Developing an Ultra-Low Power Remote Infrastructure Monitoring System

Ross Liederbach\textsuperscript{1}, Ashley Little\textsuperscript{1}, Flora Xiao\textsuperscript{2}, Cornel Zlibut\textsuperscript{3}, Guoqing Zhou\textsuperscript{1}, Chris Farnell\textsuperscript{1}, Brett Sparkman\textsuperscript{1}, Jingxian Wu\textsuperscript{1}, and Scott C. Smith\textsuperscript{1}

Department of Electrical Engineering, University of Arkansas, Fayetteville, AR\textsuperscript{1}
Department of Material Science and Engineering, University of Illinois, Urbana-Champaign, IL\textsuperscript{2}
Department of Electrical Engineering, Tennessee State University, Nashville, TN\textsuperscript{3}

riederb@uark.edu, arllittle@uark.edu, ffxxiao2@illinois.edu, czlibut@tn.state.edu, gzhou@uark.edu, cfarnell@uark.edu, bsparkma@uark.edu, wuj@uark.edu, and smithsco@uark.edu

Abstract— This paper develops an infrastructure monitoring system to wirelessly monitor vibrations that may cause impairment to structures. A wireless sensor network comprised of a fusion center and four sensor nodes allows for data to be transmitted from the sensors to the Fusion Center (FC), stored on an SD card, and later uploaded to a server via Global System for Mobile Communication/ General Packet Radio Service (GSM/GPRS). Another option is to read live data straight to the computer from the FC using a LabVIEW graphical user interface. The FC is controlled by SMS from the user’s cellular phone, with the option of manual override.

Keywords- Wireless Sensor Network, Arduino, Piezoelectric Sensor, XBee Radio, Vibration Monitoring

I. INTRODUCTION

Present day infrastructure such as bridges, buildings, roads and tunnels can use wireless sensor networks (WSNs) to test for structural fatigue caused by high vibrations, fire, flood, ice, etc. It is crucial to monitor these infrastructures for pending dangers. Patterns of these conditions can be monitored by using sensor nodes that transmit information to a fusion center (FC). This allows engineers to analyze structural damage and prevent future destruction. These WSNs will be needed for civilian and military use for extended time periods, requiring the system to use ultra-low power consumption, and to harvest energy from the surrounding environment. Several challenges were presented including a low signal-to-noise ratio and low maintenance. Surrounding environment conditions like noise and interference can lower efficiency of low power signals. The WSN was designed to withstand all forms of environmental conditions and harvest energy from the environment so that the sensors need little to no maintenance. Access to the data collected by the WSN was also included in the design. The power of the XBee has a limited distance transmission as it only allows up to a mile for transmitting data. XBee has mesh-network capabilities used for extending the range of device transmissions. This allows for data harvesting by storing data at a FC and transmitting data from the FC to the computer via GSM.

II. PREVIOUS WORK

Many studies have been conducted on vibrational testing with wireless sensor nodes for infrastructure health monitoring. Factors such as cost and power vary widely for every project. This project reduces cost and uses low power while harvesting energy from the environment. It is important to point out that this is on a much smaller scale than the previous works mentioned below.

The Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks [8] used 64 nodes to monitor the Golden Gate Bridge in San Francisco, CA. Each sensor node cost approximately $600 [8] and was also tied to a high cost of installation and maintenance. However, they each achieved a sampling rate of 1kHz and accuracy of 30 μG. The WSN used mesh networks due to its scale and a laptop was used as the base station. The information was then sent long range through the internet. Towards Wireless Sensor Networks for Railway Infrastructure Monitoring [9] used its own WSN, SENSORAIL, to monitor structural health and environmental security of a railway system. The WSN involved some overhead for the local wireless infrastructures in the form of ZigBee communication protocol. The information was then sent through a Wide Area Network (WAN) such as GSM/GPRS, UMTS/EDGE, and Fiber Optics geographic networks.

III. OVERVIEW OF COMPONENTS

After complete construction of the project, the node costs is approximately $90 and the FC with included components mentioned below equates to approximately $300.

A. Arduino Uno

The Italian-made Arduino Uno [6] is an open-source single-board microcontroller. A standard boot loader and programming language compiler run directly on the board. It is programmed using a language similar to C++ [5]. Fourteen input/output connector pins are exposed as female 0.1 inch headers, allowing the Arduino to interface with accessories, called shields. Some shields may be able to communicate directly over pins while others use the IC serial bus. The
Arduino Uno uses an Atmel AVR 8-bit ATMega328P processor [1]. For USB to Serial conversion the Arduino uses an ATMega16U2 chip. The Arduino Uno functions on a 5V operating voltage, with a 7-12V recommended input voltage. Also, the Arduino includes a 16MHz crystal oscillator to provide a stable clock signal.

Along with its processing power, the low power consumption and shield compatibility makes the Arduino Uno an optimal choice for the fusion center. With several shields available for the Arduino, and with a basic programming language, the Arduino Uno is extremely versatile and user-friendly.

B. Series 1 XBee

The Digi International Series 1 XBee [3] uses the IEEE 802.15.4 protocol. XBees allow for easy communication between microcontrollers, computers, or anything with a serial port. These modules have 8 digital I/O pins (with 6 being capable of analog) and a built-in antenna. XBees are programmed using AT commands. They are compatible with X-CTU software, which is used for testing and configuration of XBee devices. The Series 1 XBee is ultra-low power and has long distance reliability allowing it to transmit up to 100’ indoor and 300’ in outdoor settings. The Radio Frequency (RF) Data rate is up to 250,000 bits per second (bps). The Serial Interface Data rate travels between 1200 bps and 250 kbps. Power requirements are between 2.8 and 3.4 V for supply voltage. The Series 1 XBee modules have point-to-point or point-to-multipoint capabilities.

C. SM5100B GPRS Shield

The RF part of the GPRS shield converts into baseband for receiver chains and translates base band signals into the RF spectrum. This module has 60 connector pins and is connected to the SM5100B evaluation board. The GSM module connects directly to a Quad-band Cellular Duck Antenna. The operational voltage is between 3.3V and 4.2V with a recommended 3.6V power supply. The available frequency bands are EGSM900, GSM850, DCS1800, and PCS1900. In America and for this project the GSM850 was used. It operates on a 5V connection, draws ~74 mA, during the sleep cycle, the node automatically adjust the current to account for the varying charge of the solar panel. This module has automatic recharging capabilities as well as a user output power switch. The batteries used in this project are 3.7V, 3000mA/Hr batteries. This means that they should supply 3000mA of current for one hour. Paired with the Photovoltaic Module, the Li-Po Rider provides an ultra-low power solution for the WSN.

D. LabVIEW Graphical User Interface

LabVIEW is system-design software allowing scientists and engineers to program tools on a GUI for measurement and control of systems. LabVIEW allows the user to program graphically in a language called “G”. This programming method allows the user to wire graphical icons together while directly compiling to the computer, greatly simplifying the debugging process. G Programming is much more intuitive because it allows scientists and engineers to think and problem solve visually. One benefit of the built-in compiler is the broken run arrow in the toolbar that does not allow the user to run the program if there is an error. LabVIEW is able to connect to the serial port on the Arduino Uno to read in live data. Also, it has the capability to read or write to spreadsheets. Because of the intuitive programming language and applicable measurement tools provided, LabVIEW offered a very straightforward approach to designing a graphical user interface (GUI) system.

IV. System Design

A. Nodal Modules

Nodes are enclosed in 4x2x1” dimensional boxes. As shown in Figure 1, they are packaged to hold an XBee wired to the ATMega328P, the 3000mAh battery and the Li-Po Rider. This gives the nodes their own sources of power, and the ability to harvest energy from the environment. The sensing pin, ground, and voltage controlled by the MOSFET, are all off-board for easy connection to piezoelectric sensors or sensors that require voltage. The MOSFET/microcontroller combination allows for less power consumption by the sensor devices and the XBee. All eight I/O pins of the XBee may be made external to the node for access. An external switch allows the user to manually turn the sensors on and off to conserve power. The ATMega328P analyzes the commands from the FC, which in turn may shut off current entirely to the XBee radio, or start transmissions, thus conserving power. This switching of power to the XBee and sensors is controlled by the MOSFET in conjunction with the microcontroller. Currently, during the sleep cycle, the node draws ~13 mA, and while continuously transmitting, the node draws ~74 mA.
1) Li-Po Rider

The Li-Po Rider [7] is extremely useful in its ability to utilize external energy, and store that energy in a battery. The Li-Po Rider also allows for switching 5V on and off from the battery to power another system. With the Li-Po Rider board, managing power at each node is simplified, with each node having its own solar panel and 3000mAh battery.

2) ATmega328P

For controlling the XBee radio, with ultra-low power consumption in mind, pre-programmed sleep modes on the XBee did not work. These pre-programmed modes on the XBee radio created problems with the Fusion Center interface; therefore, a microprocessor, the ATmega328P, allows for more specialized and specific communication between the node and the FC. The ATmega328P is first boot-loaded with the Arduino boot loader, which is shown in Figure 2.

Figure 2. Bootloading ATmega328P

After being bootloaded, this chip may be programmed using the Arduino IDE. Programming the chip involves having a digital pin written high and low based on commands sent from the FC. This digital pin is connected to a PNP mosfet, which in turn powers the XBee on and off.

3) XBee

Original programming of the XBee sets the baud rate, and Personal Area Network ID (PAN ID) to the same as the XBee at the fusion center, sets the destination address to the coordinator XBee’s address, and sets the sampling rate and “samples before TX” to A and 0, respectively (A is 10 in hex and represents a 10ms or a 100Hz sample rate). Before plugging the XBee into the system, the I/O pins must be configured as needed to digital, or ADC. These values are stored in the XBee and are not reset when powered off.

For the system of XBee and ATmega328P to work properly, the XBee must turn on periodically to check for a command from the FC. A sleep cycle consisting of “Wake” and “Sleep” stages is implemented to accomplish this. If a command is sent from the FC during the “Wake” stage of the sleep cycle, the ATmega328P allows the XBee to stay on. Once the XBee is on, other commands may be sent, such as a command to start transmitting. In the case that the “Transmit” command is sent, the ATmega328P will enter the XBee command mode and reprogram the “samples before TX” value to “A,” which again means 10, and therefore begins transmitting after acquiring 10 samples. This method allows for longer “Sleep” stages, however the longer the sleep stage, the longer it will take for all nodes to wake up. Minimizing the “Wake” stage of the sleep cycle would lower overall power consumption, but this again could affect the time it takes to wake up since the node might not receive the wake command during its “Wake” stage. All XBee AT Commands can be altered as shown in Table I [3].

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATID</td>
<td>PAN ID</td>
<td>3332</td>
</tr>
<tr>
<td>ATDL</td>
<td>Destination Low Address</td>
<td>40492D6A</td>
</tr>
<tr>
<td>ATBD</td>
<td>Interface Data Rate</td>
<td>7-115200bps</td>
</tr>
<tr>
<td>ATIR</td>
<td>Sampling Rate</td>
<td>0</td>
</tr>
<tr>
<td>ATIT</td>
<td>Samples before TX</td>
<td>A</td>
</tr>
<tr>
<td>ATD0-ATD7</td>
<td>I/O Settings</td>
<td>1-NA 2-ADC 3-DI 4-DO LOW 5-DO HIGH</td>
</tr>
</tbody>
</table>

4) Abilities/Sensor capabilities

The ATmega328P microprocessor chip is the same as in the Arduino Uno, and has a lot of processing power that is unused in this application. Alternate and additional commands may be analyzed and carried out by the ATmega328P, but in this project, only sleep, wake, and transmit commands are used.

Any sensors that are powered by 5V or less, and output between 0 and 3.3V, are capable of being used with the sensor nodes. External circuitry is helpful in scaling voltages of sensors down to what is readable by the XBee radio.

Using Series 1 XBee radios bring the advantage of higher sampling rates, up to 1KHz, and simpler setup than Series 2 radios, which are only capable of sampling rates up to 20Hz. The Series 1 radio may store up to 46 analog samples before transmission in its buffer, which allows for expansion in the number of sensors the node is capable of supporting, and allows for lower transmission rates.

B. Fusion Center

The fusion center is comprised of several elements including: SD Wireless Shield, S1 XBee Radio, SD card, two Arduinos, GPRS Shield, 12V power supply with a voltage regulator, and a multiplexer. All the components are enclosed in an 8x6x4” box, as shown in Figure 3. The box has two
switches controlling system power and a live or SD data collection and a mini USB plug for live data collection. The FC is the center piece that allows the whole system to cooperate together. The FC system can also be broken into two distinct systems sustained by individual Arduinos. The first Arduino is dealing with XBee and SD communication, while the second Arduino deals with the GPRS communication. Together they allow for a larger variety of integrations and improvements.

![Figure 3: Wiring of Fusion Center](image)

1) System One (Arduino/XBee/SD)

The Arduino Wireless Shield allows the Arduino board to communicate through the XBee with all the sensor nodes. In this system the microcontroller is programmed to deal with the XBee and SD card for collecting the data from all the sensor nodes and saving it to the SD card in separate files depending on the sensor node. If the live data collection switch is “On” then all the data will be output to a new serial port for viewing it in LabVIEW.

![Figure 4: Arduino with the XBee/SD Shield](image)

a) XBee

The XBee at the FC is working as the coordinator for the entire wireless network on the PAN. The XBee is programmed in X-CTU, which includes the networking & security, RF interfacing, serial interfacing, I/O settings, diagnostics, and AT command options. One important difference in the coordinator XBee is that its destination address is required to be set to a broadcast address over the PAN so that each node receives the commands sent from the FC. Commands like wake, sleep, or transmit are all broadcast. The RF interfacing should set the “select/read transmitter output power” value to its highest level for efficient communication between the XBees in different environments. Furthermore, the clear channel assessment threshold is set to 44dBm so that if the modem detects energy above the threshold it does not transmit, which limits the loss of data samples. The serial interfacing settings were modified to 115200 interface data rate (BD) [3]. This option allows adding more sensor nodes to the wireless network without lag or losses in the data received.

b) SD

The SD is formatted with FAT32 file system with one partition on it since Arduino can only support this system. The communication between the microcontroller and the SD card uses SPI, which takes place on digital pins 11, 12, and 13. Additionally, another two pins must be used to select the SD card for input and output of data, which in this case were digital pins 4 and 10. The SD is used to hold the received data until it is being transmitted to the remote server through the GPRS; afterwards the data is removed to prevent duplication or retransmission of the same data.

2) System Two (Arduino/GPRS)

The Arduino GPRS/GSM Shield is a physical add-on to Arduino that allows send/receive SMS and voice calls, but also establishes TCP communication [4] over the broadly spread GPRS network. In this project the GPRS/GSM shield is used for sending and receiving SMS commands and confirmations while connecting to a socket on a remote server to send data. In this system the microcontroller is programmed [4] to communicate with the GPRS/GSM in order to establish socket and SMS connection. It is fully commanded using AT commands [2] over the serial port connection from the microcontroller. The GPRS uses a SIM card from T-mobile which monthly or annually can be recharged with the data plan desired by the user.

The microcontroller on this system also controls a dual 4-channel analog multiplexer/demultiplexer with common select logic for the ability to change between serial ports on the Arduino, and three digital pins for communication between the two Arduino systems.

![Figure 5: SM5100B GPRS Shield/Arduino](image)
3) FC Mechanism

The two Arduino systems at the FC should cooperate together in order for the entire project to work properly. When the entire system is turned on, one of the microcontrollers is initializing the SD and XBee while the other microcontroller is setting up the SMS and server socket connection based on the server specified. Once the initialization and socket connection is ready, a SMS confirmation is sent to the specified user. Now the user is able to text a command, as shown in Table II. If the GPRS receives the #ON command, it sends back a confirmation to the user and lets the XBee/SD Arduino know through a digital pin to broadcast the wake command. After all the nodes respond with an acknowledgment of “On”, the coordinator gives an order for transmission. Now the microcontroller checks to see if the live data is switched “On”. If the live data switch is not “On” then it automatically writes the data to the SD card. If the live data switch is “On” it then can be viewed in the LabVIEW interface. Once the user sends the #OFF command, the GPRS again sends confirmation back to the user and then lets the XBee/SD Arduino know to end the transmission by sending the sleep command. Once the nodes are in sleep mode and SD has all the data, the GPRS lets the user know that data is ready to be transmitted to the computer/TCP server. Therefore, the XBee/SD Arduino takes over the TX/RX pins of the GPRS through a high-speed Si- gate CMOS device for ease of uploading the data from the SD card to the serial port of the GPRS. This data is continually being sent to the computer/TCP server until no more data is left. Once the data is finished, the TX/RX pins of the GPRS are switched back to the previous Arduino and confirmation of the completed data is sent back to the user. The SD files can now be deleted to make room for new files or they can be renamed to be kept for future analysis.

<table>
<thead>
<tr>
<th>SMS Command</th>
<th>Command Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ON</td>
<td>Turn Sensor nodes ON</td>
</tr>
<tr>
<td>#OFF</td>
<td>Turn Sensor nodes OFF</td>
</tr>
<tr>
<td>#OR</td>
<td>Reset GPRS</td>
</tr>
</tbody>
</table>

4) FC Optional Improvements & Capabilities

The FC can also be improved in many ways based on preferences of the user. For example, if different types of sensors are needed for monitoring the infrastructure, they can be integrated into the FC. Sensors like humidity devices and barometers were tested on the FC.

C. Node/FC Interface

Figure 6 shows the basic schematic of the communication between cell phone, computer, fusion center, and nodes. In the fusion center, two Arduino Uno boards are housed, each with their own shield and separate tasks. The Arduino with the GSM/GPRS shield handles all incoming commands from the cell phone and sends back confirmations via SMS. This shield also handles uploading data to a TCP server where it can be accessed through a computer. Because this Arduino receives commands, it passes the commands along to the Arduino coupled with the SD Wireless shield. This second Arduino directly communicates with the sensor nodes, giving the nodes commands to turn on and transmit data, or enter the sleep cycle. Once the command is sent to wake the nodes, the fusion center waits for a confirmation from each node to send the command to begin transmission. After the command to transmit is sent, the processor at each node enters into the XBee command mode and reprograms the “Samples Before TX” value to “A”, or 10 samples before transmitting. The transmit command shall only be sent once from the fusion center, and broadcast to all the nodes to prevent interference while the processor is entering command mode. Finally, when the transmission is stopped and the nodes enter the sleep cycle, the Arduino with the GSM/GPRS shield begins forming packets to send over the TCP server, and the Arduino with the SD shield writes the data, packet by packet, to the GSM/GPRS shield. Once all the data is sent, the files on the SD card are deleted, but are accessible through the TCP Server.

D. Server

TCP File is an online open-source program that allows for a socket connection and listens for incoming data. It then takes this data and stores it into a .txt file.

The data is saved to a TXT file, which is then sorted by a MATLAB program. The MATLAB program reads in the file as a string, identifies all occurrences of sensor filenames, and sorts each sensor’s data into its own column of a master matrix. The sorted matrix is given a size at the beginning of the program to minimize runtime, and after all the data is sorted into it, all rows of all zeroes are deleted, and the master matrix is written into a .csv file. The LabVIEW GUI reads in the .csv file and displays it in a comprehensible format.

E. LabVIEW Interface

For this project, two graphical user interfaces were used. Since there were two different methods for inputting data, two different interfaces were needed. A GUI was designed for reading in data from a data file, such as from an SD card. The
other interface focused on reading live data from the COM port that the FC connected to.

1) Reading Data Files

Figure 7 shows the front panel of the GUI for reading in data directly from a .csv data file. Detailed below in Table III are the functions for each designated part on Figure 7.

Before the user presses the run button, they must choose a file to read from by clicking the folder represented as number 4. This will pull up a dialog box allowing the user to select what file they would like to import. If the user presses the run button without choosing a file, it will pop up with an error and allow them to try again. Once the interface is running, the sensors will show the last data point collected on the .csv file, shown at number 5 in Figure 7.

On the waveform chart, number 6 shows the Y-axis is auto-scaled in order to display all the voltage points. The time-voltage chart allows the user to scroll backward and see previous plot points. Number 8 offers the user the option to zoom in on the data values as shown in Figure 8. When the user is finished using the program, they should press the stop button located at point 2 in Figure 7.

2) Reading from COMports

In this version of the GUI, shown in Figure 9, the user can see live data from the COM port. The FC will be connected to a USB port in the computer. The user will select the COM port, as shown as number 1 in Figure 9. Once the COM port is selected, the user can then run the program and see the plotted data as it comes in. The chart will scroll with the new points as well as auto-scale the Y-axis to ease the analysis process. The GUI also is programmed to expand into a larger chart to the right of the smaller charts to make it easier to see the graphed points, which can be seen at number 5 in Figure 9.

V. RESULTS AND CONCLUSION

Testing was done in a three part phase. First, one wireless sensor node was built and tested against a wired sensor. Then, more nodes were built and tested as a final system without the GPRS. Finally, the GPRS was connected to the final system.

1) Comparison to Wired Sensor Network

Once the first sensor node was built, an accelerometer was attached to the wireless node so that the waveform could be compared to that of a wired sensor. It was tested on a cantilever beam in a lab at the Engineering Research Center (ENRC) located in Fayetteville, AR. Results above 600mG were successful. When the signal dropped below this level, much noise was picked up resulting in faulty data. After more
testing, it was determined that either the voltage-scaling circuit for the signal caused the noise or the accelerometer was defective at low voltages.

2) Field Test
In order to test the final system without GPRS, a field test was completed at a local bridge in Fayetteville, AR. The sensors were placed underneath the bridge at its joints, which seemed the most optimal location for reading vibrations from the moving vehicles and other outside factors. To begin with, data was collected live to make sure the sensors were working properly. Then data was collected and stored in the SD card for an hour with four piezoelectric sensors connected to the four nodes. The data successfully transmitted to the FC on the ground approximately 15 meters from the sensors. All components worked as expected, and the data stored on the SD card was analyzed later at the lab. One issue is that the .txt file showed different amounts of data for each sensor node in the one hour period. This may be caused by outside interference that corrupts some of the transmissions from the XBees to the FC.

3) GPRS/GSM
Once the GPRS was connected to the FC, all commands were transmitted through SMS communication. After the data was stored on the SD card, it was then transferred through the TCP/IP server using the GPRS. The server waited for the data to store into a single .txt file, which later was converted using MATLAB into a .csv file for LabVIEW. This effectively transmitted the data for further analysis.

4) Future Work
There are several things that can be improved upon in this system. For example, a variety of sensors can be added to the system such as a water sensor to test for flooding, a barometer to test for air pressure, temperature, and altitude, or a humidity sensor. Also, more nodes can be added to the system; however, the program on the FC does not account for additional nodes. Thus, the program could be altered for easier expandability of the network, for more robustness, and for higher efficiency.

REFERENCES