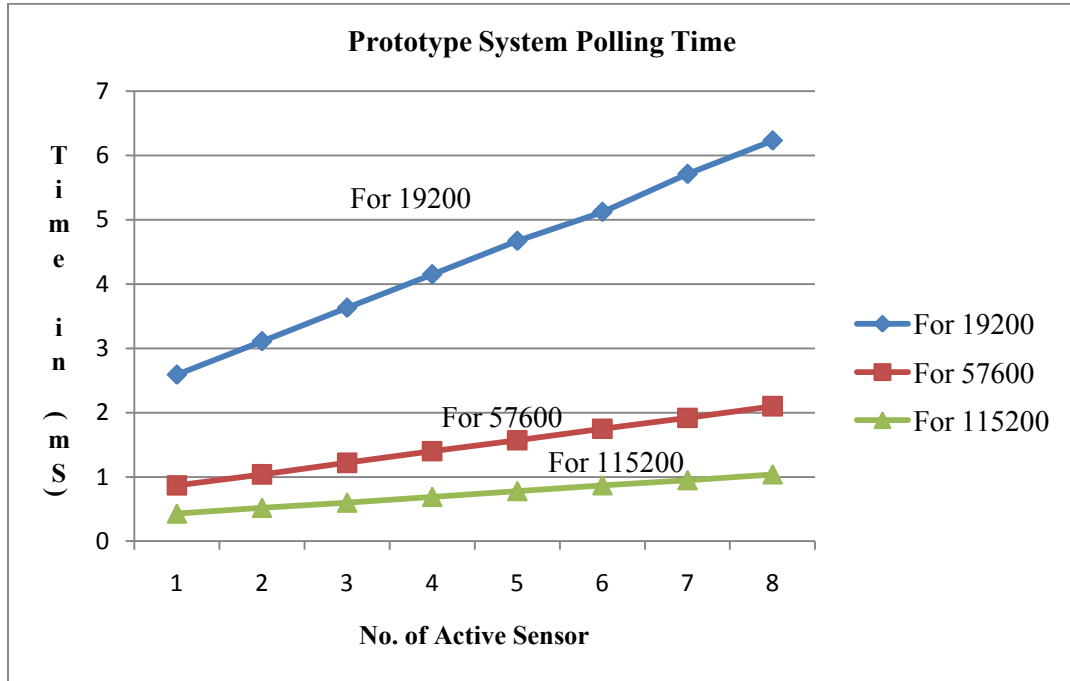


Figure 4-10: Graph for Time in ms Vs No. of Active sensors

### 4.3. Installed floor

#### 4.3.1. Observation of sensor non-activation in the Installed floor

The tests taken for the installed floor were similar to that of the Development System and the Prototype System. The setup consisted of power supply, microcontroller system and computer, where for an operational purpose, we used the power supply to provide 5v on activated sensor. Figure 4-11 shows the data received on Windows Hyper Terminal when no sensor was active. ‘S’ and ‘E’ represent the start and end of a frame

but A, B, C, and D represent the 4 installation segments each consisting of 32 sensors. The 8 0s after A, B, C and D show that none of the sensors were active.

```
SA0000000000B000000000C000000000D00000000ESA00000000B00000000C00000000D00000000ESA00
000000B000000000C000000000D00000000ESA00000000B00000000C00000000D00000000ESA000000
00B000000000C000000000D00000000ESA00000000B00000000C00000000D00000000ESA00000000B0
```

**Figure 4-11:** Data received on the Windows Hyper Terminal from microcontroller when no sensor was active. ‘S’ and ‘E’ represent the start and end of frame received respectively while A, B, C, and D represent the 4 installation systems. The 8 0s after A, B, C and D show that none of the sensors were active.

As we will show in the next section this is a consequence of the logic operations, we used to represent the reduced data.

Figure 4-12 shows the screen capture of computer display. The letters A1-A8, B1-B8, C1-C8 and D1-D8 represent the 32 sensors on each installation system. Furthermore, the 4 installation systems are visually differentiated using different colors and thick line dividers (2).



**Figure 4-12:** The Sensor display program when no sensor was activated. The letters A1-A8, B1-B8, C1-C8 and D1-D8 represent the 32 sensors on each installation system. Also the 4 installation systems are visually differentiated using different colors and thick line divider. (2)



We demonstrate the mapping for segment A and specifically for the excitation of A1. To get A01000000 required the following operations as explained in **3.10.7** and the code in appendix. The original byte of excitation data corresponding to say segment A in Figure 4-14 is  $\text{Data\_byte}[0] = 00000001$  where the most significant and least significant bit represent the status of sensors A8 and A1, we show the explicit operation.

$\text{Data\_byte}[0] = 00000001\text{b}$

$\text{Temp\_reg} = 00000001\text{b}$

4 times Right shift of  $\text{Temp\_reg}$  causes  $\text{Temp\_reg} = 00000000\text{b}$

$\text{Temp\_reg} \text{ OR } 0\text{x}30\text{h} = 00000000\text{b} \text{ OR } 00110000\text{b} = 00110000\text{b} = 0\text{x}30\text{h}$ , which is ASCII character '0'

$\text{Data\_byte}[0] = 00000001\text{b}$

$\text{Data\_byte}[0] \text{ AND } 0\text{x}0\text{Fh} = 00000001\text{b} \text{ AND } 00001111\text{b} = 00000001\text{b} = 0\text{x}01\text{h}$

$\text{Data\_byte}[0] \text{ OR } 0\text{x}30\text{h} = 00000001\text{b} \text{ OR } 00110000\text{b} = 00110001\text{b} = 0\text{x}31\text{h}$ , which is ASCII character '1'

So we have 01 followed by 'A'

Similarly if at Segment A, Column C, say we excite foils C2, C4, C5 and C7 so we get

$\text{Data\_byte}[0] = 01011010\text{b}$

$\text{Temp\_reg} = 01011010\text{b}$



4 times Right shift of Temp\_reg causes Temp\_reg = 00000101b

Temp\_reg OR 0x30h = 00000101b OR 00110000b = 00110101b = 0x35h, which is ASCII character '5'

Data\_byte[0] = 01011010b

Data\_byte[0] AND 0x0Fh = 01011010b AND 00001111b = 00001010b = 0x0Ah

Data\_byte[0] OR 0x30h = 00001010b OR 00110000b = 00111010b = 0x3Ah, which is ASCII character ':'

So we have ':5'

This is transmitted as A0000:500

### **4.3.3. Observation of data acquisition per second**

A test was carried out for observing the time taken to poll all the sensors and send data to the computer via the serial port. A program was written in the host node to change an input output pin status from high to low and vice versa whenever it finishes sending data to the computer. An oscilloscope was connected at that pin. The reading taken was 132ms. We can calculate the number of time the whole Installed Floor polled per second is  $(1000\text{ms}/132\text{ms}) = 7.5$  folds. There is a sufficient time to acquire the sensor data several times each second. This 8 folds margin allows us to scale the numbers of sensors by at least a factor of 2.

## **CHAPTER 5 : DISCUSSIONS**

In this chapter, we will be discussing the microcontroller, the software, the hardware as well as the protocols used for the development, prototype systems and the installed floor.

### **5.1. Microcontroller and software**

There are plenty of other microcontrollers available in the market provided by different manufacturers which would be also adequate for this application. We chose PIC microcontroller because of prior knowledge and the knowledge of PIC microcontroller made our development relatively easier. Also we switched from assembly language to Embedded C because it made the code more readable and intuitive. Moreover, the next team of developer will not require having knowledge of assembly. Furthermore, the cross compilers for Embedded C is available for free on Microchip website.

### **5.2. Hardware used for the development, prototype and the installed floor**

In the early stage, the development system required only 4 sensors. The PIC16F871 microcontroller had adequate number of input pins to connect to the outputs of amplifiers directly. This made program writing and testing fairly straightforward and we could read and display the sensor data on the personal computer.

In the prototype system, the number of sensors increased from 4 to 21, we could have managed to connect 21 sensors to the PIC16F871 microcontroller but it would have made an unnecessary complex circuit. Furthermore, from the future point of view, it would have been difficult to expand the number of sensors in the same circuit. We used a digital multiplexer to handle each column of 7 sensors. We used 3 digital 8:1 multiplexers to connect 21 sensors. The 3 digital multiplexers gave us a scope for connecting 24 sensors and the output of the multiplexers required a buffer to connect to the microcontroller. This in turn, increased the hardware but with lesser complexity and easier implementation. If we could find multiplexers with built in buffers, then we can reduce the hardware. Also, we used 9 pins of microcontroller for connecting it to the 9 select lines of 3 8:1 multiplexers. If we had shorted all the select lines together and used only 3 output pins of microcontroller to drive them then, we would have simplified the circuit little more and also the code size would have been decreased a lot, saving the code memory.

Furthermore, it would have been better if we could find a 16:1 multiplexer. Then we would need only 2 multiplexers (one 16:1 multiplexer and one 8:1 multiplexer) which will reduce more hardware. Also, because of using 2 multiplexers some of the input pins of microcontroller will be unused. This will give us an opportunity to connect more than 21 sensors through greater than 3 8:1 multiplexers. Had the prototype system been our goal, clearly we should have made the improvements proposed here.

Here we went directly to the installed floor. Installed floor was lot different than previous systems. We needed wireless capability along with the support for connecting

more sensors. For connecting more sensors Rohan Neelgund implemented 32:1 analog multiplexer which simplified my work a lot (1). To implement the wireless network, we used a radio transceiver module. The radio transceiver modules should be kept at least 1ft apart from each other. Moreover, the radio transceiver module works on 3.3v, we operated microcontroller using 3.3v supply. However, the analog multiplexer operates on 5v so we had to use a driver to convert the microcontroller select lines from 3.3v to 5v and a voltage divider was used to give the output of multiplexer to the PIC18F4455 microcontroller, which in turn increased hardware and complexity of the circuit. we could operate the analog circuit using 3.3v supply then it will reduce the hardware and complexity of the circuit(1).

### **5.3. Protocols used in the development, prototype systems and the installed floor**

For development system, the data sent to the computer was ranging from 0 to 4 bytes. To understand when one polling cycle is over and to differentiate data from different polling cycles, we decided to define a simple frame with the addition of symbols for a start (ASCII character of S) and end (ASCII character of E) of a frame. We chose to use ASCII characters of letters S, 1, 2, 3, 4 and E to make it easy to understand after seeing the received data on the personal computer. There could be many ways of defining the frame, but we used this format since it satisfied our requirements. We made sure that data related to active sensors won't conflict with ASCII characters 'S' and 'E'. We were able to test this using Windows hyper terminal.

As we moved on to the prototype system, there was an increase in a number of sensors from 4 to 21. We tried using 21 different ASCII characters for representing each sensor on the system this made it difficult to read the data received on the personal computer so to make it easier to understand and more intuitive we used similar format as that of the sensor board i.e. 3 columns each of 7 sensors. We used the ASCII characters of letters A, B and C in the frames to represent columns and ASCII characters 1-7 to represent each sensor in one column. Each sensor we then identified by the column and sensor number (e.g. A5). We also added ASCII characters S and E at the start and the end respectively in each frame to separate the polling cycles.

For the installed floor, we built a daisy chain of 4 installed segments; we thought of keeping the same frame format. This time there were 128 sensors if we used same frame format then there would have been 1 data byte for each sensor making a total of approximately 128 bytes data bytes for whole installed floor. Furthermore, each wireless node sends its own 8 address bytes making a total of 32 address bytes for 4 nodes. So for one polling cycle the host node would have sent more than 160 bytes to the computer. This was a lot of data, which would have increased the acquisition and transmission time. To reduce the number of data bytes, we decided to represent the status of a single sensor as a bit in an 8 bit map for each column this reduced the number of data bytes from 128 to 16 bytes. However, as the personal computer can understand only ASCII characters, which range from hex 0x00 to 0x7F (decimal 0 to 127) we decided to process the data for the personal computer. The host node is connected to the personal computer so it does bit manipulation and converts the 16 byte data to 32 byte ASCII characters and sends it to

the personal computer. This made the total of 64 bytes, which consisted of 32 data bytes and 32 address bytes. This made the data received on the personal computer little less intuitive but with a reduced number of bytes. There was an overhead of 32 address bytes, so we decided to further process the data and replace the 32 address bytes with ASCII characters A, B, C, and D which in this system were representing each of installation segments in the Installed Floor sending 38 bytes to the personal computer.

The addresses of nodes can be kept minimum 2 bytes long and maximum 8 bytes long. In our system, we have used 8 bytes long addresses but keeping the addresses 2 bytes long will make the system little faster. Because of reduction in the bytes, the transmission among the nodes will be faster also the processing of data at the host node will be faster. Moreover, each microcontroller in the network waits for 1ms after selecting each sensor through select lines of analog multiplexers; by modifying the delay routine in the code we could wait for a lesser amount of time speeding the system further. In addition to that node D which is the end node in the network, initiates the data transmission and after transferring its data to node C it waits for a programmable 100ms before it starts polling the sensors. After finishing waiting time, node D takes 32ms to poll the sensors and then again transfers its data to node C. We can program less waiting time for a faster speed. Speculating that we reduce the polling and waiting time by 50%, all the 128 sensors will be scanned approximately 15 times per second which in turn will increase the data acquisition per second.

A star network which is easier to implement and can handle 4 installed segments was possible but the star network is not appropriate for covering larger area of larger length, where as the linear system gives larger length or reach. The linear system was

more suitable for our application. Moreover, the star network is more robust while in the daisy chain network if anyone of the nodes goes down the whole network goes down.

## CHAPTER 6 : CONCLUSION

We successfully built and tested the microcontroller circuitry for the Development System, the Prototype System as well as the Installed Floor. Acquisition time is important, and we show that increasing the baud rate for polling and transferring data speeds up the system linearly. In the Development and the Prototype Systems, the highest data polling and acquisition times, no sensors were active and at 19200 baud rate came in at 1ms and 2.59ms, while the lowest time for 115200 baud rate came in at 0.16ms and 0.43ms. The increase in baud rate did not affect the appearance of the display, neither was the data lost, which indicates that the system has a significant margin for the time to acquire and transmit data. In case of the Installed Floor, the microcontrollers efficiently communicate with each other through a peer-to-peer wireless network. All the systems displayed appropriate sensors, corresponding to the activated sensors. Further work indicating that the systems are reasonably accurate and lossless. The low price of the electronic parts confirmed the low cost of the systems. The circuitry and the algorithms can be used for future expansions with minor modifications; the same hardware and the algorithm architecture can be used for similar products using different sensors.

From the Table 2-1 it is apparent that P87C52SBPN (not used) has a lower cost than the PIC18F4455 microcontroller (used for the Installed Floor). By utilizing the Serial Peripheral Interface port and making appropriate changes in the software functions used for interfacing the radio transceiver module, the P87C52SBPN microcontroller can



replace PIC18F4455 microcontroller (8). Moreover, P87C52SBPN has more input/output pins, 32 pins. These pins can be utilized to connect larger number of sensors, which in turn will reduce the number of nodes required per room, and thus improve the scalability of the system. In summary, by replacing PIC18F4455 with P87C52SBPN we can cover a bigger area using an adequate number of microcontrollers and hence reduce the cost of the system (6).

All the microcontrollers mentioned in Table 2-1 except PIC16F871 support Universal Serial Bus (USB) interfacing. USB has the potential for higher speed. The increase in data acquisition rate speeds up the data communication for the display computer. Future implementation will benefit by using USB to communicate with the computer. Additionally, the serial port is gradually being replaced by the USB, the USB has the potential for higher speed, and hence to increase the data acquisition rate of the personal computer.

In the electronics of the microcontroller as well as the software there is good scope for improvement. Use of surface mountable ICs instead of PDIP packages can reduce the size of the electronics. A single box containing electronics associated with microcontroller could be powered up using a single voltage adapter. The use of a standard version of the compiler (instead of student or lite versions which we used) will reduce the code size reducing the code memory space required. Of course the standard versions are not freely available.

## REFERENCES

1. Rohan Neelgund. Floor sensor development using signal scavenging for personnel detection system. *Computer Engineering, University Of Missouri, Columbia. 2010*. Masters thesis for fulfilment of MS degree.
2. Krishna Devarkonda. Data display for a signal scavenging personnel detective system. *Computer Engineering, University of Missouri, Columbia 2010*. Masters thesis for fulfilment of MS Degree.
3. Harry W. Tyrer, Rohan Neelgund, Uday Shriniwar, KrishnaKishor D. Faux-Floor Development System for Personnel Detection Using Signal Scavenging Sensors. *32<sup>nd</sup> Annual International IEEE EMBS Conference, Argentina, August 31-September 4, 2010* Under review.
4. Atmel Corporation, ATmega16A datasheet, Document number: 8154B–AVR–07/09, Atmel Corporation 2325, Orchard Parkway San Jose, CA 95131 USA.
5. Freescale Semiconductor, MC68HC908AB32 — Rev. 1.1, Technical Data, Freescale Semiconductor, Email: support@freescale.com, August 2, 2005
6. Philips Semiconductors, P87C52SBPN, Product specification, Philips Semiconductors 811 East Arques Avenue P.O. Box 3409 Sunnyvale, California 94088–3409, 2000 Aug 07
7. Alzheimer's Association, Alzheimer's Association National Office 225 N. Michigan Ave., Fl. 17, Chicago, IL 60601, 24/7 Helpline: 1.800.272.3900. [http://www.alz.org/alzheimers\\_disease\\_alzheimers\\_disease.asp](http://www.alz.org/alzheimers_disease_alzheimers_disease.asp)
8. Microchip Technology Inc., MRF24J40MA the Radio Transceiver module, Advance Information, Document number: DS70329A, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199

9. Microchip Technology Inc., MiWi(TM) IEEE 802.15.4 Wireless Networking Protocol Stack, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2009
  
10. Microchip Technology Inc., MPLAB C18 for PIC18 v3.35 in LITE mode, Part Number: SW006011, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2009
  
11. Microchip Technology Inc., MPLAB IDE v8.50 Full Release Zipped Installation, Part Number: SW007002, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2009
  
12. Microchip Technology Inc., PICkit™ 2 Development Programmer/Debugger, Part Number : PG164120, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2009
  
13. Microchip Technology Inc., PICkit™ 2 Microcontroller Programmer USER'S GUIDE, Document number: DS51553E, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2008
  
14. Microchip Technology Inc., MCP1702 datasheet, Document number: DS22008B, Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, AZ 85224-6199, 2007
  
15. Fairchild Semiconductor Corporation, MM74HC245AN datasheet, Document number: DS005165, [www.fairchildsemi.com](http://www.fairchildsemi.com)
  
16. Dementia Education & Training Program, PREVENTION OF FALLS IN THE DEMENTIA PATIENT, 1-800-457-5679.
  
17. National Semiconductor Corporation, DS14C232 Datasheet, Document number: DS010744, [www.national.com](http://www.national.com)

## APPENDICIES

Following appendices are provided in different document

**Appendix A** – Program written for the Development System

**Appendix B** – Program written for the Prototype System

**Appendix C** – Modified HardwareProfile.c to make the program work for PIC18F4455 in case of the Installed Floor

**Appendix D** – Modified FeatureDemoNode1.c of P2P Node 1 in as per our requirements for the host node

**Appendix E** – Modified FeatureDemoNode2.c of P2P Node 2 as per our requirements for the peer nodes

**Appendix F** – Pages from datasheet of PIC16F871 used for the Development System and the Prototype System

**Appendix G** – Datasheet of SN7417N used in the Prototype system

**Appendix H** – Pages from datasheet of PIC18F4455 used for the Installed Floor

**Appendix I** – Datasheet of MRF24J40MA Datasheet used in the Installed Floor

**Appendix J** – Datasheet of MCP1702-3302 Datasheet used in the Installed floor

**Appendix K** – Datasheet of DS14C232 used in all the systems