WIFIRES: WIRELESS FIRE SENSING

Jon Brackbill[1], Andrew Rupert[2], Nate Keen[3], Shaun Mason[4] and Srinivasa Vemuru[5]

[1]*Ohio Northern University, Ada, Ohio; Email: j-brackbill@onu.edu*
[2]*Ohio Northern University, Ada, Ohio; Email: a-rupert@onu.edu*
[3]*Ohio Northern University, Ada, Ohio; Email: n-keen@onu.edu*
[4]*Ohio Northern University, Ada, Ohio; Email: s-mason.1@onu.edu*
[5]*Ohio Northern University, Ada, Ohio; E-mail: s-vemuru@onu.edu*

1. PROBLEM STATEMENT

In the confined conditions of the modern urban environment, the ability to quickly recognize and respond to environmental dangers is becoming increasingly more difficult. Any problem has the potential to have a severe effect on the inhabitants of that area if left unchecked for even a short length of time. One of the most devastating threats to urban areas is fire. On average, the 500,000 residential fires reported to fire departments each year lead to the deaths of more than 4,000 people. Typical residential fire detection systems only operate with the goal of immediate personal safety. This helps save lives in the building where the fire originates, but provides no warning for residents of neighboring buildings. Another shortcoming of these systems is that they rely on residents to relay fire information to the local fire department and if no one is in the vicinity when the fire occurs there is potential for severe property loss. Fires cause four billion dollars of damage annually in the United States. This amount could be greatly reduced if fire response teams were automatically notified of when and where a fire starts as well as the real-time status of the fire.

We are proposing to build an automatic and cost-efficient fire detection and monitoring system for wireless-enabled cities as our senior design project. The system will alert emergency personnel via the internet when potentially hazardous conditions arise. The warning system will consist of small wireless enabled sensor modules that will be positioned around the city and periodically transmit data back to a central server at the city's emergency response center. This system will allow authorities to constantly monitor city conditions and should lower response time since officials will no longer depend on citizens alerting fire emergency personal. The hardware and software will be modular enough to allow for further expansion of monitoring capabilities as necessary. With this type of system in place cities could significantly improve their response time to fires and, as a result, reduce the loss of life and property caused by fires.

2. OPERATIONAL DESCRIPTION

This system is a distributed remote sensor network with each remote node gathering data and communicating with a central server. The sensors will constantly monitor their environment and notify the onboard processor when an event is detected. The onboard processor will process the data from its sensors and prepare it for the central server to store. That data will be stored in XML format and transmitted from the sensors over a wireless interface for storage in a central database for use in the main program. An operational block diagram of the entire system is shown in Figure 1. The server side data collection is shown in Figure 2. This flowchart depicts the flow of data into the central server which it then processes and stores it into a MySQL database. This data will be used for the main C# driver application and may possibly be provided to authorities through a web interface. The main C# application flow chart is show in Figure 3, illustrating the data analysis and user components of the application.

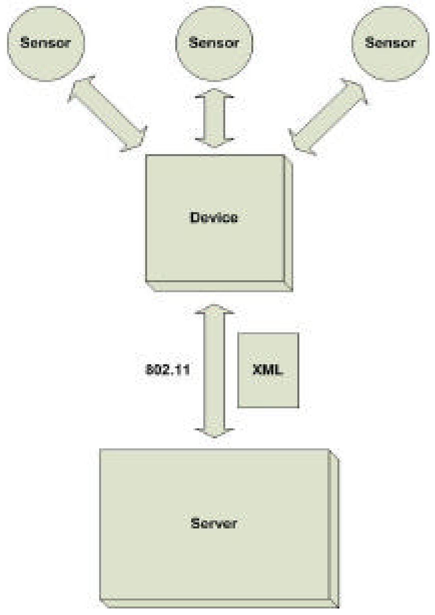

Figure 1: System Block Diagram

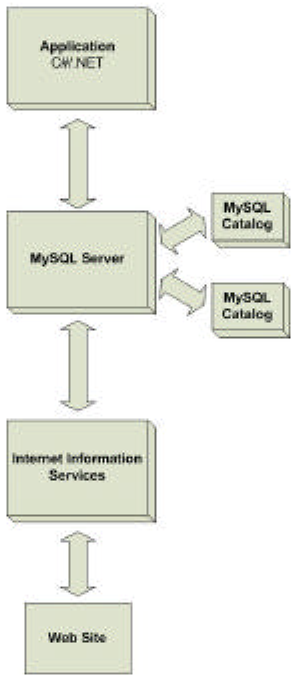

Figure 2: Server Side Data Handling

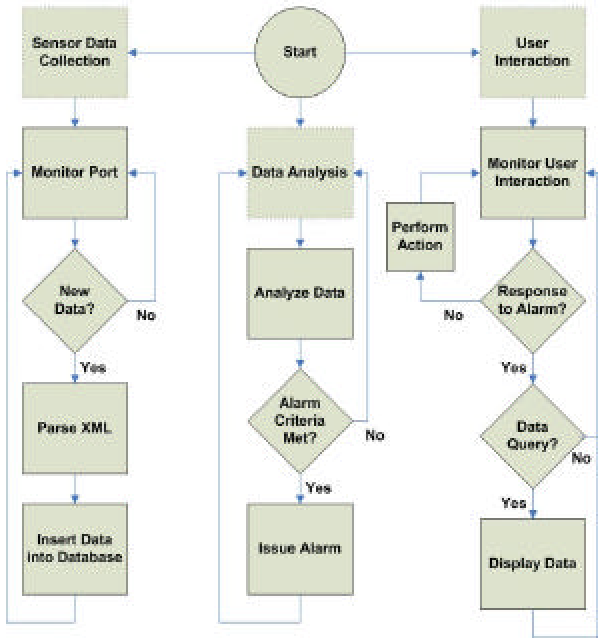

Figure 3: C# Server Application Flow Chart

3. DESIGN CONSIDERATIONS FOR THE SYSTEM

The design implementation for this project involved a number of factors in the hardware and software aspects. In this section, we present some of the options available and factors that led to our final choices.

3.1. COMMUNICATION PROTOCOL

Communication between the remote devices and the central server is an important decision. The Wi-Fi, or IEEE 802.11g protocol was considered because of the vast amount of hardware available for the protocol and the introduction of city wide Wi-Fi in Philadelphia and San Francisco. Wi-Fi is normally used in a point to multipoint system where an access point communicates with all devices within its range. The range of Wi-Fi is based on the power of the transmitter, and the data rate for 802.11g is 54Mb/s with an average throughput of 24.7Mb/s and fallback speeds of 11Mb/s, 5.5Mb/s, 2Mb/s, and 1Mb/s when signal strength to the device drops (IEEE, 2005).

3.2 SENSORS

There are two popular methods of smoke detection currently in use. One method uses a photoelectric chamber that detects the smoke entering the chamber by using beams of light. Instead of relying on smoke to block the beam of light the light transmitter and sensor are positioned at right angles to each other. When the smoke enters the chamber the light will scatter off from the smoke particles and hit the sensor (Figure 4). Since the sensor relies on light reflecting off of the smoke particles it can sense a small amount of smoke and is well suited for a smoldering fire (Marshall, 2005).

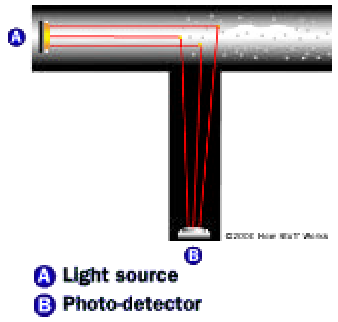

Figure 4: Diagram of Photoelectric Chamber.

The second smoke detection method is to use an ionization chamber and ionizing radiation to detect the smoke. This method works by using a minute amount of the radioactive element americium-241 (1/5000th of a gram). This small amount of americium is used to generate the alpha particles needed in the detector. The ionization chamber contains two metal plates, one connected to a positive charge and one connected to a negative charge, separated by a gap of air. The alpha particles ionize the oxygen and nitrogen in the air thus creating free electrons and positively charged particles. The two different charged particles are attracted to the corresponding plate of opposite charge (Figure 5). This creates a current that the detector monitors. When smoke enters the chamber it disrupts this current by neutralizing the ions. The detector senses this current drop and sends out an alarm. The ionization implementation is generally better than its photoelectric counterpart at detecting fast flaming fires. Although the ionization detector contains radioactive material this method is relatively safe because alpha particles can not pass through your skin or through more than a few centimeters of air (Marshall, 2005).

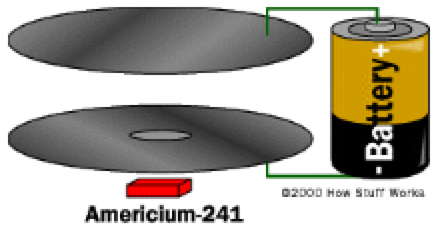

Figure 5: Layout of Ionization Chamber

Since both of these smoke detectors are relatively cheap, and are good at detecting different kinds of fires they can both be used in the device simultaneously. The safety added by having two smoke detectors present in the device is well worth the small increase in cost. Thus both smoldering and full blown fires can easily be detected.

Carbon-Monoxide (CO) is put out by the fire and can be detected by using an electrochemical CO sensor. The sensor consists of a diffusion barrier that is porous to gas, a reservoir of electrolyte, a sensor electrode and a counter electrode. The gas enters the sensor through the diffusion barrier and reacts at the surface of the sensing electrode. The sensing electrode is made to catalyze a CO reaction. When the gas diffuses into the sensor it becomes oxidized at the sensing electrode. This chemical reaction causes the potential of the sensing electrode to either rise or fall. The amount of CO present is proportional to the amount of current generated by the sensor. Therefore, a CO sensor is also included in the smoke detection unit.

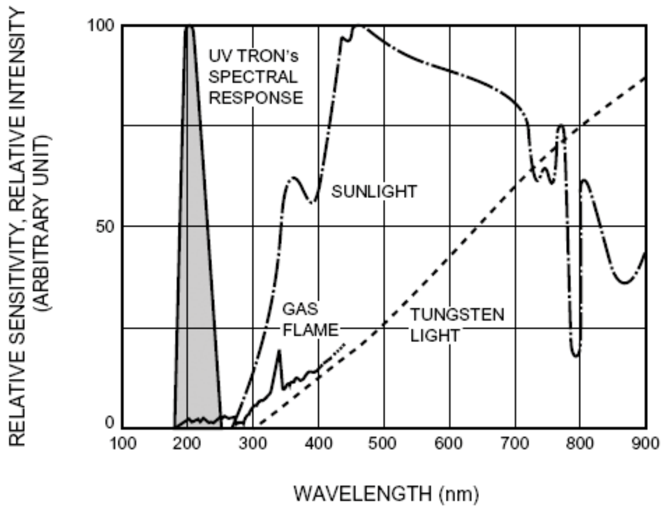

Figure 6: Wavelength detected by the Hamamatsu R2868 (Hamamatsu 1998).

The final sensor in the design will detect UV frequencies of light that are only present in a fire and almost nowhere else. These kinds of sensors can be relatively small and accurate making them a good choice for the fire detection device. The Hamamatsu R2868 UV detector only responds to light that is between 185nm and 260nm in wavelength. This cuts down on the false alarms since the sensor cannot detect most of the wavelengths that are not related to a flame (Figure 6). The R2868 has a very high sensitivity to detect flames (Hamamatsu, 1997).

3.3. HARDWARE PLATFORM FOR REMOTE DEVCE

The hardware platform evaluated for this application was an embedded x86 system on chip. The specific device examined was the eBox II and the relevant system specifications are shown in Table 1.

Table 1: eBox II Specifications (eBox 2005)

CPU	200 MHz Vortex86 system on chip
Memory	128MB SDRAM onboard 32MB Flash Disk
Ethernet	10/100Mbps Ethernet
I/O	1 LPD printer port 3 USB ports 1 Serial port
Power	+5V@3A

The eBox is a powerful system in terms of computing power and provides plenty of room for future expansion. It has the capability to run any version of Microsoft's Windows because it is based on an x86 processor. This platform also uses an Atmel microcontroller on the serial port to collect data from the sensors and feed it into the onboard program.

3.4. REMOTE DEVICE OPERATING SYSTEM

The x86 device can run any Windows operating system, and the two versions of particular interest for this embedded application are Windows XP Embedded and Windows CE. Windows XP Embedded is identical to Windows XP Professional but it offers all of the 12,000 system components as removable components. Of the 12,000 packages approximately 9,000 are device drivers and 3,000 are core operating system components. Since Windows XP Embedded is functionally equivalent to XP Professional any software program or driver designed for Windows XP or 2000 can be natively run on an XP Embedded system as long as the required operating system components are included. This allows for reduced development time and simple project migration for developers already familiar with Microsoft Windows products (Hall, 2005).

Windows CE .NET is designed for small devices in need of a real time operating system with limited amounts of system hardware. Windows CE differs from XP Embedded in terms hardware support and it will run on ARM, MIPS, SHx, and x86 based architectures. Windows CE supports the high level programming languages C, C++, and the .NET framework but cannot run native applications developed in Windows XP or 2000. At the very least they need to be recompiled for the CE environment because Microsoft altered the implementation of some critical operating system components (Hall, 2005).

Windows CE images are built through Microsoft's Platform builder where designers start by selecting a base hardware architecture and then add components as required for their particular application. The control over the system is not as in depth as Linux, because the kernel is not modifiable in Windows CE. A base system consisting of only a bootable kernel requires approximately 200kB and can grow to almost 18MB in size if all the possible components are added (Hall, 2005).

The Windows CE packages offer both hardware drivers and software capabilities for CE enabled devices. Platform builder comes with a wide array of hardware support to enable connectivity through many wired and wireless network protocols. In the software packages CE contains support for many standard server and application protocols, specifically web and SQL server capabilities. The http server allows a CE based device to host web pages for network management and the SQL abilities allow the device to host a local database and sync with a central device. Windows CE also provides a .NET runtime so C, C++, and C# applications targeted for a CE device may be run on the system.

3.5. PROGRAMMING LANGUAGE

The choices for the programming language to be used throughout this project were plentiful. Java, C#, Visual Basic .NET (VB.NET), C++, and C were all considered. The two most important considerations for this project are plotting and displaying data for analysis over time and having an intuitive and well-design graphical user interface (GUI) for the person monitoring the system to use. These considerations are closely followed by the familiarity of the language to the programmer(s) and a comprehensive set of language features such as database interaction and advanced XML support. The final consideration was platform portability. Java and C# both emerged as excellent choic es for the project. Both languages have powerful drawing and design features as well as a generous collection of general language features. After examining the two languages side-by-side and after factoring in a Windows platform for the server, C# became the best choice for the project

3.6. DATABASE

The database solutions considered for this project were SQL Server 2000 Enterprise Edition, Oracle 9i, and MySQL 5.0. All three databases are reliable, functional, and popular solutions for modern data storage and access needs. One of the most important considerations relating to this project is the reliability of the database and its ability to handle large amounts of data. A comprehensive set of built-in database functions and features is a less important consideration factor as the degree of complexity of the database interaction is small. Platform portability is also considered, although it is not a deciding factor (Chigrik, 2005). MySQL 5.0 emerged as the best overall choice for a database.

4. TESTING PLAN

4.1 HARDWARE TESTING PROCEDURES

The design is currently being implemented. The following testing procedures will be used to verify the correctness of the implemented system. The initial hardware testing phase consists of testing individual sensors and components. Once the individual functionality is verified the system assembly will begin with incremental testing through the entire design process. The Ethernet Bridge will be tested by connecting it to a working Windows based machine and attempting to connect to the existing wireless network.

Each of the four sensors are tested independently and interfaced with the microcontroller. The eBox hardware will be verified on a Window CE image created in Platform builder. The Ethernet port will be connected directly to a wall outlet and tested for functionality. Once the Ethernet driver is known to be working it will be tested with the wireless bridge to verify compatibility between the devices.

The microcontroller is tested for interrupt functionality and serial communication functionality. The serial communication will be tested by initially sending some ASCII data to the eBox and verifying the proper characters in the Window CE program. The framework for serial communication can then be used through the project when serial communication is required for sensor data transmission.

4.2. SOFTWARE TESTING PROCEDURES

The thorough testing of software will be an integral part of the project. Two factors that can create problems or bugs in software before any code is even written are incomplete or unclear requirements and a cramped schedule. These will be circumvented through the development of solid requirements and the use of detailed flowcharts and block diagrams. In addition to ongoing testing of each build of the software, a rigorous active software testing phase will be necessary. This phase will encompass testing all the functionality of the software. Some of the features that will require additionally scrutiny include communication between the server and the remote device, interaction between the software and the database, determination of an alarm condition and issuance of a notification, and user interaction. Testing will be comprised of several different methods. Unit testing will be accomplished by creating drivers to test the correctness of each function. An overall test of the requirements from outside the software and without testing each individual function will serve as system testing.

5. DESIGN DELIVERABLES AND RESULTS

The final deliverables for this project will be a working prototype remote sensor device and the server side C# application. The project scope will not encompass housing, power supply design, or temperature regulation for the remote device. The prototype will have a wired Ethernet connection bridged in hardware to Wi-Fi. The remote device will be able to sense flames and smoke using a UV sensor, CO sensor, and a combination of smoke detectors. The remote device will also have the capability to host files and communicate with the central server. The server side C# application will store data results from the sensor in a MySQL database and display the data in a readable form to the user. If the basic implementation is completed ahead of schedule the design team may attempt to implement network booting for the remote stations. If network booting is implemented the team will work on designing software that would enable the device to function without an internet connection. The local station would take over data analysis and provide audible warnings to residents if dangerous conditions arise.

The prototype for this project is still under development and will be completed by the time of conference. Currently the sensors are communicating with the Atmel ATMega32 utilizing the external interrupts. The onboard Atmel UART can successfully communicate with an x86 based computer running a C# program written in the Microsoft.NET 2.0 framework. The protocol for data transportation between the Atmel and eBox, the server application, and the database are still under development.

6. REFERENCES

[1]. Brian, Marshall. (2006). "How Smoke Detectors Work" [Online]. Available: <http://www.howstuffworks.com/smoke.htm>.

[2]. Chigrik, Alexander. (2003). "The comparison of SQL Server 2000 with MySQL v4.1" [Online]. Available: <http://www.databasejournal.com/features/mssql/article.php/3087841>

[3]. Chigrik, Alexander. (2005). "SQL Server 2000 vs Oracle 9i" [Online]. Available: http://www.mssqlcity.com/Articles/Compare/sql_server_vs_oracle.htm

[4]. Flame Sensor UV TRON R2868 Datasheet. (1998). Hamamatsu Corp. [Online] Available:<http://sales.hamamatsu.com/assets/pdf/parts_R/R2868.pdf>.

[5]. Hall, Mike. (2005) Comparing Windows CE and Windows XP Embedded. Embedded Technology Journal. <http://www.embeddedtechjournal.com/articles_2005/20051004_msft.htm>.

[6]. IEEE 802.11.(2005). Wikipedia <http://en.wikipedia.org/wiki/802.11>.

[7]. eBox II. 2005. Media Stream Technologies Inc. < http://www.compactpc.com.tw/ebox-II.htm>.

[8]. UV TRON Driving Circuit C3704 Datasheet. (1997). Hamamatsu Corp. [Online] Available:<http://sales.hamamatsu.com/assets/pdf/parts_C/C3704.pdf>.