

Surveillance System Using Wireless Sensor Networks

Dan Nguyen, Leo Chang

Computer Engineering, Santa Clara University

Santa Clara, California, USA

dantnguyen84@gmail.com

chihshun@gmail.com

Abstract—The low power, extended range and low cost make wireless sensor network(WSN) an ideal solution for wide area surveillance. Compare to 802.11, this solution has hundred times battery life, 2-5 times wireless range and 20% or less in cost. The only caveat is the data rate is less than 1Mbps. Thus, WSN is widely implemented to collecting environmental data except streaming audio/video which may use a lot more bandwidth. In this project, we propose using 3 WSN nodes (sensor node, router node and sink node) to simulate the possible applications WSN can provide. We connect “temperature sensor” and “door magnetic switch sensor” to sensor node. The system allows both ambient temperature and door open event report to sink node. Sink node process the data and trigger the event we programmed, such as display the temperature reading in LED and/or activate buzzer when door is opened. We have implemented the proposed system successfully and the result shows that our surveillance system is adaptable to variant environments and provides real time information of the monitored environment.

I. INTRODUCTION

In traditional home sensor networks, you would use wires to connect the different peripherals and sensors. This is a very cluttered method depending on how large your home is and in order to hide the wires, it is a difficult install. If you are wiring an outdoor sensor, more maintenance is required for this wire as animals may chew through it or the elements may destroy it over time.

With a wireless sensor network, instead of using an Ethernet connection of some sort, we’ll be using a wireless ZigBee connection. This protocol will span layers 1-5 of the network at the very least and possibly more depending on how program execution goes wirelessly transmit data between the different sensors.

Fig. 1 is an overview of our Surveillance system. Our design can be separated into three parts. Node #1 and Node #2 are sensor part, Node #3 is router part and Node #4 is processing part. In our system, the sensor nodes (#1, #2) are used to track and report temperature of outdoor and movement of door to a processing unit through a routing node (#3). Controller node (#4) will process the reported data and then 1) display outside temperature to the LCD 2) trigger alarm if door is opened.

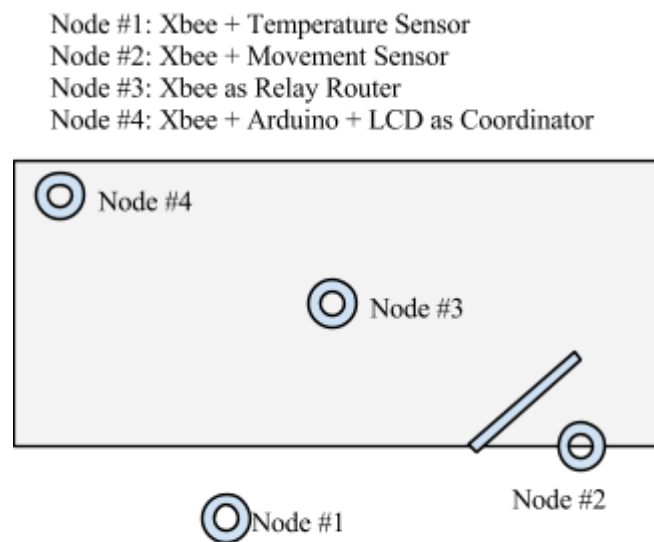


Figure 1. System Overview

We will try to create a wireless sensor networking that includes a temperature gauge, a LCD screen to display this temperature, a door sensor and speaker to signal if the door is open. This will be used as an environmental detection system.

II. THEORETICAL BASES AND LITERATURE REVIEW

We have looked at different literature and see if there are any similar solutions. This does not seem to have been done, so this will be a unique solution. We plan to send the temperature data to a LCD screen located somewhere else. Some projects will use one portion of our overall system, but never the entire system as a whole. Our system is made to fit our needs and will only have the bare minimum to keep the costs low. This is a cheap, efficient, and highly portable solution. We plan to make it very modular, so it can be expanded to take other nodes easily.

III. HYPOTHESIS

We believe that we will finish our solution to build this in one month and under \$150. Both of these are realistic goals and should be attainable during the date of our project defense, 06/11/13. We plan to have a working demo available at this time.

IV. METHODOLOGY

We will buy individual parts and assemble this solution together.

Expected list of parts:

- 4 Xbee and adapters
- 2 Arduino Uno
- 1 Temperature sensor
- 1 Lcd screen
- 1 Magnetic door sensor
- 1 Speaker
- 2 Breadboards

Each Xbee unit is a wireless sensor that has an antenna on it. Since each unit isn't made to fit onto a traditional circuit board, we used adapter kits for ease of use. This allowed us to plug them directly into a circuit board and was able to easily connect to pins this way. The rest of the parts required no other adapters so we were able to begin testing and design.

We started off by working on creating a simple system with two Xbee devices. One Xbee acted as a coordinator in AT mode and the other acted as a router in AT mode. We set them both to the same network ID and sent a chat value from one Xbee to the other. We then programmed each Xbee to use the same PAN ID and have the router have channel verification turned on. By having two terminals open on two different computers, we could type on the router's terminal and it would show up on the coordinator's screen. The router is wirelessly sending data to the coordinator. We used two different laptops; each connected to a separate Xbee and we saw on screen one the data coming from screen two.

After that successful test, we were able to confirm our Xbee devices were working properly and that we understood some of the basics of using them in other ways. We now integrated the usage of an Arduino into the mix. This would give us more flexibility with programming and the ability to run calculations and powerful algorithms. The next design we were to implement is a basic implementation of the Xbee, with an Arduino, and a magnetic door sensor connected. The idea of this design is to connect the magnetic door sensor to the router Xbee and wirelessly send the data to the coordinator Xbee, which is connected to the Arduino. The Arduino will parse this data and convert the digital input and let us know if motion is occurring or not on our sensor. This would be

equivalent to having a sensor on a door or window and this would alert you if someone was breaking into your home or if the door is opening.

One note about the Xbee units and its adapter, the coordinator will have a blinking green light and as well as the router. The router will be blinking twice as fast as the coordinator as it will read and send data back to the coordinator. This was the same for all of our different tests. The coordinator is set to look for data every second while the router will have transactions every half second or so. There is a red LED also that shows the coordinator and router are talking to each other and they will both stay a solid red if connected.

We now set the Xbee to be in coordinator API mode. The API mode is used so the Xbee router and Arduino will have the ability to read the data frames. The router will be in AT mode. Some changes will be made to the router options this time. We enabled a Digital IO pin, which we decided to use pin 4 for our design at random. On the circuit board, we also set up a magnetic door sensor that is a simple switch. It will output a '1' when closed and '0' when opened. A red LED on the board is also connected for ease of use to show us when the switch is opened and closed.

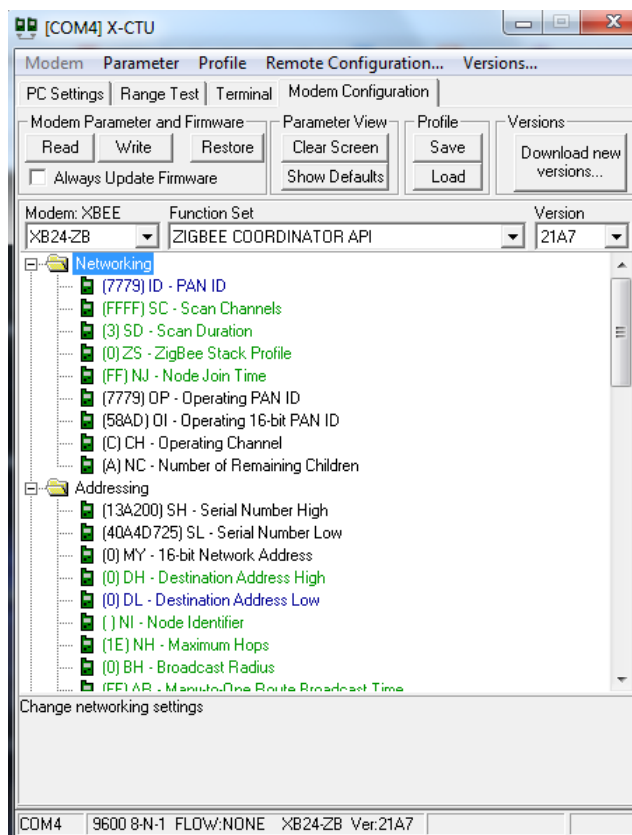


Figure 2

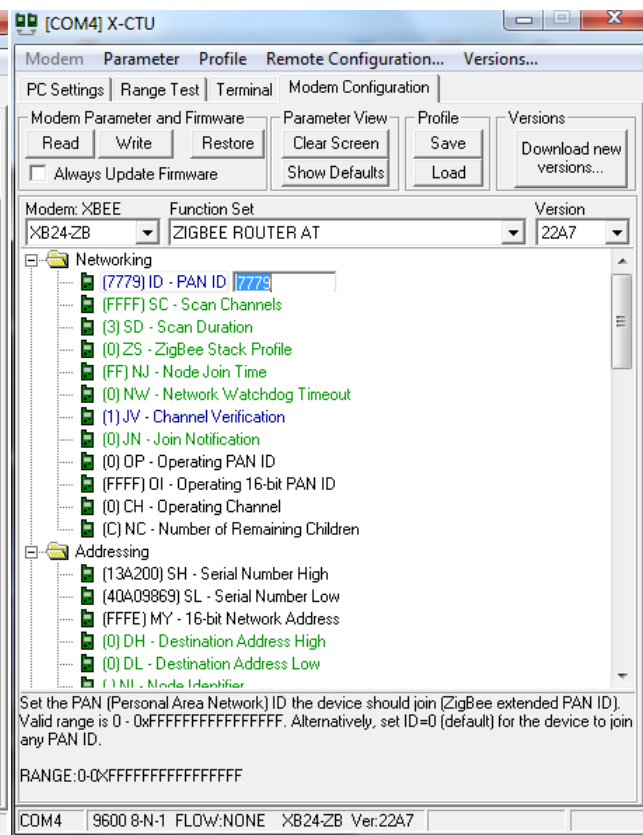


Figure 3

The next portion of the design was where a lot of research had to be done. We had to understand how the API format of the received data. The data frame is broken up into 23 bytes with a start byte of "7E". From here we would create a loop in C to read the rest and we would discard bytes until we reached byte 19. This is where the digital data would be held if any is read. Next up is to parse this byte as each bit will correspond to a different digital IO pin. The data comes in big endian and the most significant pins come first, so pin D12 down to D0. Byte 19 has the outputs for D12-D10 and byte 20 has the outputs of D7-D0. As we used D4, we will have our program look for byte 20 and look at bit 5 of the byte. We then set up our code to look at this byte and show us when the sensor is open and closed with some print statements.

This turned out much more simple than expected after reading the documentation a few times and understanding which data was important and what isn't. Grabbing the starting byte was the most important lesson to be learned during this test and then creating a loop to discard data until you hit the proper byte.

The next test we were to do was to independently get a temperature monitor to work and report data back to us via a terminal. Once again we'll want the coordinator to be in API mode and the router to be AT mode. Now instead of reading the digital IO, we'll be reading an analog IO. Some of the pins are the Xbee both analog and digital. From viewing the schematic, we decided pin ADIO 3. This shows that it is both analog and digital and will parse data based on what type of configuration setting it is given. We go into the program and select Analog to Digital Conversion (ADC) and now move into C code to parse the values given to us. Due to the way the temperature sensor returns values, we had to read their documentation in order to figure out how to represent the data in Fahrenheit and not an analog value.

```
int analogReading = analogLSB + (analogMSB * 256);  
temp = analogReading / 1023.0 * 1.23;  
temp = temp - 0.5;  
temp = temp / 0.01;  
temp = temp * 9/5 + 32;  
DigitShield.setValue(temp);
```

You can see a function called DigitShield, this is an Arduino function set and is used to display the temperature onto a LCD screen connected to our Arduino. This turned out a bit more straight forward than expected as the DigitShield was very easy to use and intuitive. The coding portion was simple and with a bit of debug, we were able to get the temperature sensor up and running. We expected more resistance here as we expected to have to come up with an LCD screen function on our own.

The last portion of our sensor network was to be able to integrate both sensors onto the same Xbee unit. We combined the two settings of the previous experiments. We kept ADIO3 as our temperature sensor, but we then also configured DIO4 as our motion sensor. Unfortunately we ran into some problems with this. We ran into some issues with the data conflicting as we were using both digital and analog data. After reading some more documentation and fiddling with the outputs of the frame, we understood that it pushed down where the data sits and we were able to parse both properly. Now we are able to get an output to the terminal if the door is closed and what the temperature the room is at. On top of this, we have a digital display on the Arduino with the LCD sensor showing us the temperature also.

One thing that we tried to implement, but gave us major headaches was introducing a buzzer into the system. We planned on having a buzzer to alert us when the magnetic door sensor was open, but we had issues integrating it with the rest of the design. Standalone, the magnetic door sensor and the buzzer worked perfectly fine, but we ran into some issues with the code crashing once the buzzer went off on hardware. We looked into this more and found out that this was a common issue with the buzzer with this model of the Arduino. Turns out there is a driver issue and was only rectified on a different model of Arduino.

V. IMPLEMENTATION

The controller unit (in sink node) of proposed system is implemented in Aduino R3 which uses a microcontroller, ATmega328. The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library) available.

The Wireless sensor network we used is XBee S2 which is based on ZigBee - an IEEE 802 standard. A data transmission rate is 250 kbps in the 2.4 GHz frequency band. Power consumption is 3.3V @ 40mA. Output power is 2mW (+3dBm) with 400ft (120m) range.

The sensor node device can sense temperature and door opening event, as shown in Fig. 2 and 3. We connected two sensors (temperature and magnetic switch) in one ZigBee in order to test if it can transmit two sets of data (temperature data and magnetic switch on/off data) simultaneously.

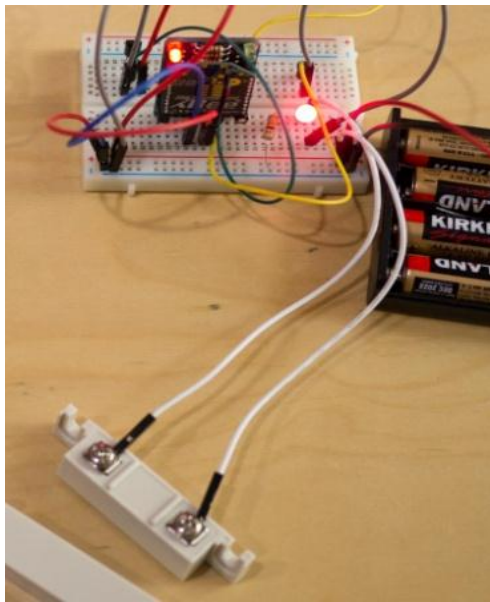


Figure 2

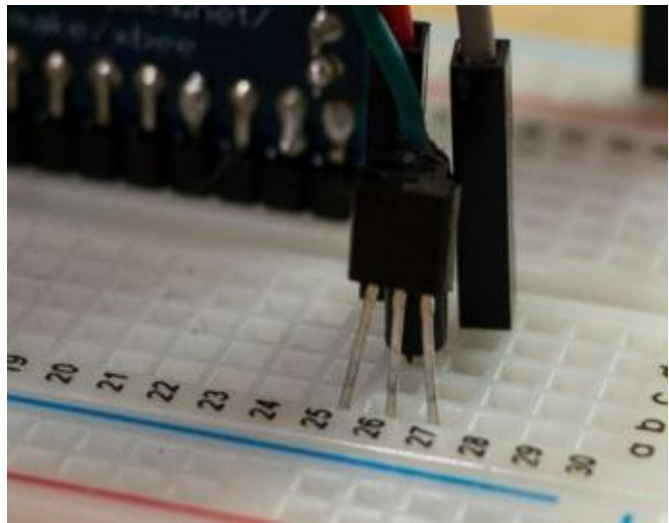


Figure 3

When the sink node receive both sets of data, the Aduino R3 processes the data. This way we can control the output with C coding and run our own functions (Figure 4).

An experimental prototype has the sensor node mounted on a room's main entrance to detect the intrusion from outside, also it provide the temperature of the house for your convenience in daily life.

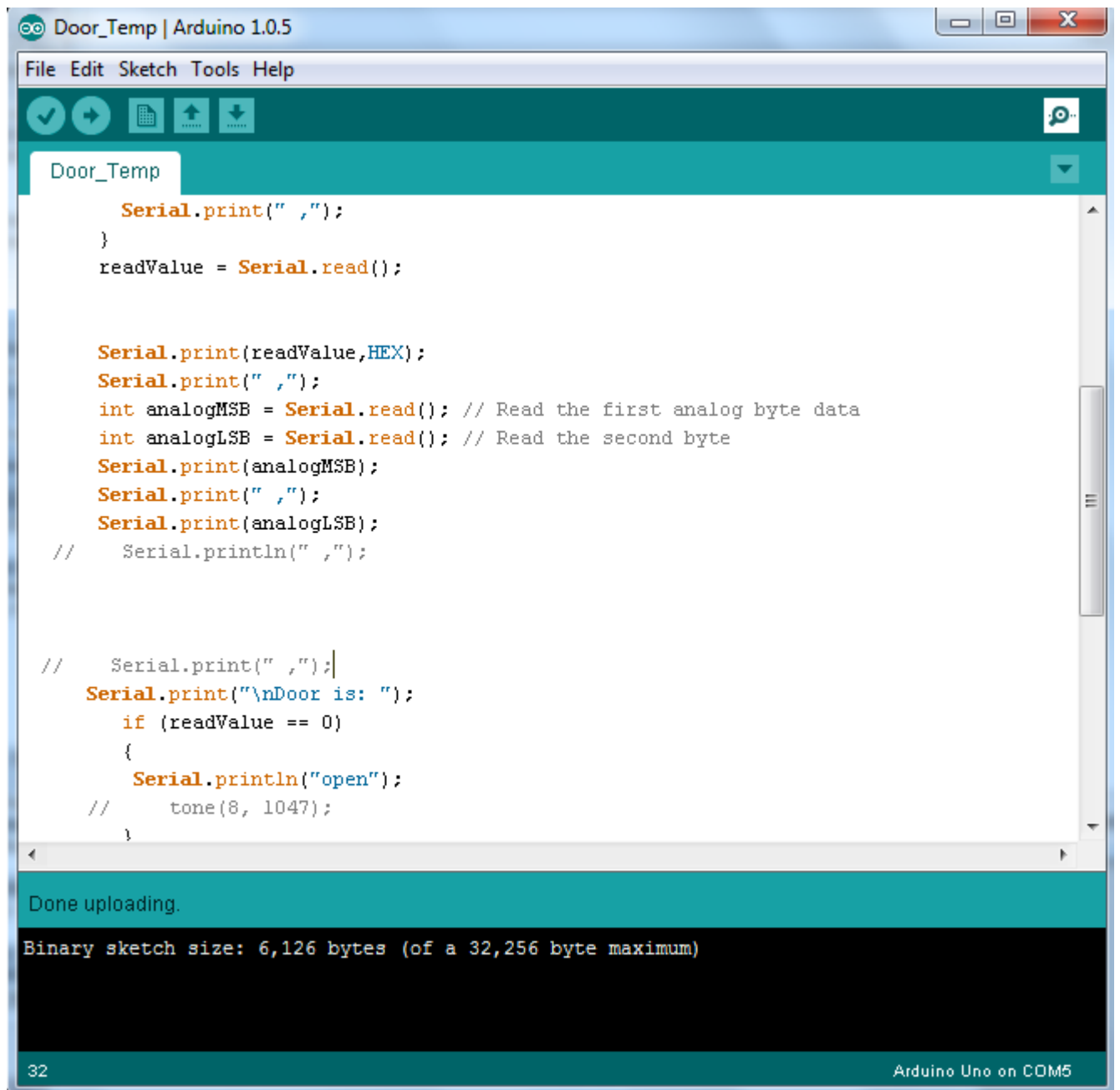
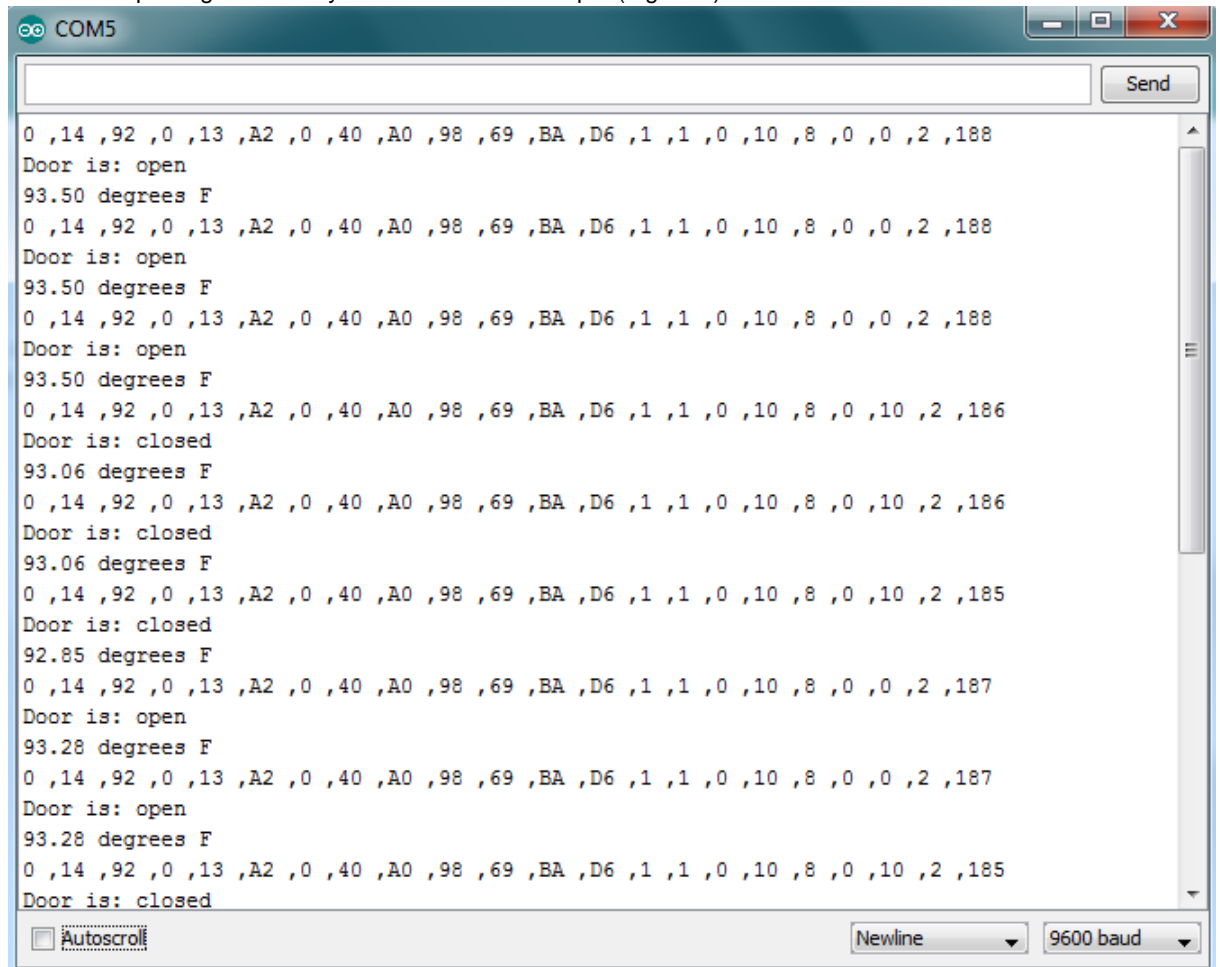


Figure 4

VI. DATA ANALYSIS AND DISCUSSION

The raw data output is generated by Arduino R3 serial output (Figure 5).



```
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,0 ,2 ,188
Door is: open
93.50 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,0 ,2 ,188
Door is: open
93.50 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,0 ,2 ,188
Door is: open
93.50 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,10 ,2 ,186
Door is: closed
93.06 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,10 ,2 ,186
Door is: closed
93.06 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,10 ,2 ,185
Door is: closed
92.85 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,0 ,2 ,187
Door is: open
93.28 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,0 ,2 ,187
Door is: open
93.28 degrees F
0 ,14 ,92 ,0 ,13 ,A2 ,0 ,40 ,A0 ,98 ,69 ,BA ,D6 ,1 ,1 ,0 ,10 ,8 ,0 ,10 ,2 ,185
Door is: closed
```

Figure 5

The serial output byte table is as follows (Figure 6)

API format for I/O Data Sample RX Indicator	Byte	Example	Description	<pre>long chexsum = 0x17 + 0xFF + 0xFF + 0xFF + 0xFE + 0x02; Serial.write(0xFF - (chexsum & 0xFF)); // Checksum</pre>
	0	0x7e	Start byte – Indicates beginning of data frame	
	1	0x00	Length – Number of bytes (ChecksumByte# – 1 – 2)	
	2	0x14		
	3	0x92	Frame type - 0x92 indicates this will be a data sample	
	4	0x00	64-bit Source Address (Serial Number)	
	5	0x13	MSB is byte 4, LSB is byte 11	
	6	0xA2		
	7	0x00		
	8	0x40		
	9	0x77		
	10	0x9C		
	11	0x49		
	12	0x36	Source Network Address – 16 Bit	
	13	0x6A		
	14	0x01	Receive Opts. 01=Packet Acknowledged. 02=Broadcast packet	
	15	0x01	Number of sample sets. Always set to 1 due to XBEE limitations	
	16	0x00	Digital Channel Mask – Indicates which pins are set to DIO	
	17	0x20		
	18	0x01	Analog Channel Mask – Indicates which pins are set to ADC	
	19	0x00	Digital Sample Data (if any) – Reads the same as Digital Mask	
	20	0x14		
	21	0x04	Analog Sample data (if any)	
	22	0x25	There will be two bytes here for every pin set for ADC	
	23	0xF5	Checksum(0xFF - the 8 bit sum of the bytes from byte 3 to this byte)	

Sleep Mode

Endpoints can sleep to save power. A endpoint that only wakes up every 5 minutes to send data may only be awake for 6 seconds a day.

SM – 4 = Cyclic sleep

SP – Sleep time (up to 28 secs)

SN – Number of sleep cycles

ST – Time awake

Digital Ch Mask

First Byte
n/a n/a n/a D12 D11 D10 n/a n/a

Second Byte
D7 D6 D5 D4 D3 D2 D1 D0

Example:
0x00 0x13 = 0000 0000 0000 1101

Pins D3, D2 and D0

Analog Ch Mask

(volt) n/a n/a n/a A3 A2 A1 A0

Example:
0x05 = 0000 0101 = Pin A2 and A0

Figure 6

We can discard the Bytes #0-#18 and take Bytes #19 & #20 for digital sample data (Door on/off information). Bytes #21 and #22 are analog sample data which give us analog reading of temperature.

For digital sample data we got here (0, 0) or (0, 10). (0, 0) is received when the magnet switch is at "off" position, otherwise it receive (0, 10). We can read the data and make buzzer buzz when it receives (0, 0).

As for analog sample data, it vary from (2, 0) to (2, 512). Apparently, we need to do some conversion in our program. The conversion part in code is as follows.

```
temp = analogReading / 1023.0 * 1.23;
temp = temp - 0.5;
temp = temp / 0.01;
temp = temp * 9/5 + 32;
```

VII. Conclusion

This paper documents our steps and the process we took to create this design. Fortunately we were able to create a simple sensor network using XBee and Arduino to have a motion sensor and a temperature sensor. We were able to meet our goal of spending less than \$150 and finishing by 06/11/2013. In the end, we spent about \$100 dollars on the parts and were able to finish on 06/08/2013. The XBee is a simple, but powerful method of wirelessly transmitting signals across a home while the Arduino makes this simple data transfer even more useful as were able to process information on the fly and relay commands if necessary. For the scope of this project we didn't add it to our final product, but we did do some testing to control the output from the coordinator XBee and it worked flawlessly. When expanding this network, adding more commands and the ability to parse more data is trivial as it just means some upgrades to the code. Overall, this is a fun project any hobbyist can attempt if they were looking to create their own home security sensor network. With a bit of patience, knowledge and some parts, you can get a full security system up and running for half the price of a commercial system set up exactly the way you want it.