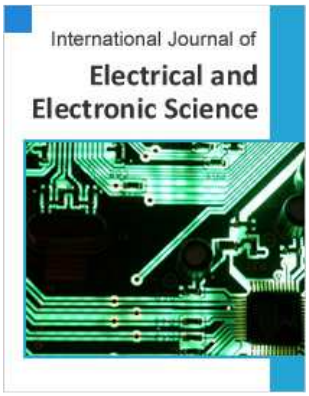
International Journal of Electrical and Electronic Science
2015; 2(3): 64-73
Published online September 30, 2015 (http://www.aascit.org/journal/ijees)
ISSN: 2375-2998

American Association for Science and Technology



Keywords
Arduino,
Trainer,
Wireless,
Sensor,
Networks,
Experiment

Received:August 29, 2015
Revised: September14, 2015
Accepted: September15, 2015

Development of an Arduino-Based Trainer for Building a Wireless Sensor Network in an Undergraduate Teaching Laboratory

M. O. Onibonoje, U. N. Ume, L. O. Kehinde

Department of Electronic and Electrical Engineering, Obafemi Awolowo University, Ile – Ife, Nigeria

Email address
femi_onibonoje@yahoo.com (M. O. Onibonoje)

Citation
M. O. Onibonoje, U. N. Ume, L. O. Kehinde. Development of an Arduino-Based Trainer for Building a Wireless Sensor Network in an Undergraduate Teaching Laboratory. *International Journal of Electrical and Electronic Science*. Vol. 2, No. 3, 2015, pp. 64-73.

Abstract
Wireless sensor network (WSN) is widely considered as one of the most important technology for the twenty-first century. Wireless technologies have achieved dramatic growth in recent years. They have received tremendous attention from both academia and industry all over the world, capturing the interest of various sectors. The increasing popularity of WSN has motivated universities to include the subject in their curricula. Effective learning and teaching of WSN however, requires the students to gain hands-on experience in developing WSN projects, which help the students to understand the strength and limitations of this new technology. In this paper, WSN has been simplified such that, students with little or no knowledge about microcontrollers could build one to various applications. It aims at developing a step-by-step approach to building a WSN. It first ensures that students have a friendly exposure to some components required for the tasks. The students will perform four experiments in varying complexity; starting with involving them in the use of the microcontroller, (Arduino board in this case), establishing a chat between two XBee radio modules, and to an advanced project of building a wireless sensor network.

1. Introduction

A wireless sensor network (WSN) is a network made up of a set of independent sensor nodes deployed over a geographic area to collect environmental data and to transmit the gathered data to a base station typically through wireless channels. The base station is generally a computer connected to a gateway, which is a device that collects data from the sensors. An application running on the base station analyzes the received data, performs appropriate computation, and displays the information on the user screen. Wireless Sensor Networks has in recent years been found to have vast application in various areas; from health care to utilities and in remote monitoring.

The WSN elements are basically the sensor nodes and gateway or sink node as shown in Fig 1.1.The composite units include: a number of sensors (such as temperature, moisture, and vibration sensors), a power source, a radio transmitting/receiving unit (transceiver), and an electric "brain" (processing unit, processor) (Huang *et al.,* 2015). The gateway node often consists of transceiver, microcontrollers and data collector, most times a personal computer (PC).

According to Faludi (2007), building WSN requires essential guides for anyone

interested in wireless communications for sensor networks, device hacking or home networking. The processes require correct selection, configuration and applications of components. In this paper, WSN is demystified by first ensuring that the students have a friendly exposure to the components used in building the WSN. The students will perform four experiments in varying complexity; starting with the use of the microcontroller, Arduino board in this case, establishing chat between two XBee radios and to a slightly advanced project of building a simple WSN.

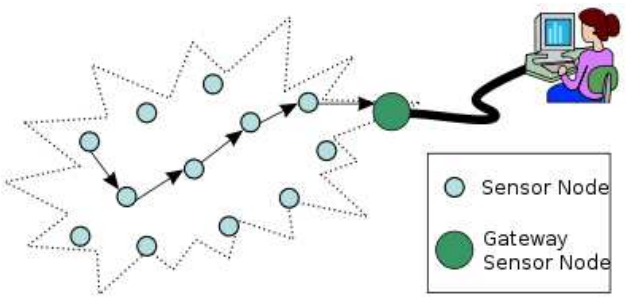


Fig. 1.1. *Parts of a WSN.*

As discussed by Hariprabha and Vasantharathna (2014), wireless sensor network (WSN) is widely considered as one of the most important technologies for the twenty-first century. In the past decades,it has received tremendous attention from both academia and industry all over the world. According to Bokare and Ralegaonkar (2012) and Santoshkumar *et al.* (2012), a WSN typically consists of a large number of low-cost, low-power, and multi-functional wireless sensor nodes, with sensing, wireless communications and computation capabilities; and a base station responsible for receiving and processing data collected by the nodes. These sensor nodes communicate over short distance via a wireless medium and collaborate to accomplish a common task, for example, environment monitoring, military surveillance, and industrial process control. Guo *et al.* (2014) and Kurose *et al.,* (2003) explained the basic philosophy behind WSNs that while the capability of each individual sensor node is limited, the aggregate power of the entire network is sufficient for the required mission. Advances in wireless sensor network (WSN) technology has provided the availability of small and low-cost sensor nodes with capability of sensing various types of physical and environmental conditions, data processing, and wireless communication (Younis and Akkaya, 2013; Hawbani and Wang, 2013).

2. Methodology

In demonstrating the development of a training module for the WSN, various experiments categorized into three sections were conducted with the selected component units. The first section involved working with the Arduino, while the second section involved working with XBee modules. The third section combines the features of the first and second sections to build a WSN. The composite units of WSN include both hardware and software required units. The experiments described in this paper focused mainly on Windows operating system (OS) on the PC. The experiments could also be replicated for other OS like Linux and Macintosh by specific modifications of the procedures.

2.1. Selecting Components

The basic block diagram for the proposed module is as shown in Fig 2.1.

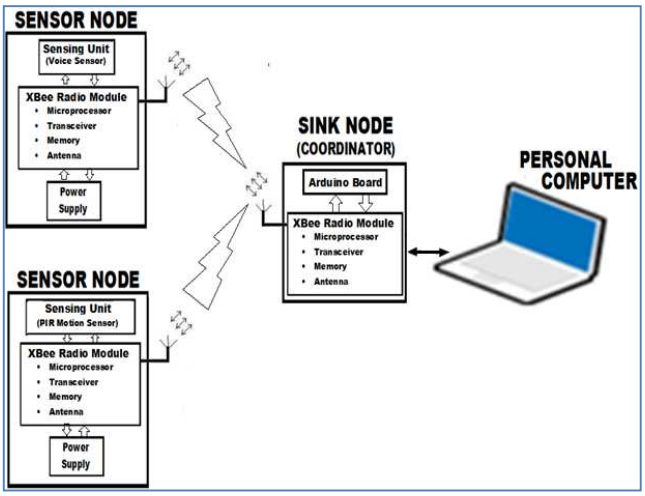


Fig. 2.1. *Block Diagram of Proposed Module.*

Sensing Unit: This unit consists of the sensing device of interest. The sensors in this work were the Sensitive Voice Sensor and the PIR Motion Sensor as shown in Fig 2.2

Fig. 2.2. *(a) Voice Sensor (b) PIR Motion Sensor.*

Microcontroller Unit: The basic function of the unit is to

process the data collected from the sensors for further operations. The microcontroller unit in this trainer is the Arduino board as shown in Figure 2.3. Arduino is an open -source physical computing platform designed for electronics experiments with more fun and intuitiveness. Arduino has its own unique, simplified programming language, a vast support network, and thousands of potential uses, making it the preferred platform for a beginner.

Fig. 2.3. *The Arduino Board (Processor).*

Radio Module: The unit consists of a radio transceiver, microprocessor, memory and antenna. XBee (S1) produced by Digi International as shown in Fig2.4 was used in this work. All XBees communicate over radio with each other in the same way. However, they can use their local serial connection in two very different ways: Application Programming Interface (API) mode and Transparent 'Attention' (AT) mode. Radios configured for API mode utilize a data-enveloping format that is great for computers talking to each other but is not easily human-readable. XBees that are configured to use AT commands are designed for more direct human interaction. AT-configured radios switch back and forth between two modes:

Fig. 2.4. *XBee S1 Module.*

Power Supply: The power supply needed for the experiments can be gotten from 9V AA batteries or from the USB port of the computer.

Other Components:

a. The XBee UartsBee Adapter: This is as shown in Fig 2.5 (a). It was easy to plug the radio module directly into the USB port of the personal computer (PC) using USB cable.
b. XBee Explorer Regulated**:** This component as shown in Fig 2.5 (b) takes care of the 3.3V regulation, signal conditioning, and basic activity indicators (Power, RSSI and DIN/DOUT activity LEDs). It translates 5V serial signals to 3.3V so that it could be connected to any XBee module with the 3.3V requirement.

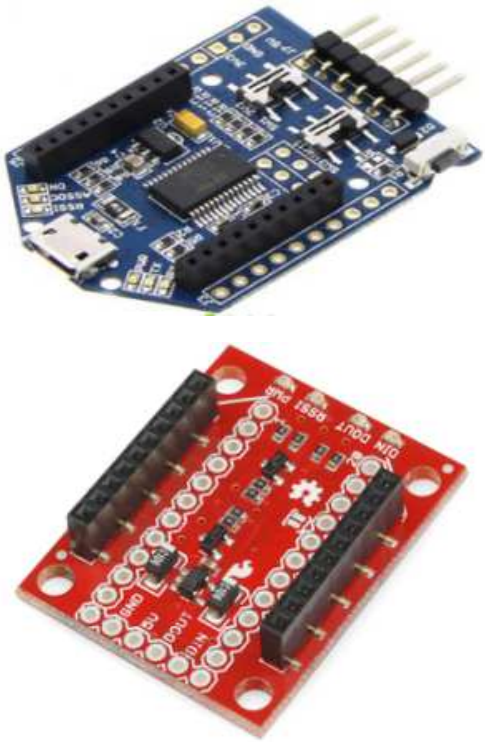


Fig. 2.5. *(a) XBee UartsBee (b) Explorer Regulated.*

2.2. Working with the Arduino Board

Installation Requirements and Procedures:

STEP 1: Download Arduino Integrated Development Environment (IDE) from *www. Arduino.cc/en/Main/Software* for the appropriate computer operating system.

STEP 2: Connect the Arduino to the PC using USB cable that sometimes come with the Arduino kit.

STEP 3: Install the driver for Windows operating system based computer using *http://Arduino.cc/en/Guide/Windows*. Driver installations for other operating systems could be made using *http://Arduino.cc/en/Guide /MacOSX* for Macintosh OS X and *http:// www.Arduino.cc/playground /Learning/Linux* for Linux 32 / 64 bits.

STEP 4: Open the Arduino IDE software on the computer. Poke around and get to know the interface and identify the icons. This step is to set the IDE to identify the Arduino Uno being used in this work. Take note of the four most important commands; Open, Verify, Upload, Serial monitor.

STEP 5: Select the board following the trend: Tools > Board>Arduino Uno.

STEP 6: Select the serial device using: Tools> Serial Port> COM 'port number'. With the Windows OS used in this work,

select the COM number for the serial device of the Arduino board from the Tools/Serial Port menu. This is likely to be COM3 or higher because COM1 and COM2 are usually reserved for hardware serial ports. To find out which port has the Arduino device, disconnect the Arduino board and re-open the menu. The Com port number that disappears should be that of the Arduino board. Reconnect the board and select that serial port. For other types of OS, this procedure is also easy to follow and execute.

Experiments 2.2.1 and 2.2.2 are instructive to demonstrate the working of Arduino Uno board in building a wireless sensor network.

Experiment 2.2.1: Blinking a Light Emitting Diode (LED)

Objective: Starting off by blinking an LED is as simple as turning a light ON and OFF, and changing the rate at which the light blinks.

The components needed for this experiment include Arduino Uno**,** one LED**,** 330 Ohms resistor and Jumper wire.

Procedure: Repeat STEP (4 – 6) of the installation procedures. Connect the circuit following the schematic in Fig 2.6.

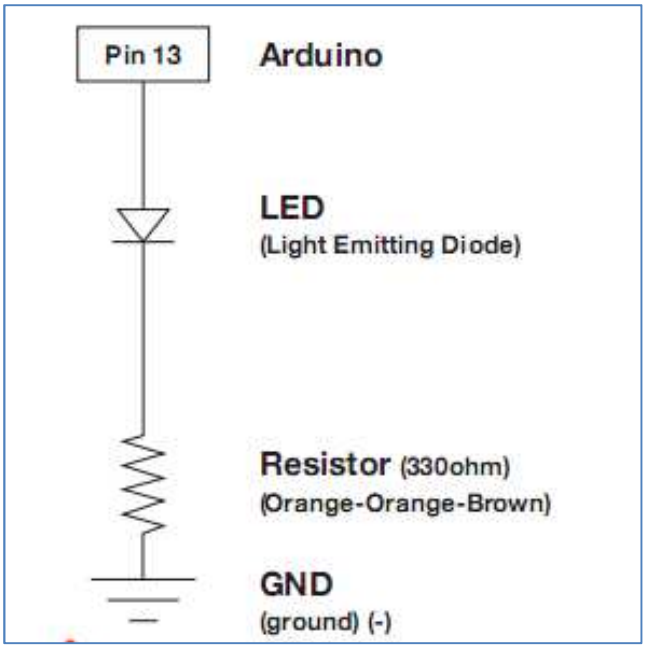


Fig. 2.6. *Connections for blinking an LED.*

Connect the Arduino to the computer using the USB cable. Launch the Arduino IDE. Select File > Examples > Basics > Blink. A new code window pops up as shown in Fig 2.7

Compile the code by clicking the Verify button. Doing so highlights any errors and turns them red when they are discovered. If the code compiles correctly, click upload to send the sketch (code) to the Arduino board. The LED should start blinking. Multiple LEDs can also be blinked sequentially using this same experimental procedure, and by modification of the circuits and codes.

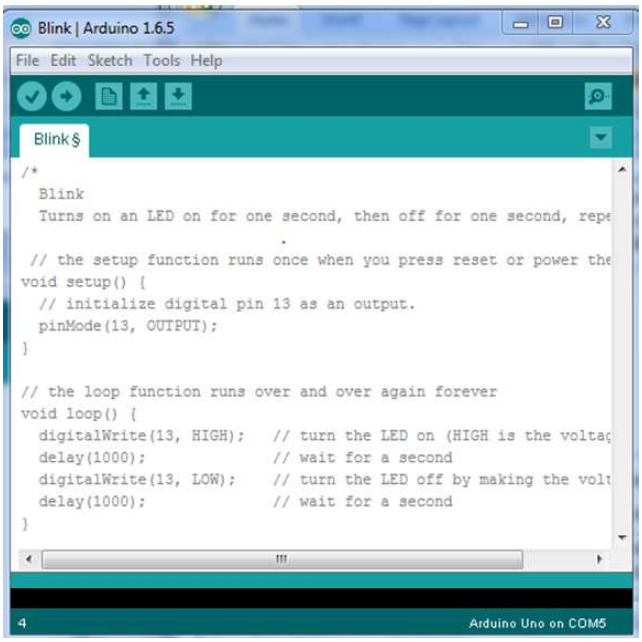


Fig. 2.7. *Arduino IDE with Codes.*

Experiment 2.2.2: Detecting Movement

Objective: To learn how to use the SE-10 PIR sensor to detect movement with the Arduino.

The components needed include Arduino Uno, a bread board, an SE-10 PIR motion sensor and jumper wires

Procedure: This particular PIR sensor has three wires: RED: is the power source and should be connected to 5V. BLACK: is the signal wire and not ground. BROWN: is wired to ground as shown in Fig 2.8.

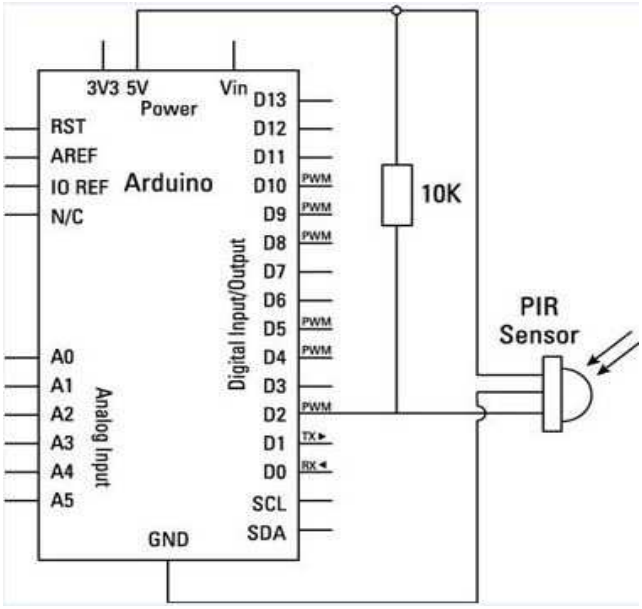


Fig. 2.8. *Connection of PIR Sensor to Arduino.*

Repeat the procedure in experiment 2.2.1, but with the codes shown in Fig 2.9. Open the serial monitor of the Arduino IDE to view the stream of the result.

```
/* project with PIR motion sensor*/
int pushButton = 2;

void setup()
{
  Serial.begin(9600);
  pinMode (pushButton, INPUT);
}

void loop()
{
  int buttonState = digitalRead (pushButton);
  Serial.println(buttonState);
  delay(500);
}
```

Fig. 2.9. *Codes for Detecting Movement.*

Opening the serial monitor resets the sketch and the sensor calibrates itself in the first 1 to 2 seconds. When movement is detected, the output pin state values change from high (no movement) to low (movement).

2.3. Working with XBees

Installation and Configuration:

X-CTU is the officially produced program by Digi International for configuration of XBEE radios. The installation instructions are also available at *www.digi.com*. X-CTU can be used to configure radios' settings. Once the firmware is loaded, a different serial terminal program like Coolterm, HyperTerminal, Tera Term, ZTerm, etc could be used to communicate with the XBee in AT command mode. It is very helpful to have familiarity with one or more serial terminal as access to the X-TCU may not be absolute always. In this paper, we would make use of the terminal on the X-CTU. In addition to the X-CTU, appropriate hardware device drivers are needed to be installed for the XBee adapter board.

Procedures:

Plug one of the XBee radios into the XBee UartsBee adapter and connect the adapter to one of the computer's USB ports. Launch the X-CTU application. Click to discover the radio module attached to the computer. Select the COM port and check the port parameters settings with default (Baud: 9600, Data: 8 bit, Parity: None, Stop bits: 1, Flow control: None); and click 'Finish'. If the connected module is detected, it lists the module(s). Clicking the listed module populate the features embedded in the module(s). These features include but not limited to 64-bit serial number address as printed on the back of each module indicating both the high and the low parts. These procedures are illustrated in Fig (2.10 – 2.12).

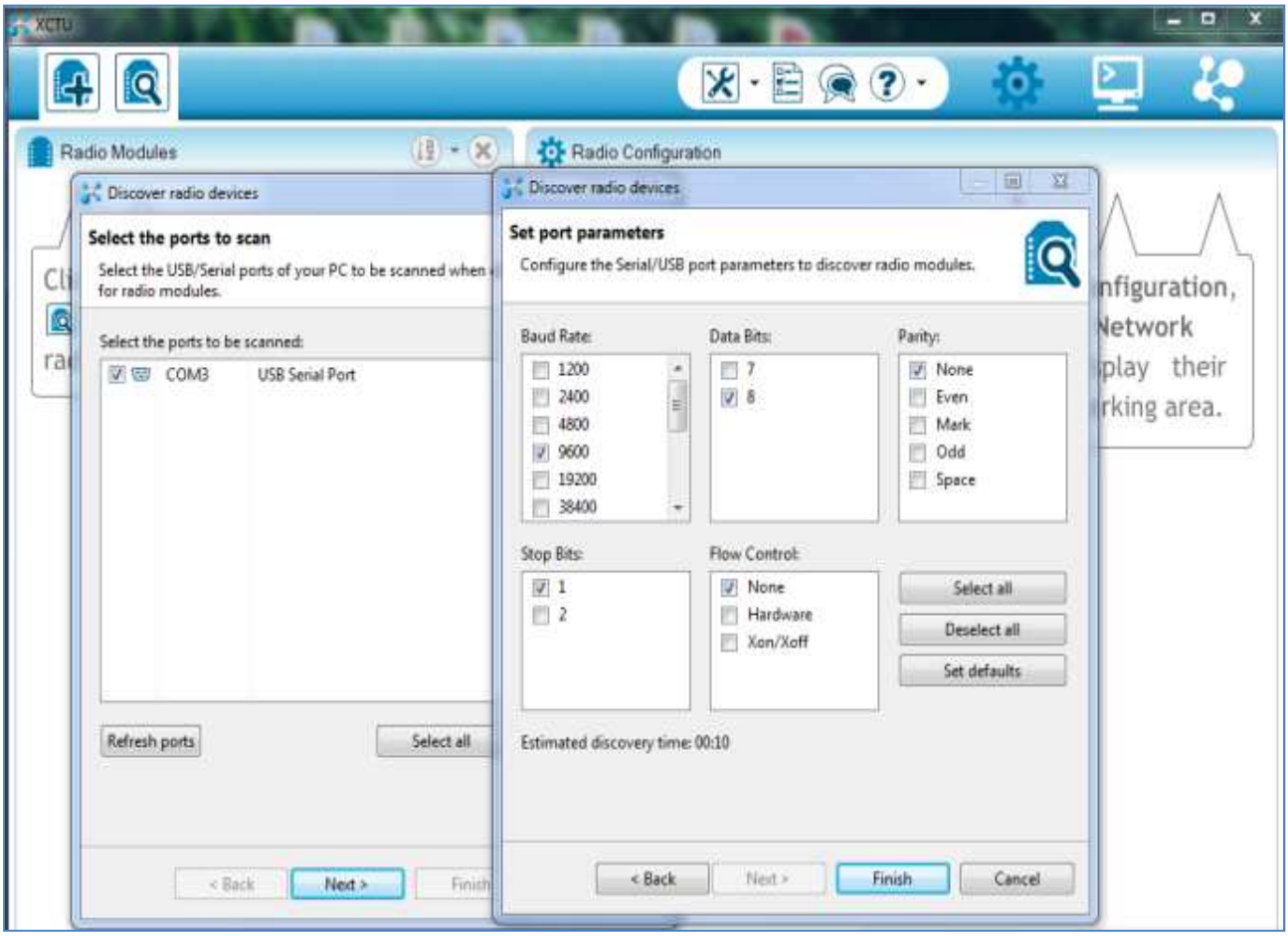


Fig. 2.10. *Port Detection and Parameters Setting in X-CTU.*

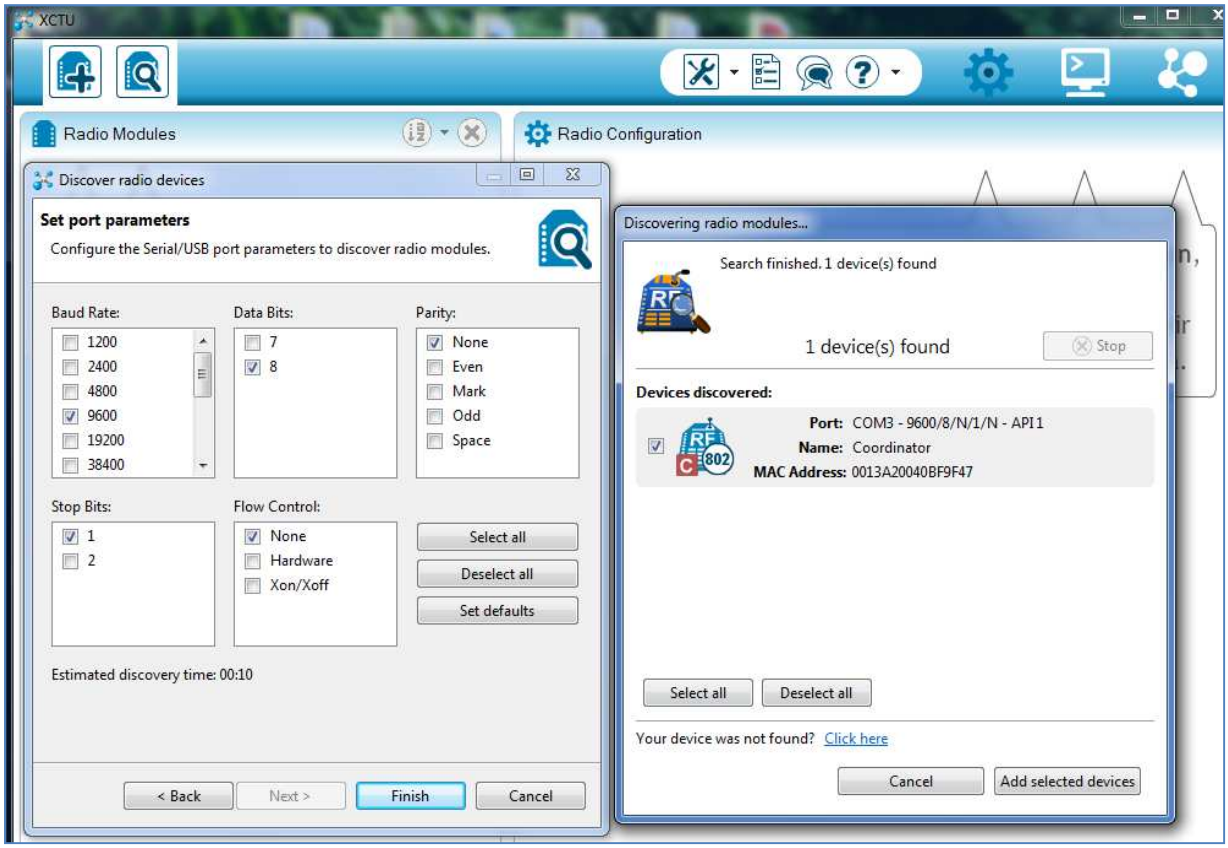


Fig. 2.11. *Discovering Radio Module byX-CTU.*

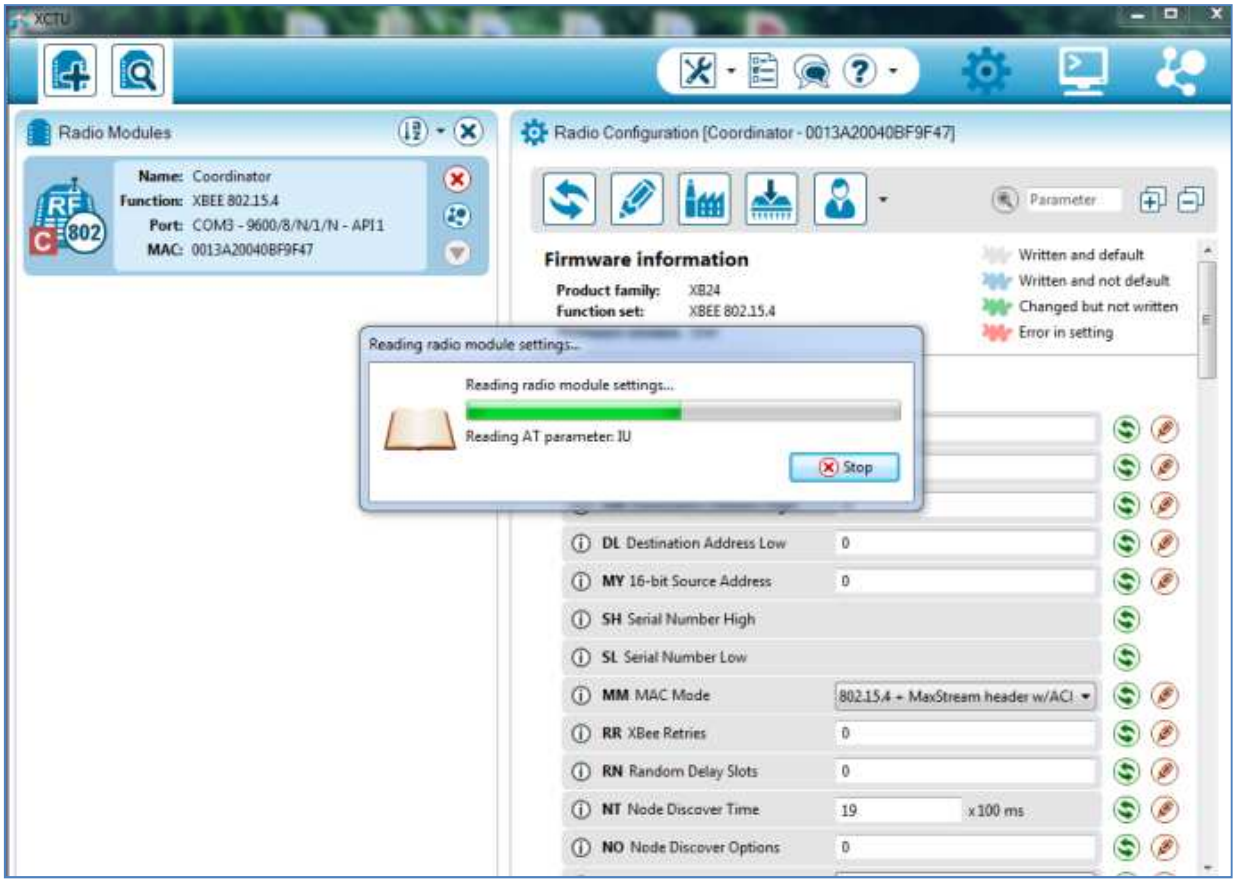


Fig. 2.12. *Radio Module Configuration with X-CTU.*

Experiment 2.3: Two Way Chat between XBees Series 1
Objective: To perform a basic Zigbee chat

The components needed include: Two XBee Series 1 radio modules to be configured for communication in AT command mode, two XBee USB adapter boards, and one computer running two different serial terminal programs or two computers, each running a serial terminal program (used in this paper). The set-up is as shown in Fig 2.13.

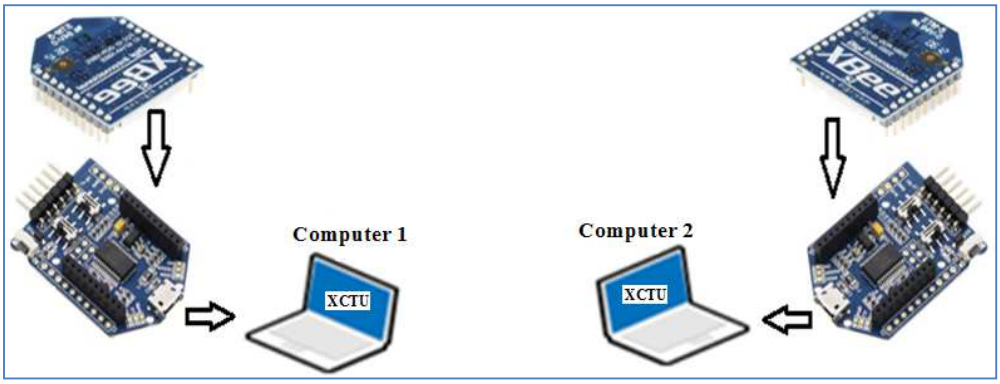


Fig. 2.13. *Set-up for Two-way Chat of XBees.*

Procedure:

Step 1: Configure Radios

Insert the XBees into the XBee adapter boards and connect each to a USB port of a computer. Repeat the procedures in section 2.3 for each of the modules. Select a Personal Area Network Identification (PAN ID) number between 0 and FFFF in hexadecimal. The value will be the same for both modules. Ensure that Destination Address High (DH) and Destination Address Low (DL) are the same for the two radios. Leave the API Enable at API disabled [0] (default). Write the changes in the radio settings by clicking the write button. By this configuration, the first XBee to be powered-up acts as the coordinator and sets up the mini network. The other XBee joins the network when it is powered-up and acts as the router or end device to communicate with the coordinator.

Step 2: Chat between the modules

Switch to the console working mode on each of the X-CTU application pages and open the serial connection with the radio modules. If everything is set up properly, any text that is typed in the console log on the first computer will be relayed to the second computer and appears on its console log screen as well.

2.4. Building Wireless Sensor Network

Wireless networks are all about wireless connections. So, configuring a single radio or connecting LED or a sensor directly to an Arduino does not qualify as a wireless sensor network. In achieving this section, this paper combines the ideas in sections 2.2 and 2.3.

Experiment 2.4: Wireless Sensor Network
Objectives: To build a wireless network of sensors with the sensors data visualized on the screen.

The components needed include: two bread boards, three XBee Series 1 modules, three explorer regulated boards, two 9V batteries, Two 7805 voltage regulators, one Arduino Uno, PIR sensor, Voice sensor, and Jumper wires. The block diagram of our wireless sensor network is as shown in Fig 2.1.

Procedure:

Connect each of the XBee modules to a USB port of the computer for configuration: one as the 'Coordinator', and the remaining two as 'Router'. Launch the X-CTU and modify the settings. For the coordinator, set the CE Coordinator Enable to Coordinator [1] and API Enable to API enabled [1]. For the other two modules, leave CE Coordinator Enable at End Device [0] and API Enable at API disabled [0] (default). Ensure pin D4 is set to '3' Digital Input' (We chose that because it is a digital I/O pin). Set the sampling rate to '3e8' (1000 Seconds delay). Ensure the device type identifier under 'Diagnostics' is set to '10000'. and write to the radios.

Mount the XBee radios on the regulated explorer board as shown in Fig 2.14. Be sure to connect pin 1 and pin 20 of the radio to pin 1 and pin 20 of the regulated board respectively as labelled on them.

Fig. 2.14. *XBee on a Regulated Board.*

Make connections for the two sensor nodes as shown in Fig 2.15. Take note of voltage input, signal output and ground terminal of the different sensors. The modules configured as the routers are to be connected in the sensor nodes; with PIR motion sensor connected in one sensor node and Voice sensor connected in the other sensor node.

Connect the base station as shown in Fig 2.16. The module configured as the coordinator will be connected in the base station.

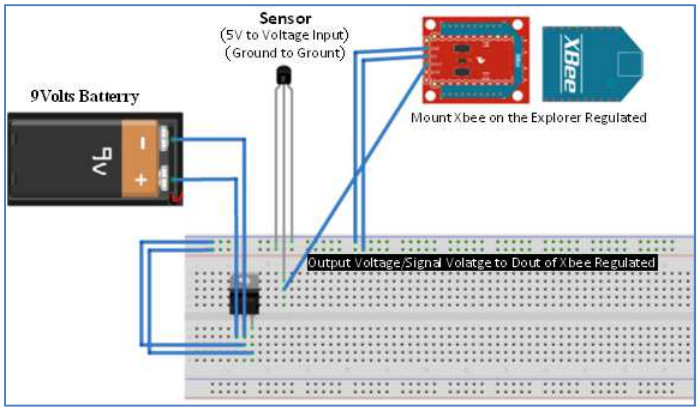


Fig. 2.15. *Schematic for Sensor Nodes.*

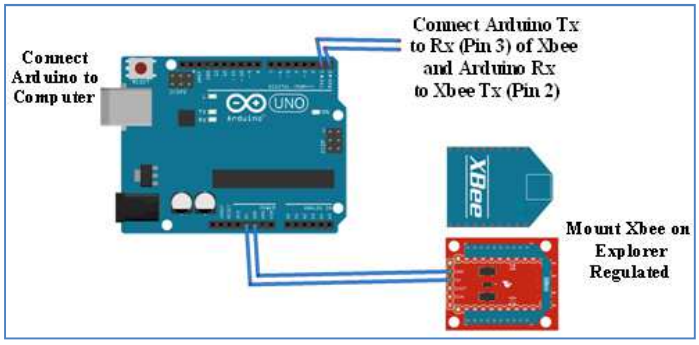


Fig. 2.16. *Schematic of the Coordinating Unit.*

Connect the Arduino Uno to the computer using the USB cable. Launch the Arduino IDE and type the code in Fig 2.17 into the window.

```
int value = 0;
int valueee = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
   if(Serial.available() > 21)
 {
   if(Serial.read() == 0x7E)
 {
    for(int i = 0; i< 13; i++)
 {
    byte k = Serial.read();
    if(i == 5)
  {
     Serial.print("PIR : ");
     if(k < 60 )
    {
      Serial.print('0');
    }
    else
    {
      Serial.print('1');
    }
    //Serial.print(k);
       Serial.print(",");
    }
       if(i == 12){
       Serial.print(" Voice Sensor : ");
       Serial.print(k);
       Serial.print(",");
       }
    }
    Serial.println();
   }
   }
   }
```

Fig. 2.17. *Code for Wireless Sensor Network.*

Disconnect the XBee pins connected to the TX and RX of the Arduino; compile the code and upload the code to the Arduino board. Reconnect the XBee to the Arduino and open the serial monitor of the Arduino IDE to see the communication. The serial monitor shows the data received from the sensor nodes. Each sensor node communicates its data directly to the base station in a star topology mode.

3. Results and Discussion

The result of blinking of LED in Experiment 2.2.1 is as shown in Fig 3.1. The voltage level on the output pin of the Arduino Uno goes high and the LED turns on. After a 1 second delay, the voltage level on the pin goes low and the LED goes off. The process repeats over and over.

The result of two-way chat of radio modules in Experiment 2.3 is as shown in Fig 3.2. The text ‘‘Heello friend, are you in my network?’’ that was typed in the first radio’s console appeared in the second module’s console. Also, the response ‘‘Yes, I am with you’’ typed in the second console appeared in the first console.

The wireless sensor network built in Experiment 2.4 is as shown in Fig 3.3 and the result of data collected is as shown in Fig 3.4. The PIR sensor waits 1-2 seconds after power-up for the sensor to get a snapshot of the still room with digital output of ‘1’ (high voltage level). When movement was detected, the output value changed from 1(no movement) to 0(movement). The voice sensor provided analog representations of the

amplitude of sound from clapping of hands.

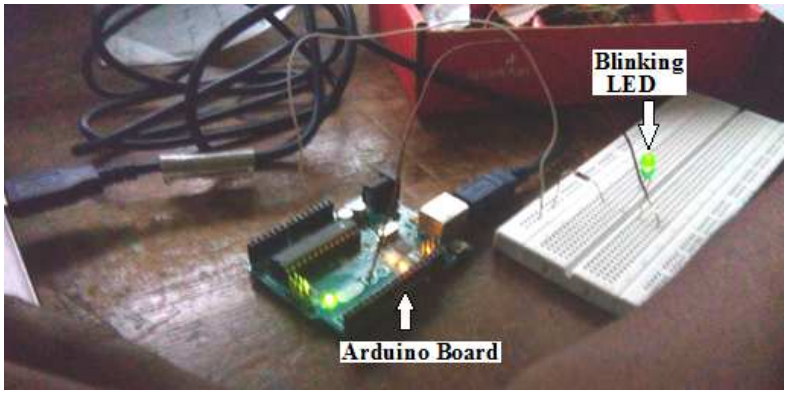


Fig. 3.1. *Blinking an LED.*

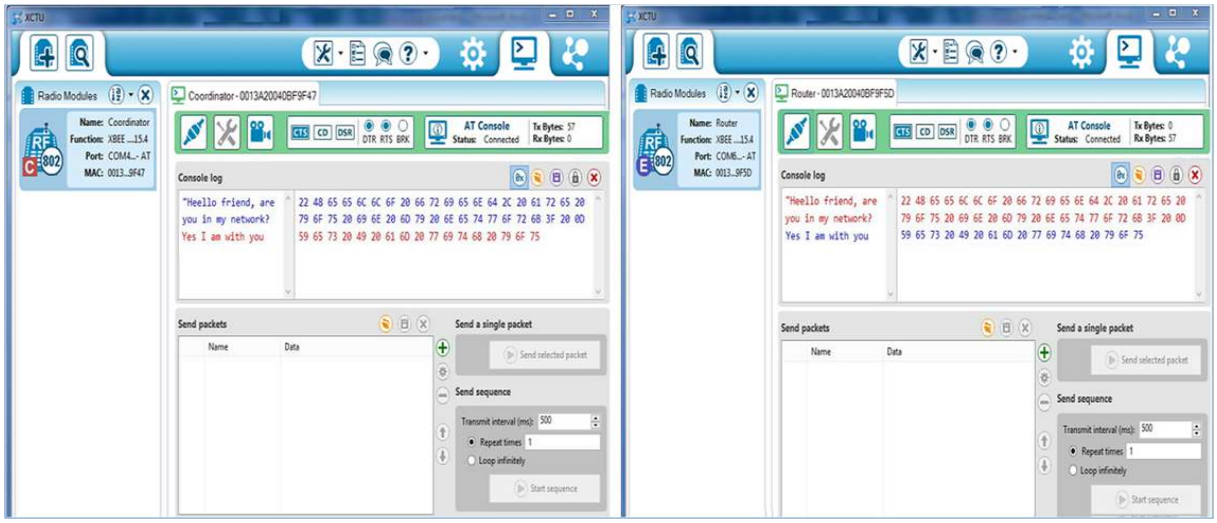


Fig. 3.2. *Display Screen of Two-Way Chat between Radio Modules.*

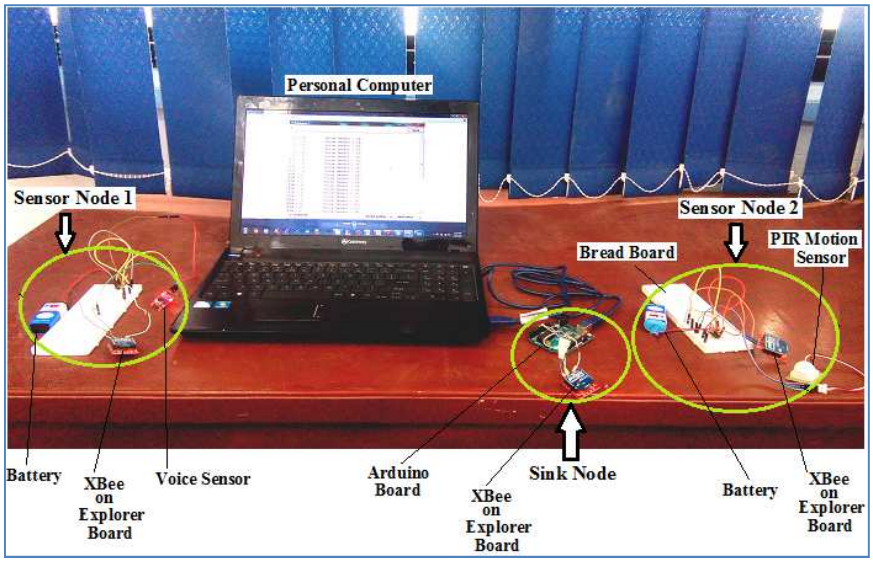


Fig. 3.3. *WSN of Three Nodes.*

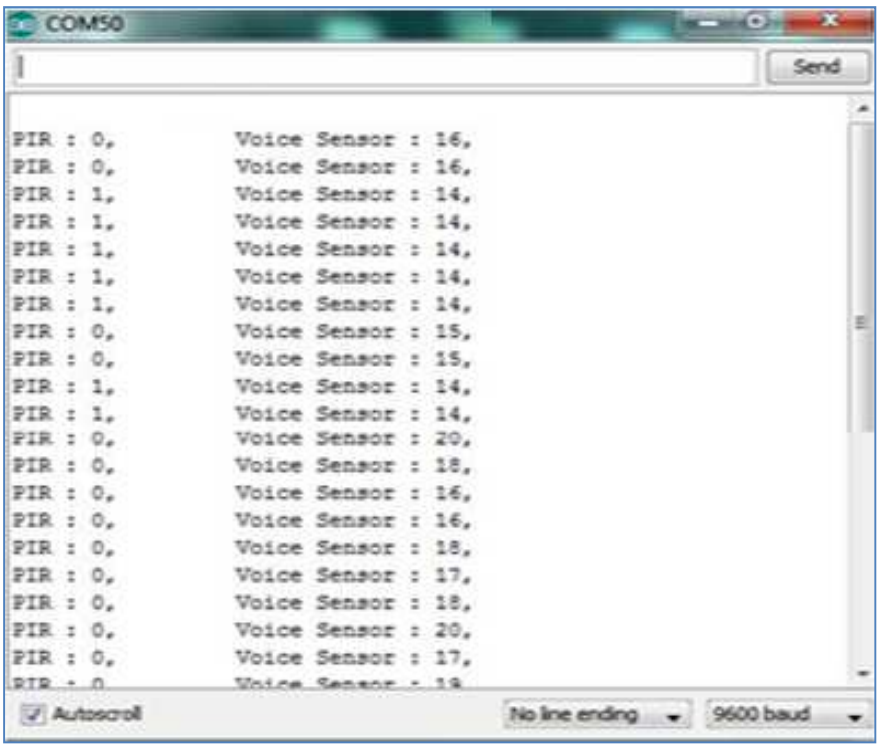


Fig. 3.4. *Data Collected with Arduino IDE.*

4. Conclusion

It can be concluded that the step-by-step wireless sensor network (WSN) building approaches highlighted in this study are instructive and serve as a veritable trainer platform for students desirous of WSN projects. It should be noted that the running of Experiment 3.4 consumes a lot of energy and would require constant replacement of the 9V battery. It is advisable to disconnect the battery when not in use. When the battery is low, communication is likely to be affected. With the understanding of these basics, further study can be carried out to learn power management tricks.

References

[1] Bokare, M. and Ralegaonkar, A. (2012). Wireless Sensor Network: A Promising Approach for Distributed Sensing Tasks, *Excel Journal of Engineering Technology and Management Science*, Vol 1 No 1.

[2] Faludi, R. (2011). Building Wireless Sensors Networks. ISBN 978-0-596-80773-3.USA.

[3] Guo, J.; Orlik, P.; Zhang, J.; Ishibashi, K. (2014). Reliable Routing in Large Scale Wireless Sensor Networks. *Journals of Ubiquitous and Future Networks (ICUFN).*

[4] Hariprabha , V. and Vasantharathna, S. (2014), Monitoring and control of food storage depots using wireless sensor networks, *International Journal of Industrial Electronics and Electrical Engineering,* Volume-2, Issue-6

[5] Hawbani, A. and Wang, X. (2013). Zigzag Coverage Scheme Algorithm and Analysis for Wireless Sensor Networks, *Network Protocols and Algorithms,* Vol. 5, No. 4, ISSN 1943-3581

[6] Huang,Y., Zheng, J., Xiao, Y and Peng, M. (2015). Robust Localization Algorithm Based on the RSSI Ranging Scope, *International Journal of Distributed Sensor Networks*Volume 2015, Article ID 587318, 8 pages

[7] Kurose, J., Lesser, V., Silva, E., Jayasumana, A. and Liu, B. (2003). Sensor Networks Seminar, CMPSCI 791L, University of Massachusetts, Amherst, MA, Fall.

[8] Onibonoje, M. O, Jubril, A. M., Owolarafe, O. K. (2012). Determination of Bulk Grains Moisture Content in a Silo Using Distributed System of Sensor Network. *Ife Journal of Technology.* 21(2), 55-59.

[9] Smith, A.G. (2011). Introduction to Arduino. *www.introtoArduino.com (2nd August, 2015).* ISBN-13: 978-1463698348

[10] Santoshkumar, Hiremath, V, and Rakhee, K.(2012). Smart Sensor Network System based on ZigBee Technology to Monitor Grain Depot. *International Journal of Computer Applications (0975 – 8887)* Volume 50– No.21

[11] Yang, T. A., Jain, D., Sun, B. (2008). Development of Emulation-Based Projects for Teaching Wireless Sensor Networks.*Journal of Computing Sciences in Colleges.*24, 2, 64-71. ACM Press, New York, NY, USA.

[12] Younis, M. and Akkaya, K. (2013). Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey, Dept. of Computer Sc. & Elec. Eng. Univ. of Maryland Baltimore County, Baltimore

[13] www.digi.com, Accessed 10th May, 2015

[14] www.seeedstudio.com, Accessed 12th May, 2015

[15] www.sparkfun.com, Accessed 10th May, 2015