**ABSTRACT**

We used a mix of pressure and temperature sensors to design a wireless seat monitoring device that will be able to detect a person in a seat. The sensors are connected to a microcontroller PIC that performs ADC conversion. A transmitter then takes this information and wirelessly transfers this data to a wireless receiver that is connected to a server that will have the ability to determine which seats are full .

**1. INTRODUCTION**

**1.1 Purpose**

Our goal was to design a system that would make seat monitoring more convenient. We used a combination of temperature and pressure sensors to detect if a person is sitting on the seat or not. The sensors are connected to a microcontroller PIC that performs ADC conversion. A transmitter then takes the information and wirelessly transfers this data to a wireless receiver that is connected to a server that will have the ability to determine which seats are full. Our system would be useful in a variety of applications such as the library, lecture hall, movie theatre, or airplane (just to name a few.) In a lecture hall setting, the seat detection information will help people make decisions such as lowering or raising the number of copies to make for distribution. In a library setting, students could first check and determine whether they should make a trip into the library. If used in a movie theatre, a screen could be implemented outside the theatre gates so customers could see what seats were open and thus not have to wander around aimlessly in the dark looking for seats. If implemented on an airplane, flight attendants would not have to take the time to make sure the passengers are seated during takeoff or landing (note: the system could also be modified to also check if seat belts are fastened.)

**1.2 Project Functions**

Our seat monitoring device offers the following benefits and features:

*Benefits& Features:*

\_ Easy and efficient to use

\_ Time-saver

\_ Easy maintenance

\_ Wireless transmission of seat availability data

\_ End users can check seat availability data using a computer.

\_ Can be modified for a variety of different applications (lecture hall, library, movie theater)

In addition, our system offers the following performance requirements:

\_ It can accurately determine the presence of a human sitting on the chair

\_ The transmitter can send the correct information to the receiver within a 300 feet radius

**1.3 Blocks**

Please refer to Diagram A.1 in the Appendix A section.

Seat Cushion – The temperature and pressure sensors are placed inside a seat cushion in order to increase device mobility.

Temperature Sensor – The temperature sensor senses the temperature to determine if a person is sitting on the chair. The threshold temperature is set at around 30 °C, which is near body temperature. The information from the temperature sensor is then sent by wires to PIC.

Pressure Sensor – The pressure sensor senses the pressure to determine if a person is sitting on the chair. if a person is sitting on the chair for 5 second and the sensor have a voltage reading then this mean there is somebody on the chair .

PIC #1 – This component is the microcontroller that takes the output of the two sensors and determines if a person is sitting on the chair. Once the chair is activated, the information will be sent by wires to the wireless transmitter.

Wireless Transmitter – The wireless transmitter component takes the output of PIC #1 and wirelessly sends the information to the wireless receiver.

Wireless Receiver – The wireless receiver takes the data sent by the wireless transmitter and outputs the information to PIC #2.

PIC #2 – This component deciphers the information sent by the receiver and outputs the information to the server. The RS-232 protocol will be implemented on PIC #2 in order to interface with the computer.

Server – The server component takes all of the information and I will see the result using visual basic program.

**2. DESIGN PROCEDURE**

**2.1 Temperature Sensor**

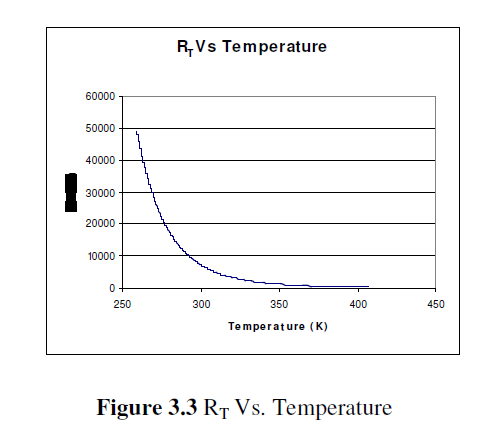
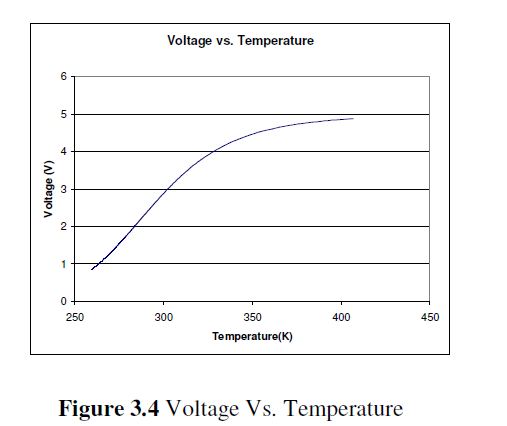
We decided to use a thermistor (thermally-sensitive resistor) for the temperature sensor design because they have low power consumption, fast response time, simple two-wire connection, ruggedness, high sensitivity to low temperature changes, and low cost. We learned that there are actually 2 types of thermistors: NTC and PTC.

** **

**Figure 2.1** R Vs. T graph for **Figure 2.2** R Vs. T graph for

NTC thermistor PTC thermistor

A Negative Temperature Coefficient (NTC) thermistor decreases in resistance when temperature increases, while the Positive Temperature Coefficient (PTC) thermistor increases in resistance when temperature increases. The NTC thermistor is more suited for temperature measurements because it exhibits steep resistance change as temperature increases.

Initially, we were going to use the LM235Z temperature sensor, which is a temperature sensor that operates as a 2-terminal Zener diode. We connected the sensor in the configuration shown below, touched the sensor with our fingers, and measured the output voltage with an oscilloscope.

**Figure 2.3** LM235Z Circuit Diagram **Figure 2.4** Voltage output graph of LM235Z

As one can see, the output voltage change from the LM235Z temperature sensor was very small, so it was not sensitive enough for our seating monitoring application.

**2.2 Pressure Sensor**

We decided to use ***FlexiForce sensor***

The ***FlexiForce*** sensors use a resistive-based technology. The application of a force to the active sensing area of the sensor results in a change in the resistance of the sensing element in inverse proportion to the force applied. The sensor acts as a variable resistor in an electrical circuit. When the sensor is unloaded, its resistance is very high (greater than 5 Meg-ohm); when a force is applied to the sensor, the resistance decreases. Connecting an ohmmeter to the outer two pins of the sensor connector and applying a force to the sensing area can read the change in resistance. The ***FlexiForce*** sensor is an ultra-thin and flexible printed circuit, which can be easily integrated into most applications. With its paper-thin construction, flexibility and force measurement ability, the ***FlexiForce*** force sensor can measure force between almost any two surfaces and is durable enough to stand up to most environments. ***FlexiForce*** has better force sensing properties, linearity, hysteresis, drift, and temperature sensitivity than any other thin-film force sensors. The "active sensing area" is a 0.375” diameter circle at the end of the sensor.



**2.3 PIC**

The microcontroller is the “brains” behind our project, handling communications between the sensors and the transmitter and between the receiver and PC. We decided to use the PIC16F877A from Microchip because it was readily available in the ECE Part Shop and because we found a number of PIC tutorials on the ECE 445 course website. We were also pleased with the fact that the PIC16F877A has a built-in ADC conversion that we could use for the sensors. The 16F877A has three main options for device communications: RS-232, SPI, and I2C. While each of these protocols has their own advantages and disadvantages, we decided to use RS-232 to implement our communications. The SPI and I2C options can support multiple devices at once, but since at a given time, our PIC only needs to communicate between the sensors and transmitter or between the receiver and PC, we decided that RS-232 was sufficient enough for our application.

We also found out that RS-232 has better noise immunity than SPI, especially over long distances. Noise immunity is very crucial aspect in our project because communication errors will lead to incorrect seat detection.

1. Pic#1:

This component is the microcontroller that takes the output of the two sensors and determines if a person is sitting on the chair. Once the chair is activated, the information will be sent through wires to the wireless transmitter.

At a predetermined time, the PIC will tell the transmitter and sensors to power on, and it will select the channels on the transmitter. The PIC interprets data from the temperature sensor through Analog/Digital Conversion (ADC.) In order to perform ADC Conversion on the PIC, we first set a reference voltage to power the PIC (5V). Since our PIC will be using a 10-bit ADC, we resolved the measured voltage by 1 of 1024 values. Therefore, if the reference voltage is 5V, the measured accuracy will be 5/1024 over a 0 to 5V range. By using this information, we then programmed a threshold voltage value to determine whether a chair is in use based on temperature sensor readings.

b)Pic#2:

This component deciphers the information sent by the receiver and outputs the information to the server.

The RS-232 protocol will be implemented on PIC #2 in order to interface with the computer.

PIC #2 is responsible for receiving the data stream from the Receiver and getting the unique ID and information regarding the chairs current occupancy. It will then send this information through the interface with the computer via the RS232 connection. RS232 requires a range of +/- 3V to +/- 15V so we will be using a MAX232 chip to help us get the correct the TTL voltage levels of 0-5V to an acceptable RS232 protocol range. Interfacing with RS232 can be simplified in design to use only four ofthe pins: RxD (Received Data), TxD (Transmitted Data), RTS (Request to Send), and CTS (Clear to

Send). RTS and CTS are used as a handshaking protocol between the PIC and Computer and are required for our design. We only plan to use the RxD pin as our data line since we only send data one way (PIC to Computer).

**2.4 Server**

Our original objective was to use the PC as a way of transferring chair information into the database so that it could be read by a website. For our final design, we decided that this was still the best course of action in getting data from the chair to the database. We chose to use the Serial Port as our communication interface because the PIC and serial port programming would allow us to implement the relatively robust RS-232 protocol of transferring data. Also, the PC would be able to handle the flood of data coming into the serial port due to more memory compared to the PIC. Most importantly, the direct connection that could be established with the database through the PC would allow the most efficient way of storing the data for the show. Visual Basic was chosen as our language of implementation

**2.5 Wireless Transmitter/Receiver**

First we choose Linx RXM-900-HP3-PPS receiver and Linx TXM-900-HP3-PPS transmitter but we did not find this kind of transmitter and receiver.

We decide to use XBee series 2 (2.5) transmitter and receiver

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

**Long Range Data Integrity**

•Indoor/Urban: up to 100’ (30 m)

•Outdoor line-of-sight: up to 300’ (100 m)

•Transmit Power: 1 mW (0 dBm)

•Receiver Sensitivity: -92 dBmXBee

**Advanced Networking & Security**

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Source/Destination Addressing

Unicast & Broadcast Communications

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Coordinator/End Device operation

Serial Data

Data enters the module UART through the DI pin (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted. Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.



The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).



**APPENDIX A – BLOCK DIAGRAMS**

Diagram A.1 below is a block diagram of the overall seat monitoring system. The sensors send a signal to PIC #1 which performs ADC conversion. PIC #1 then sends out the data to a transmitter that transmits the information wirelessly to a receiver. The receiver takes the information and sends it to PIC #2, which transfers the data to a PC to be processed.



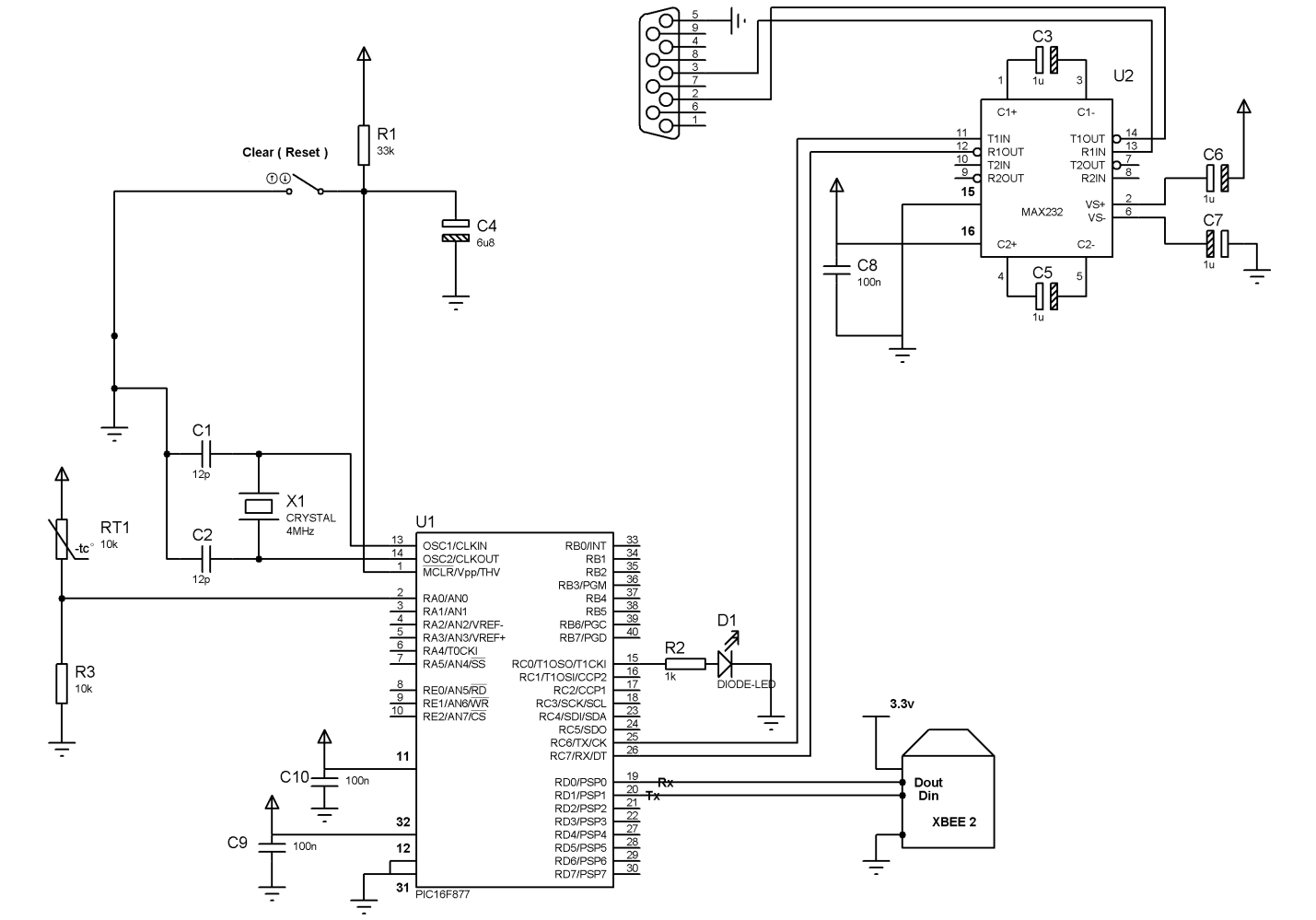
**Diagram A.1** Block Diagram

**APPENDIX B – SCHEMATICS**

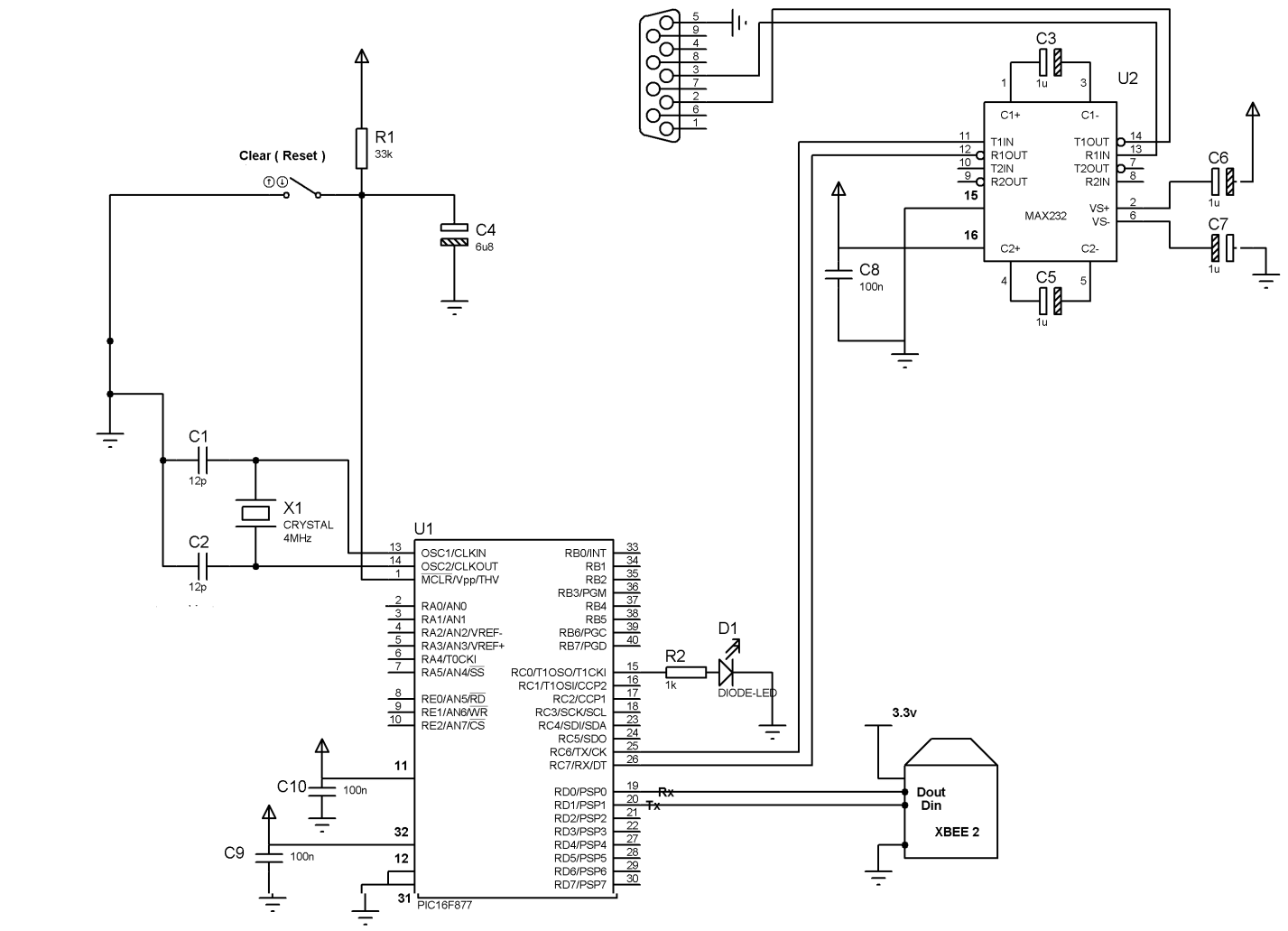
Figure B.1 is a schematic for the Microchip PIC16F877A microprocessor we used in our design. Figure B.2 is a schematic for the transmitter component described by the block diagram above, including interconnections between the sensors to PIC #1 and PIC #1 to the transmitter. Figure B.3 is a schematic for the receiver component described by the block diagram above, including interconnections between the receiver to PIC #2 and PIC #2 to the PC.



**Figure B.1** Schematic of Microchip PIC16F877A



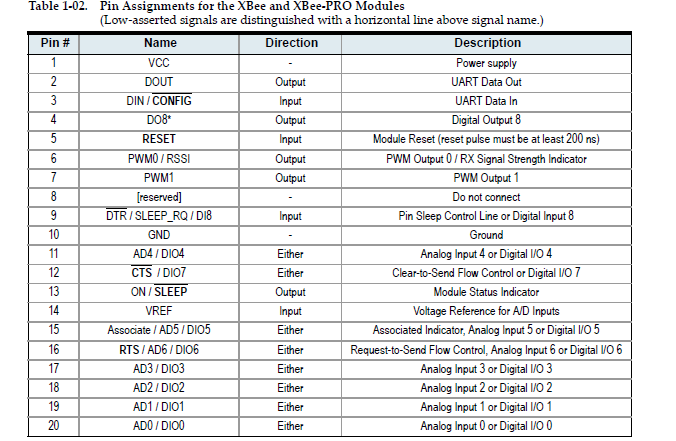
Transmitter cct



Receiver cct



The figure B.2 is a schematic for the XBee transmitter and receiver



The figure B.3 is the connection of the XBee transmitter and receiver with the microcontroller



The figure B.3

**APPENDIX C**

**MAX232 Chip Input/Output**

We need to test whether the MAX232 chip is converting our signal from 0-5V to an RS-232 level of +/- 10V. We have our PIC transmit a byte of data into the input of the MAX232 chip and read the output with an oscilloscope. The resulting waveforms from the MAX232 output show a voltage level of +/-12V that is within our acceptable range of serial port communication using the RS-232 protocol.



**Figure C.1** MAX232 Voltage Conversion

**APPENDIX D**

**Pic code**

transmitter pic code

#include "Tx.h"

void main()

{

while(1)

{

readADC\_1();

if(ADC\_1 > tempthreshold)

{

fprintf(PC\_COM,"Temp Sensor=%lu\n\f",ADC\_1);

fprintf(XB\_COM,"T%lu",ADC\_1);

}

else

{

fprintf(PC\_COM,"Temp Sensor=%lu\n\f",ADC\_1);

fprintf(XB\_COM,"T%lu",ADC\_1);

}

/\* readADC\_2();

fprintf(PC\_COM,"Press Sensor=%lu",ADC\_2);

fprintf(XB\_COM,"P%lu",ADC\_2);\*/

led();

delay\_ms(500);

restart\_wdt();

}

}

void init\_app()

{

setup\_timer\_0(RTCC\_INTERNAL|RTCC\_DIV\_256|RTCC\_8\_BIT);

enable\_interrupts(INT\_RTCC);

enable\_interrupts(GLOBAL);

setup\_adc(ADC\_CLOCK\_INTERNAL );

fprintf(PC\_COM,"Initializing");

}

//////////////

void led()

{

output\_high(PIN\_C0);

delay\_ms(250);

output\_low(PIN\_C0);

delay\_ms(250);

restart\_wdt();

}

void readADC\_1( void)

{

set\_adc\_channel( 0 );

delay\_us(10);

ADC\_1 = read\_adc();

}

/\*

void readADC\_2( void)

{

set\_adc\_channel( 1 );

delay\_us(10);

ADC\_2 = read\_adc();

}

\*/

Receiver pic code

#include "Rx.h"

void main()

{

char ch;

while(true)

{

fprintf(PC\_COM,"%c",fgetc(XB\_COM));

restart\_wdt();

}//Main Loop

}