Wireless Multi-Sensor Monitoring System Utilizing IEEE 802.15.4 Communication Standards for Water Leakage Detection

Neda Noorani
Department of Electrical and Computer Engineering
California State University of Northridge
neda.noorani.39@my.csun.edu

Abstract - This paper presents the design of a water leakage monitoring system which includes wireless networked sensors monitored from a Windows based PC. The purpose of such system is to detect possible water leakage for residential water pipes. Utilizing three small Printed Circuit Boards (PCB) s, data from remote sensors of different types (acoustic, pressure, temperature, flow rate, etc.) are collected and monitored on a PC for further processing and analysis. ZigBee technology, which is built on top of the IEEE 802.15.4 standard, is used for wireless communication in the network.

1. INTRODUCTION

Increases in residential plumbing, treatment and operational costs make the losses associated with underground water system leakage prohibitive. To combat water loss, many utilities are developing methods to detect, locate, and correct leaks.

In fact, accurate and efficient residential leak detection technology encompasses a wide range of benefits including but not limited to: economic benefits, increased knowledge about the distribution system, more efficient use of existing supplies, delayed capacity expansion, improved environmental quality, reduced property damage, reduced legal liability, reduced insurance and reduced risk of contamination[1].

Hence, this paper strives to delineate design of a water leakage monitoring system to detect possible water leakage for residential water pipes. To that end, the system collects and monitors data on a PC from remote sensors-located next to pipes for further processing and analysis to detect water leakage. Reliable communication within the network is provided by ZigBee technology, which is built on top of IEEE 802.15.4 standard.

More specifically, to collect and monitor data on a PC, three Printed Circuit Board (PCB) s, populated with the ZigBit 900 RF modules and a matched antenna are used. The ZigBit module featuring ultra small size and superior RF performance enables the board's wireless connectivity and facilitates its functionality a as a node in the ZigBee network. The PCBs include temperature sensor. In addition, these PCBs support standard extension connectors to connect to external sensors such as acoustic sensor, pressure sensor and etc. The PCBs are powered by one C-sized battery.

Importantly, this paper is organized as follows: Section 2 presents the basic concepts of Wireless Sensor Network (WSN). Section 3 elaborates on WSN standards including IEEE 802.15.4 and ZigBee standard. Section 4 elucidates the hardware component of the water leakage system. Section 5 discusses software component of the design. Section 6 explains how sensor data displays on PC in GUI format. The conclusion remarks are included in the end.

2. WIRELSS SENSOR NETWORKS

Wireless Sensor Network (WSN) typically consists of small spatially distributed devices to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration and etc. With WSN connectivity, data from remote sensors of different types are collected by central unit for further processing and analysis.

WSNs are less expensive and more flexible than wired monitoring systems. There are applications that become feasible only with WSNs because using wires between devices are too expensive or impossible at all. For instance, in many industrial, agricultural, military or ecological problems physical wiring is impossible or would create extreme disturbance for other operations. WSN, compared to other existing wireless technologies, is the only technology that targets simple communication with low data rates and low power consumption.

Each WSN node is typically equipped with:

- One or more sensors;
- A wireless transceiver including antenna or other wireless communications device;
- A microcontroller and memory to process received data and prepare data for transmission and execution of required networking tasks;
- A networking and application software which specifies networking protocols and application functionality; and
- An energy source, usually a battery.

ZigBee is a suitable standards-based wireless protocol technology that addresses the unique needs of remote monitoring, control and sensor network applications. The

ZigBee wireless standard enables broad-based deployment of wireless networks with low cost, low power solutions in a typical monitoring application.

ZigBee takes full advantage of the IEEE 802.15.4 physical radio specification and operates in unlicensed bands worldwide at the following frequencies: 2.400-2.484GHz, 902-928MHz and 868.0-868.6MHz. The ZigBee protocol carries all the benefits of the 802.15.4 protocol with added networking functionality. The ZigBee protocol was engineered by the ZigBee Alliance, a non-profit consortium of leading semiconductor manufacturers, technology providers, Original Equipment Manufacturers (OEMs), and end-users worldwide [2].

3. WIRELESS SENSOR NETWORK STANDARDS

It is extremely common to have standardized technologies in communication industry. Standard protocols make the technology more attractive for end users by its independence of a single vendor. Moreover, openness and large number of participants involved in standard development process increases technology reliability and safety. Furthermore, organizations which are responsible for standard specification are constantly improving their standards according to market needs. The most popular standards for wireless sensor networks are IEEE 802.15.4 and ZigBee, which are described in details below.

3.1 IEEE 802.15.4

IEEE 802.15.4 specified by Institute of Electrical and Electronics Engineers (IEEE), is a standard which specifies the physical (PHY) layer and Media Access Control (MAC) for Low-Rate Wireless Personal Area Network (LR-WPANs). It is the basis for the ZigBee, WirelessHART and MiWi specification, which attempts to offer a complete networking solution by developing the upper layers which are not covered by this standard.

The main features of IEEE 802.15.4 are network flexibility, low cost, very low power consumption and low data rate. It is developed for applications with relaxed throughput requirements which cannot handle the power consumption of heavy protocol stacks.

IEEE 802.15.4 defines two types of network node. The first one is the Full-Function Device (FFD) which contains the full set of IEEE 802.15.4 features. It can serve as the coordinator and as an end-device of a personal area network. It implements a general model of communication which allows it to talk to any other device. On the other hand, there are Reduced-Function Devices (RFD). These are meant to be extremely simple devices with very modest resource and communication requirements. Hence, RFDs can only communicate with FFDs and can never act as coordinators. Normally, FFD consumes more energy compared to RFD because it requires extra memory and processing power.

In terms of possible interconnections, networks can be built as either Peer-to-Peer or Star networks. Fig.1 illustrates IEEE 802.15.4 star and peer-to-peer topology. However, every network needs at least one FFD to work as the coordinator of the network.

In peer-to-peer, model an FFD can communicate to all other devices within its transmission range while an RFD can talk only to an FFD which is currently associated with. In Peer-to-Peer model, large spatial areas can be covered by a single network but complex packet routing algorithms are required. A Peer-to-Peer network can be self-organizing and self-healing. Advanced functionality of the Peer-to-Peer model is available only if an efficient network management protocol is realized on top of IEEE 802.15.4 stack.

In Star model, devices are interconnected in form of a star. Star network necessarily has the central node and all the network nodes (FFDs and RFDs) can directly communicate only to the coordinator. Star network is simple in set up and deployment. Moreover, in Star network, data forwarding is possible only by coordinator (two-hop only) and coverage area is limited by one-hop transmission range.

IEEE 802.15.4 standard specifies only the lowest part of OSI communication model, PHY layer and MAC sub-layer. PHY layer is the lowest level in communication model. The PHY provides services such as activation and deactivation of the radio transceiver, frequency channel tuning, carrier sensing, received signal strength estimation Received Signal Strength Indication (RSSI), Link Quality Indicator (LQI), error correction, data coding and modulation. The MAC sub-layer provides services such as data framing, validation of received frames, device addressing, channel access management, sending acknowledgement frames, device association and disassociation.

3.2 ZIGBEE STANDARD

ZigBee technology is a low data rate, low power consumption and low cost wireless networking protocol targeted towards automation and remote control applications. ZigBee Alliance and the IEEE decided to join forces and ZigBee is the commercial name for this technology. ZigBee is expected to provide low cost and low power connectivity for equipment that needs battery life as long as several months to several years but does not require data transfer in high rates.

ZigBee compliant wireless devices are expected to transmit 10-75 meters, depending on the RF environment and the power output consumption required for a given application, and will operate in the unlicensed RF worldwide (2.4GHz global, 915MHz Americas or 868 MHz Europe). The data rate is 250kbps at 2.4GHz, 40kbps at 915MHz and 20kbps at 868MHz.

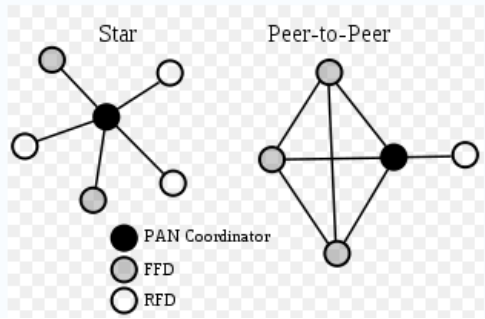

Fig.1 - IEEE 802.15.4 Star and Peer-to-Peer Topology [9]

IEEE and ZigBee Alliance have been working closely to specify the entire protocol stack. IEEE 802.15.4 focuses on the specification of the lower two layers of the protocol (Physical and Data Link layer). On the other hand, ZigBee Alliance aims to provide the upper layers of the protocol stack (from Network to the Application layer) for interoperable data networking, security services and a range of wireless home and building control solutions, providing interoperability compliance testing, marketing of the standard and advanced engineering for the evolution of the standard.

ZigBee standard specifies three different types of nodes that might be present in a ZigBee network: Coordinator, Router and End Device. Coordinator is the most capable device that forms the root of the network. Coordinator is responsible for configuring key networking parameters, network start, admission of other nodes and network address assignment. There is exactly one ZigBee Coordinator in each network. Coordinator should be connected to a steady reliable power supply source because of high processing power and inability to sleep. Only FFD in IEEE 802.15.4 terminology can act as a network Coordinator. Router passes messages from ZigBee End Devices to other Router or to the ZigBee Coordinator. Router is used to extend network coverage area and increase network reliability. End Device can talk only to the Coordinator or a Router. It cannot relay data from other devices. This relationship allows the node to be asleep a significant amount of the time, thereby; giving long battery life. End Device requires the least amount of memory, therefore, it can be less expensive to manufacture than a Router and Coordinator. ZigBee End Devices correspond to RFD in IEEE 802.15.4 standard [6].

Routers and End Devices enter the existing network by associating themselves with a node already present in the network. Only Coordinator and Routers can provide network access. ZigBee network hierarchy can be visualized as a tree with Coordinator being on top and End Devices being tree leaves. Each node that joins ZigBee network receives temporary 16-bit long network address. Communication on network level is performed based on this address while direct transmission between two neighboring devices is done based on MAC address.

ZigBee networks can be configured to operate in a variety of different ways to suit the application and environment. Supported topologies in ZigBee Network include: Star topology, Cluster Tree topology and Mesh topology.

In Star topology, using a single Personal Area Network (PAN) Coordinator, each node connects directly to the central Coordinator – all inter-node communications are passed through the Coordinator. Moreover, in Star topology, the network coverage area is limited by Coordinator transmission range but network is simple in set up and deployment.

A Cluster Tree network consists of a number of Star networks connected whose central nodes are also in direct communications with the single PAN Coordinator. Using a set of Routers and a single PAN Coordinator, the network is formed into an interconnected mesh of Routers and End nodes which pass information from node to node using the most cost effective path. Should any individual router become inaccessible, alternate routes can be discovered and used. Hence, Cluster Tree Network provides a robust and reliable network topography. Fig. 2 illustrates Cluster Tree topology.

A key component of the ZigBee protocol is the ability to support Mesh networking. In a Mesh network, nodes are interconnected with other nodes so that multiple pathways connect each node. Connections between nodes are dynamically updated and optimized through sophisticated, built-in mesh routing table. Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. Also, as nodes leave the network, the Mesh topology allows the nodes to reconfigure routing paths based on the new network structure. The characteristics of Mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes. Fig. 3 illustrates Mesh network topology.

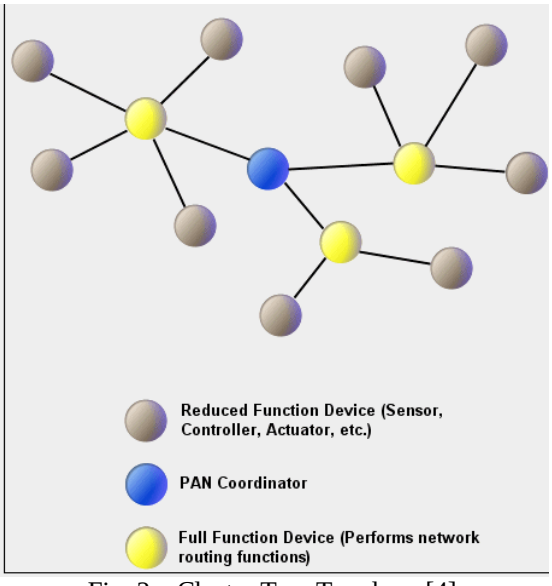

Fig. 2 – Cluster Tree Topology [4]

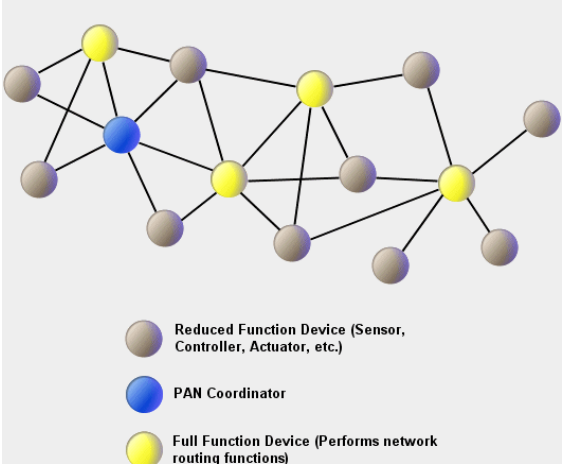

Fig.3- Mesh Network Topology [4]

4. HARDWARE COMPONENTS

Hardware component of the designed system is comprised of three Printed Circuit Boards (PCBs) which can be configured to operate as a network Coordinator, Router or an End Device by setting Dual In-Line Package (DIP) switches. Each of the PCBs contains the following components:

- ZigBit 900 module (MNZB-900-B0)
- Sensor
- Universal Serial Bus (USB) to Universal Asynchronous Receiver/Transmitter (UART) Bridge
- 20-pin Expansion Slot
- Power Supply
- 3 Push Buttons
- DIP Switches
- Software-Controlled LEDs
- Sub Miniature version A (SMA) Connector
- Silicon Serial for Unique Identifier (UID) Storage

In the following, the functionalities of these components are briefly described.

DIP Switches

DIP switches configure each node as a Coordinator, Router and End Device by using the codes downloaded to the microcontroller on PCB, according to the following table:

DIP switches			Role
1	2	3	
ON	OFF	OFF	Coordinator
OFF	ON	OFF	Router
OFF	OFF	ON	End device

ZigBit 900 Module

ZigBit 900 module (Part Number: MNZB-900-B0), as one of the most important PCB parts, is a low-power and high-sensitivity IEEE 802.15.4/ZigBee-compliant OEM module. Zigbit 900 module occupies less than a square inch of space. ZigBit 900 offers an unmatched combination of superior radio performance, ultra-low power consumption and exceptional ease of integration. ZigBit 900 contains Atmel's ATmega1281V Microcontroller and AT86RF212 RF Transceiver [4]. Fig. 6 illustrates MNZB-900-B0 Block Diagram. The module features 128K bytes flash memory and 8K bytes RAM. ZigBit 900 already contains a complete RF/MCU design with all the necessary passive components included. The module can be easily mounted on a simple 2-layer PCB with a minimum of required external connection.

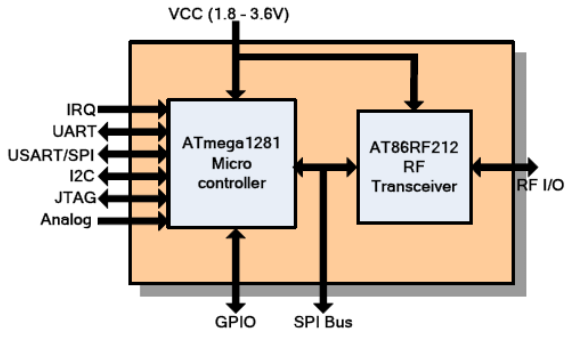

Fig.6 - MNZB-900-B0 Block Diagram[4]

Sensors

The PCBs include temperature sensor LM73CIMK connected to the I2C bus. In addition to the built-in onboard sensor, external sensors - to help detect water leakage in pipes - can be connected through serial port and Analog to Digital Converter (ACD) Input.

USB to UART Bridge

USB to UART Bridge provides seamless USB interface to any RS-232 legacy device.

20-Pin Expansion Slot

20-Pin Expansion Slot contains external ZigBit's interfaces including Serial Port Interface (RS-232), Universal Synchronous/Asynchronous Receiver/Transmitter (USART), Buffered Inter-Integrated Circuit (I2C) interface with Electrostatic Discharge (ESD) protection and voltage level translation, ADC_ inputs and General Purpose Input Output (GPIO).

Power Supply

The PCBs work with 3 volt C size batteries.

Push Buttons

There are three push buttons on PCBs. One of them is used as a reset button and the other two push buttons are controlled by software.

SMA Connector

PCBs are equipped with SMA Connector in order to attach an external antenna. The external antenna (Part Number: 17010.10) has frequency range of 2.35-2.5 GHz. Antenna is matched and tuned with taking into account all adjacent components, including the ZigBit 900 module shield, battery compartment and plastic legs. Any object approached or placed closely next to antenna might affect its performance.

Silicon Serial for UID Storage

The PCBs also contain Silicon Serial Number for UID storage (Part Number: DS2411R). UID is HEX value and 8 bytes. UID is used for setting unique MAC address of the node. To be connected with WSN network, each node should be identified with a unique MAC address. If MAC address is not defined by a UID hardware chip, the address of the node should be programmed manually. MAC address is utilized for identification of the node within the network.

The PCB can be connected to host PC via USB port, using USB 2.0 A/mini-B cable. No battery is required once a PCB is powered via USB. USB power is not stable enough, which in turn can affect transmission of power level or RF parameters.

End Device reads data from the onboard and external sensors. End Device follows a duty cycle, waking up occasionally to transmit the sensor data and sends the readings to router. Router also sends the data to Coordinator in packets. Coordinator sends the data to the PC's COM port. A special GUI application named WSNMonitor running on the PC displays the network topology and sensor data in an easy-to-interpret graphical form.

5. SOFTWARE COMPONENTS

The software part of the project involves programming of ATmega1281 microcontroller utilizing BitCloud Stack. BitCloud is a full-featured, professional grade embedded software stack from Atmel. BitCloud provides a software development platform for reliable, scalable, and secure wireless applications [3].

BitCloud internal architecture follows the suggested separation of the network stack into logical layers as found in IEEE 802.15.4 and ZigBee. Besides the core stack containing protocol implementation, BitCloud contains additional layers implementing shared services such as task manager, security and power manager and hardware abstractions such as Hardware Abstraction Layer (HAL) and Board Support Package (BSP) [5].

Next, different layers of BitCloud, BitCloud programming styles, BitCloud application structure, and the programming environment - Atmel AVR Studio, will be presented.

5.1 DIFFERENT LAYERS OF BITCLOUD

The following explains different layers of BitCloud:

- *Application Support Sublayer (APS):* APS is the topmost of the core stack layers. It provides the highest level of networking-related Application Programming Interfaces (APIs) visible to the application. [3]

- *ZigBee Device Object (ZDO):* ZDO enables main network management and functionality such as start, reset, formation and join. [3]

- *Multitasking Management Layer:* This layer mediates the use of Microcontroller (MCU) among internal stack components and user application. It implements a priority-based co-operative scheduler specifically tuned for multi-layer stack environment and demands of time-critical network protocols [3].

- *Hardware Abstraction Layer (HAL):* This layer includes a complete set of APIs for using on-module hardware resources (EEPROM, sleep, and watchdog timers) as well as the reference drivers for rapid design-in and smooth integration with a range of external peripherals (IRQ, TWI, SPI, USART, 1-wire) [2].

- *Board Support Package (BSP):* BSP includes a complete set of drivers for managing standard peripherals (sensors, UID chip, sliders, and buttons) placed on a development board [3].

5 .2 BITCLOUD PROGRAMMING STYLES

All applications based on the BitCloud SDK are written in an event-driven or event-based programming style. Event-driven programming or event-based programming is a programming paradigm in which the flow of the program is determined by events such as sensor outputs , key presses or messages from other programs. In fact, all internal stack interfaces are defined in terms of forward calls and corresponding callbacks. Each layer defines a number of callbacks for the lower layers to invoke, and in turn, invokes callback functions defined by higher levels. There is a generic type of user-defined callback which is responsible for executing application-level code called TaskHandler. APL_TaskHandler is the reserved callback name known by the stack as the application TaskHandler. The need to decouple the request from the answer is especially important

when the request can take an unspecified amount of time. For instance, when requesting the stack to start the network, the underlying layers may perform an energy detecting scan which takes significantly longer than we are willing to block for [8].

Apart from request/confirm pairs, there are cases when the application needs to be notified of an external event which is not a reply to any specific request. For this, there are a number of user-defined callbacks with fixed names which are invoked by the stack asynchronously. These include events indicating loss of network, readiness of the underlying stack to sleep, or notifying that the system is now awake [7].

5.3 BitCloud Application Structure

A BitCloud application has the following typical structure.

- Every application defines a single TaskHandler which contains-in its scope-the bulk of the application's code.
- Every application defines a number of callback functions contributing code executed when an asynchronous request to the underlying layer is serviced.
- Every application defines a number of callbacks with known names executed when an event is processed by the stack.
- Every application maintains global state which is a shared state between the callbacks and the TaskHandler.

The BitCloud stack provides an extensive set of configuration parameters which determine different aspects of network and node behavior. These parameters are accessible for application via Configuration Server interface (ConfigServer, CS for short).

In this project, the network and radio frequency performance of the hardware components is demonstrated by coding based on BitCloud API. The application code consists of the embedded firmware which supports functions of Coordinator, Router and End Device.

The application code is split up among the following C files:

- #include Directives
  Example: #include <taskManager.h>
- Function Prototypes
  Example:
  static void ZDO_StartNetworkConf
  (ZDO_StartNetworkConf_t *confirmInfo);
- Global Variables
  Example:
  AppState_t appState = APP_INITING_STATE;
- Application TaskHandler
  Eexample:

```
void APL_TaskHandler()
 {
    switch (appState)
       {    case APP_IN_NETWORK_STATE:
                    ...
              break;
              case APP_INITING_STATE: //node has
initial state
                    ...
             break;
             case
APP_STARTING_NETWORK_STATE:
                    ...
             break;
       }
}
```

- Implementation

The application code encompasses five major parts:

- *Configuration Server Interface (ConfigServer, CS for short) & CS Read/Write functions*: In this project, in order to perform parameter read/write procedure at run-time, the API functions, CS_ReadParameter and CS_WriteParameter are used. Both functions require parameter ID and a pointer to parameter value as arguments. Parameter ID identifies which CS parameter the function is applied to and is constructed by adding "_ID" at the end of CS parameter name.

- *Network Information and Join to Network*: In this project, network start procedure performs in 4 steps: First, configuring node parameters; second, specifying target network parameters; third, initiating network start request; and finally, receiving network start confirmation.

- *Data Exchange*: In order to perform data transmission between End Device, Router and Coordinator, first a data transmission request of APS_DataReq_t type is created. That specifies Application-layer Service Data Unit (ASDU) payload. Second, various transmission parameters are set and callback function (APS_DataConf) is defined. This callback function is executed to inform the application about transmission result. Also End Device is registered using APS_ReqisterEndPoint() function with an argument of APS_RegisterEndpointReq_t type. The argument specifies Endpoint descriptor (simpleDescriptor field) which includes parameters such as endpoint ID (a number from 1 to 240), application profile ID, number and list of supported input and output clusters.

- *Power Management*: In this project like other ZigBee networks, power consumption level is a major concern because the End device is powered only by battery. By using BitCloud API and switching between awake and sleep modes as well as turning off the radio chip, power consumption is reduced.

- *Hardware Control:* In this project the sensors are connected to microcontroller through UART, ADC and I2C port. The BitCloud API also provides an extensive support of these common HW interfaces. In order to enable communication over UART interface, application first configures corresponding UART port using static global variable of HAL_UartDescriptor_t type. Second, data reception over UART is configured for operation in callback mode. Moreover, UART settings is applied using HAL_OpenUart() function with argument pointing to global variable of HAL_UartDescriptor_t type with desired port configuration. Returned value indicates whether port is opened successfully and can be used for data exchange. When there is no more need in keeping UART port active application closes it using HAL_CloseUart() function. Reading data over the ADC and I2C is mostly the same as UART port, only the functions and Global variable type which is corresponding to ADC and I2C should be used.

5.4 Development Environment - Atmel AVR Studio

In this project, Atmel AVR Studio is used to develop custom applications based on BitCloud API. This multiplatform Integrated Development Environment (IDE) provides the options for editing source code, compilation, linking object modules with libraries, debugging and making executable file automatically. In AVR Studio, the development of an application is organized under particular project. All the necessary information about a project is kept in project file. Such files assigned to the AVR Studio have an *.aps extension, so they open in AVR Studio automatically when double-clicked. The easiest way to configure an AVR project is to use Makefile that is a plain text file which name has no extension. Makefile specifies compilation and linking flags. Makefile also specifies corresponding directories in order to include header files and to link the system object libraries. An illustration of the development environment is in Fig. 5.

After the program code is completed and complied with AVR Stuido, a Bootloader program is used to download the code to the microcontroller on board. Bootloader.1.1.0 is available on www.atmel.com website. To download the code to the PCB, first, the board should be connected to the PC via USB or serial port. Then Bootloader should be run. In command line, the image file (.scre with extension) and the COM port will be specified. Then reset button on the board should be pressed and released within approximately 30 seconds. If this does not happen, the booting process would stop. Then, Bootloader indicates the operation progress. Once an upload is successfully completed, the board would restart automatically. If an upload fails, Bootloader would indicate the reasons. Fig. 6 displays a typical graphical window of the Bootloader.

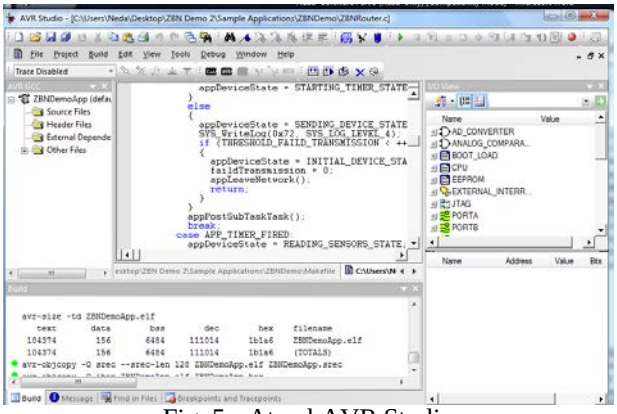

Fig. 5 - Atmel AVR Studio

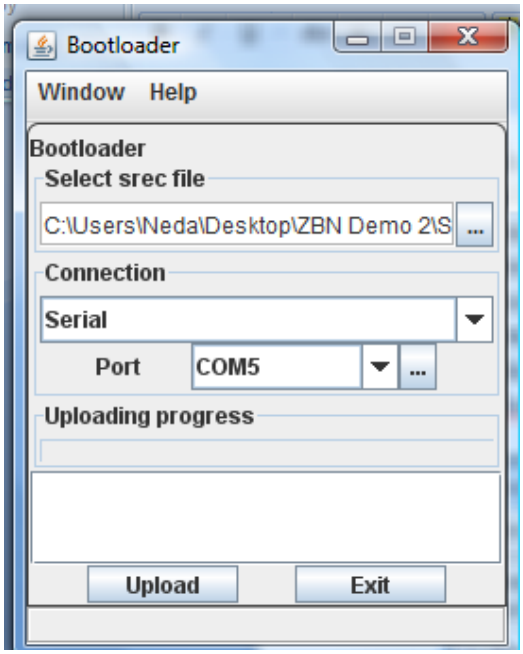

Fig. 6 – Bootloader

6. Wireless Sensor Network Monitor

In addition to the hardware and software components presented above, a PC-based graphical user interface (GUI) is necessary to display the map of the sensor nodes and the status of each node in the network. In this project, we customize an existing application by Atmel named Wireless Sensor Network (WSN) Monitor.

WSNMonitor displays the network graph in real time and updates it automatically as the nodes join or leave. The nodes are represented by icons.

In order to display sensor data on PC, the following procedure should be followed sequentially:

1. Connect the Coordinator to the PC.
2. Run WSNMonitor program on the PC.
3. Click on the connect button on the Main Toolbar.
4. Set Connection Properties on Connection wizard. Connection wizard allows user to set folder with protocol configuration files, connection type and properties.
5. Click the Finish button to connect to the specified port.

The working area of WSN Monitor consists of Network View and Node Parameter Table. Network View displays the entire network in a graphical form. Network View displays the network topology in real time, which helps the user monitor the formation and evolution of the network while the nodes join and send data. Network View is updated automatically while the nodes are discovered and while they join through the coordinator. The network is drawn in its star form, with the Coordinator node positioned in center of the view and its descendants ordered around. The links between the nodes are visualized by lines. Node Parameter table contains a table of parameters names and their values. This interface is illustrated in Fig. 7.

In addition, WSNMonitor can be customized with a protocol file. The protocol file is an XML document used in the Connection wizard when the Connection Properties are set to display sensor's data on PC, as delineated above. The protocol file introduces the parameters displayed on the Node Parameter table. For instance, the following protocol file code can be used to display the UARTSensor on the Node Parameter Table.

<value name="UARTSensor" type="int32" />

7. Conclusion

Wireless multi-sensor monitoring system utilizing IEEE 802.15.4 and ZigBee standards represents a low cost and low power consumption method for water leakage detection in residential pipes. In this system, three PCBs are utilized. Each PCB can be configured to operate as a network Coordinator, Router or an End Eevice. End Device collects data from pipe line and sends it to Router and Coordinator to be displayed on PC in GUI format. Moreover, application code based on BitCloud Stack is used to program the microcontrollers on PCBs.

Advantages of WSN system utilizing IEEE 802.15.4 and ZigBee standards make it applicable for monitoring in many industrial, agricultural, military or ecological projects. By using IEEE 802.15.4 and ZigBee standards, if a node that was retransmitting the data suddenly fails, wireless links between devices allow simple data rerouting over the other best suitable node and data is delivered to the destination via best suitable path.

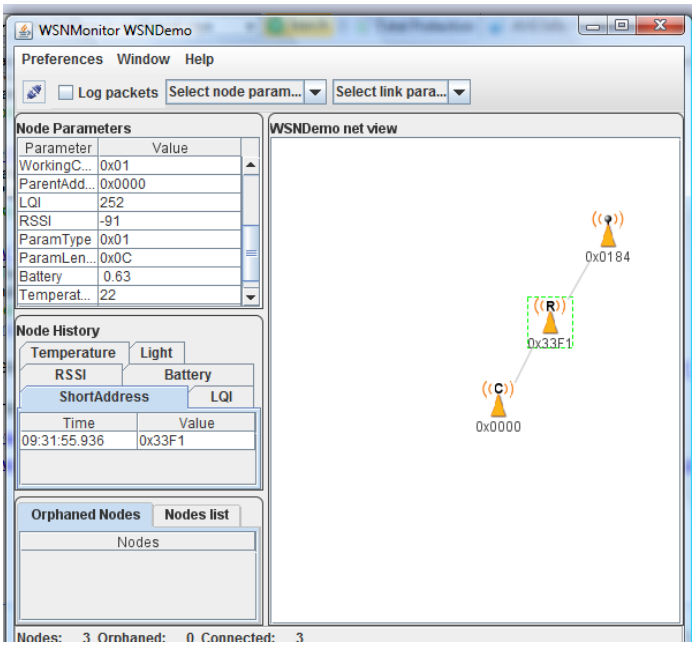

Fig.7- WSNMonitor WSNDemo

8. References

[1] http://www.nesc.wvu.edu/
[2] http://www.ferret.com.au/
[3] http://www.atmel.com
[4] http://www.meshnetics.com/
[5] AVR2051: BitCloud Stack Documentation
[6] http://www.ember.com/zigbee_index.html
[7]BitCloud™ IEEE 802.15.4/ZigBee Software. Product Datasheet. MeshNetics Doc. M-252~08
[8]BitCloud™ Software 1.0. BitCloud Stack Documentation. MeshNetics Doc. P-ZBN-452~02
[9]http://en.wikipedia.org/wiki/File:IEEE_802.15.4_Star_P2P.svg