TELEMETRY SYSTEM FOR REMOTE MONITORING OF UTILITY USAGE IN COMMERCIAL AND RESIDENTIAL STRUCTURES

Steven Grott, David Lecko, Ryan Parker and Nathan Price (Students)
Kurt Kosbar and Gordon Eden (Advisors)
Telemetry Learning Center
Department of Electrical and Computer Engineering
Missouri University of Science and Technology

ABSTRACT

The system described in this paper can monitor utility usage in commercial and residential structures, and send an alert message over conventional cell phone networks when it detects an anomalous condition. Such a condition could indicate a utility outage, structure failure, HVAC system failure, water leak, etc. The microcontroller-based system can measure electrical current, carbon monoxide, methane, liquid propane, temperature, barometric pressure, and altitude using a wired and wireless sensor network. The microcontroller displays the measurements on local and external graphical user interface, and sends SMS alert messages when necessary. The system may be retrofitted into existing structures.

Keywords: Autonomous Systems, Telemetry Systems, Smart-Buildings, Wireless Sensor Networks, Microcontroller Based Systems

INTRODUCTION

Approximately 6 billion devices currently connect to mobile broadband networks [1]. Cellular phones currently account for the majority of this usage. In the past decade, network speed growth, reduction in computing power cost, and battery technology has resulted in the expansion of device capabilities. There are predictions that by the year 2020, the number of connected devices in operation may reach 50 billion [1]. Many of these devices will not be cell phones. Some of these devices will perform a specific function for a user. Other devices will not have a user interface, but only provide information to another computer (known as machine to machine devices, or M2M). Over 3 billion of these devices are expected to perform utility metering telemetry functions.

Telemetry system applications have grown in the last few years due to network, computing, and battery technology advancements. Telemetry systems include the process of connecting remote sensors to a centralized data monitoring location using various communication protocols. These systems use various sensors to communicate about the surrounding environment to a central unit. Telemetry systems allow the central unit to monitor sensor data without the need for manually measuring sensor values. Users may configure a unit to store data for an extended time for automated or human analysis. This analysis may detect and alert an individual automatically, should the sensor data exceed the bounds of a specified range.

Some advanced systems analyze stored data to learn trends and determine the bounds of a specified range autonomously. Regardless of how the bounds are determined, telemetry systems may intelligently alert a user to trigger an action based on the received data. Telemetry data acquisition systems may further use alerts when interfaced with remote controls, forming a SCADA (Supervisory Control and Data Acquisition) system. SCADA systems react to out-of-range data by triggering alarms, which alert software or a control room operators of the need to address the problem to prevent a larger problem.

Telemetry, especially through SCADA, serves various industries. For example, power distribution, factory automation, security systems, and weather monitoring all use some form of telemetry. All of these applications take data, analyze trends, and trigger signals to respond appropriately. Our system, like SCADA systems, uses environmental inputs to monitor the home or a commercial structure.

PROJECT OBJECTIVES

The project met three major goals as a fully functional telemetry device:

1) A central device monitored utility (water, electricity, natural gas, etc.) use in a structure
2) Unusual detected events triggered SMS alerts to a remote user
3) A software interface provided access to data and device controls

The backbone objective for the project included the development of the central monitoring device. First, decisions were made regarding important utilities that should be monitored to prevent health and safety concerns and financial disasters. This device required sensors to monitor the utilities. Moreover, a suitable microcontroller gathered, processed, displayed, and stored the data from the sensors. Lastly, extensive testing was conducted before final installation to ensure all devices communicated as anticipated.

Detecting abnormalities in the utility system of a structure was a vital objective for the project. Each sensor was calibrated in a closed environment to determine the important thresholds. From here, the micro-controller was programmed to distinguish normal usage of the specific utilities from the unusual activity. Once a problem was detected an alert was sent to a remote user via SMS. Originally, the project plan would route a message via cellular Internet connection. However, this functionality was changed to SMS to reduce the programming, hardware, and billing complexities required. Once the device was installed and a sensor was intentionally pushed to threshold, a message was received by a remote user. As intended, the device successfully noticed abnormal utility usage and alerted the remote user.

A Graphical User Interface (GUI) was an important objective for the project because it provided situational awareness and the ability to easily change sensor sensitivity. The project has an on-device display to see real-time data and change between operating modes. In addition to the on-device display, the project has a computer interface to alter parameters. For instance, sensors can be turned on/off, sensor thresholds can be set, graphical scalars can be changed for display preferences, and real-time data or previously logged data may be viewed. The GUI offers convenient interaction with the microcontroller, sensors, and data, which was an important concept to the project design.

PROJECT SPECIFICATIONS

The primary control unit of the project was a Spectrum-Ace™ 2A [2] microcontroller board with a 14.5 cm touch screen display, which is based on a Dallas Semiconductor 89C450 microcontroller. Multiple analog sensors maybe wired to the main unit through standard headphone jack connectors on the outside of the unit. A board based on the open source Arduino™ [2] language provides an interface to a 2.4 GHz transceiver. The communication between the transceiver and microcontroller is over a 7-wire parallel interface. The main unit was powered by a standard AC to DC power adapter, which connected to the unit through a standard sized 2.1mm center positive connector. Also on the exterior of the unit was a standard 9 pin serial connector for the purpose of connecting to a personal computer.

With the featured software the user could use the serial port to change settings and usage thresholds in the device. Contained within the main unit's housing was an uninterruptable power supply (UPS) that allowed the project to continue operating in the event of a power outage. The user would be informed by text message once the unit switched to UPS power. One of the original goals of the device was to inform the user if a sensor goes out-of-bounds of normal usage. The UPS allowed for the project to continue functioning for a period of time even when the power was out. An additional wireless sensor unit could be added to the unit to expand its capabilities. The housing was an extra-large electronics hobby box as shown in Figure 1.

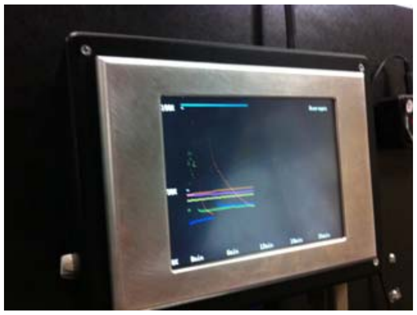


Figure 1: Primary Control Unit

The wireless sensor unit featured humidity, pressure, altitude, and temperature sensors. Two sensor boards allowed for these sensing functionalities over digital I^2C protocol [3]. These sensors were managed by an Arduino microcontroller that wirelessly transmitted the data using a

2.4GHz transmitter. The transceiver interfaced with the Arduino over an SPI interface [4]. A standard AC to DC power adapter powered the entire unit. The housing was a standard electronics hobby box as shown in Figure 2.

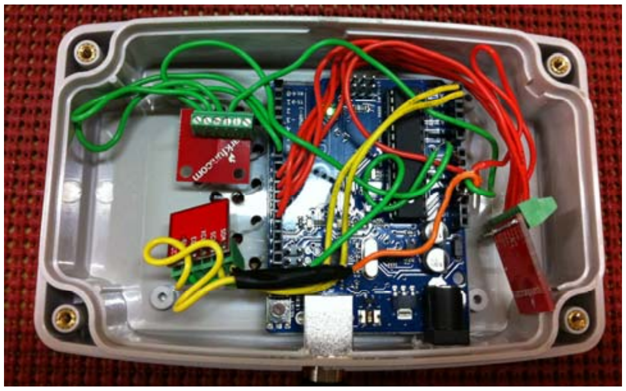


Figure 2: Wireless Sensor Unit

The current censor was a clamp over wire unit. Sensing functionality came from a Hall effect transistor capable of reading an AC or DC maximum current of approximately 20A. A small form factor allowed permanent or temporary installation nearly anywhere. The sensor required a standard AC to DC power adapter, which connected to the sensor unit using a standard sized 2.1mm center positive connector. The sensor connected to the main unit over a wire using a 1.8mm headphone jack connection. The housing was a standard electronics hobby box as shown in Figure 3.

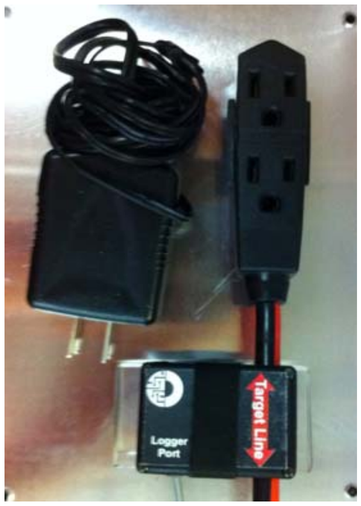


Figure 3: Current Sensor

The gas sensor detected liquid propane gas (LPG) and was housed in the main unit. The sensor received DC power from the main unit's power supply. The gas sensor communicated with the main unit over a wire directly connected to the Spectrum 2a's analog to digital converter.

Similarly to the LPG sensor, the methane sensor was housed in a standard electronics hobby box. The sensor received power from an external AC to DC power adaptor. The gas sensor communicated with the main unit over a wire using a 1.8mm headphone jack connection.

DETAILED DESIGN

Wired Sensors

Originally, sensors were expected to detect the following phenomena: electrical current, water flow, gas flow, and temperature. These were to be digital, using SPI communication.
Our final array of wired sensors included: electrical current, LPG, and methane gas. All wired sensors were chosen as analog. This took advantage of the main unit's DS2450 analog to digital converter, capable of 2.45 volt to 5.12 volt input with 1-16 bit output. The methane sensor was constructed with the main unit's battery backup to prove sensors could still function during power outage situations. All sensors were placed in a protective housing and affixed with a user-friendly cord and plug system.

Wireless Sensors

A wireless sensor node was planned to communicate with the Spectrum Ace from up to ten meters away. A third-party microcontroller was to interpret the data, and send over an unlicensed wireless channel.

Barometric pressure and temperature sensors were configured on an Arduino UNO [5]. The Bosch BMP085 barometric pressure sensor [6] was found to measure 300 to 1100 hPa with an accuracy of 0.03 hPa over an I^2C interface. The Sensirion™ SHT15 temperature and humidity sensor [7] communicated over a digital 2 wire interface. The temperature was read with +/- 0.3 degrees Celsius accuracy, and +/- 2% relative humidity. A Nordic Semiconductor™ nRF24L01 2.4 GHz transceiver [8] was selected to communicate between the base controller third-party microcontroller. The wireless transceiver communicated with the Arduino over SPI. Data was configured to transmit wirelessly in bursts of 4 bytes. Difficulty controlling the receiving-end transceiver with the main unit occurred. As a remedy, a second Arduino was set to receive the wireless signal and relay data to the base station serially. This serial connection occurred over seven individual wireless, and read all at once over evenly-space intervals. For future improvements, a driver may be written to directly control the Nordic transceiver directly with the base station. This would eliminate the need for a second Arduino, decrease latency, and decrease cost.

Mobile Connectivity and Data Storage

Originally our plan connected the project to the cellular network via GPRS data and SMS. The device uploaded data to servers via the Internet, and alerted users via text message.

The latest design connected only to the SMS functionality of GSM. Data was stored on a USB flash drive and displayed on a touch screen connected to the base station. This allowed for less use of the data network, therefore reducing cost. Also, a local user may have viewed data and control the device without a separate computer.

Graphic User Interface on PC

The PC interface stored and displayed data from the project on a windows computer as well as way to adjust settings. Microsoft's C++ .NET platform was originally chosen for development of the GUI, as it provided a framework of purpose-built, prewritten class libraries that would act like a toolbox for working within Windows. Later, it was discovered that not all .NET libraries were available to the C++ programming language. To avoid creating new graphics libraries for Windows, Visual Basic was used to complete the user interface.

The PC database brought data from the main unit and processed the data as in Figure 4. Once the data from the main unit transmitted to the computer over a serial interface, the GUI program looked for user input, the serial input. If serial input was received, the data would be stored in a Microsoft Access database. Lastly, the GUI charted the data and the program loop started over.

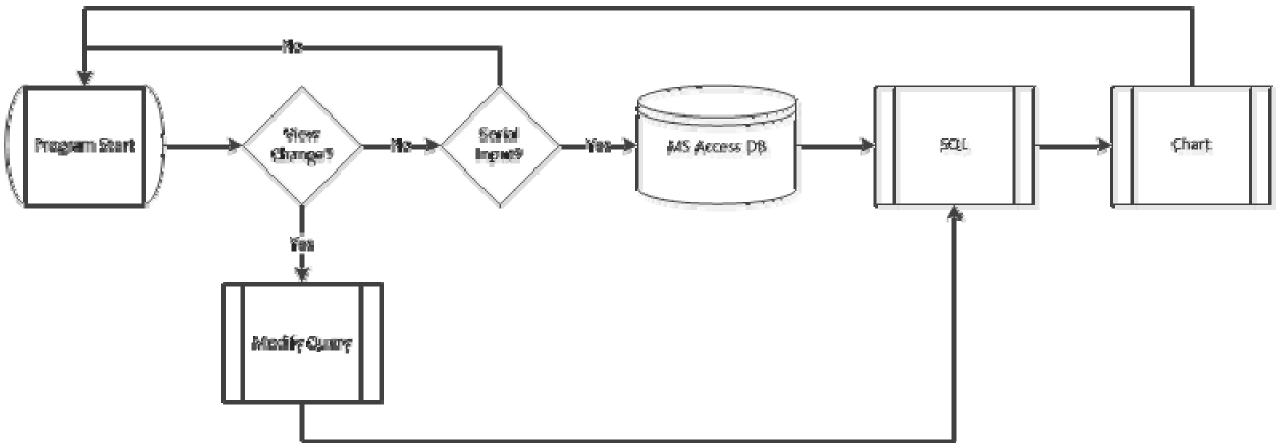


Figure 4: GUI Data Flow

RESULTS

Wired Sensors

The wired sensors included the LPG, Methane, and current sensors. The LPG and methane gas sensors were each designed to produce a voltage in the range of 0-5 volts, directly proportional to the concentration of each respective gas in the atmosphere. These two sensors were tested individually. Each gas was released near the sensor while a multi-meter read 0-5 volts accordingly.

The current sensor was attached to a power-drawing extension cord. A multimeter was connected to the sensor output, which read 0-200 mV based on the current flowing through the extension cord. An appliance known to draw 20 amps was connected to the extension cord. The voltage at the multimeter showed 200mv and 0mV while the 20amp appliance was cycled on and off, respectively. The reading at the meter was proportional to the current flowing through the extension cord. Independently, each of the wired sensors were tested and verified to work.

Wireless Sensors

The wireless sensors were connected to the remote Arduino board. These sensors measured humidity, temperature, altitude, and barometric pressure. The Arduino was configured to read data from each sensor and display the value on a connected computer screen. Atmospheric humidity and temperature were varied and the appropriate readings were observed on the display. In addition, the sensor was placed in a high-pressure environment and the associated reading was realized. Lastly, the altitude was compared to elevation of the test site. Although the altitude sensor did not vary in value, the static reading was confirmed accurate for the testing location.

Next, the remote Arduino sent sensor data over the 2.4Ghz wireless transceiver. The Arduino at the primary control unit was connected to a computer and configured to display received data on an attached computer. The Arduino boards were moved apart more than 30 feet, and the values at the remote sensor continued to match those values displayed at the main unit board.

The receiving Arduino was configured to calibrate each value in a unique, established fashion. This calibrated value was displayed on the computer screen, as shown in red boxes below. The post-calibration values were then converted to 7-bit binary, and sent using 7 parallel data wires to the main unit Spectrum 2a. The Spectrum was connected via serial interface to a computer, and the calibrated values were successfully received.

Main Unit Data Processing

Next, the main unit's algorithm for reading sensor data was tested. Every three seconds, the unit would read data from all sensors. In the display terminal below, each horizontal line of data represents a full reading. The four columns boxed in red in Figure 5 show the wireless sensor data. Wired sensors show data in columns 2, 3, and 4. The data values shown in the main-unit test were verified with the readings taken directly at the analog sensors, and at the remote Arduino for wireless sensors.

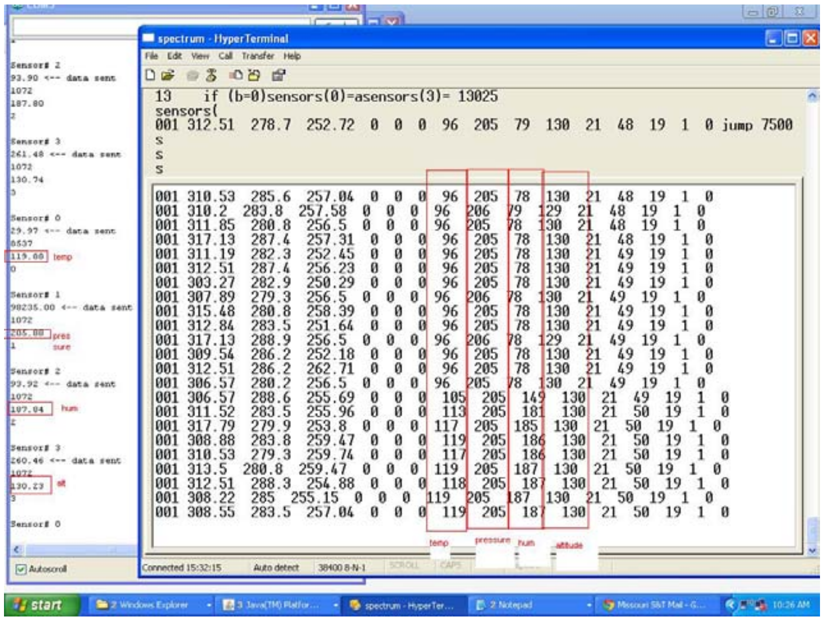
GSM Functionality

First, the GSM modem was connected directly to a computer with a serial interface. AT commands were used to send a test SMS to a mobile device. When the mobile device received the SMS, the test was successful.

Secondly, the GSM modem was connected to the main unit. The main unit was programmed to send an SMS alert when any of these thresholds were reached. Sensors were exposed to threshold breaching conditions to generate an SMS alert. Also, the power was disconnected at each sensor, which also triggers a specific SMS alert.

Touch Screen GUI

The GUI on the main unit was programmed to show each sensor value vs. time on a graph. Each sensor was represented by a different color. The graph values were calibrated to fit the window proportionally to each sensor's input range. As the main unit read data from each sensor every 3 seconds, the data was plotted on the touch screen. These graphic values were compared and verified with values displayed on the computer terminal interface.



```
Sensor# 2
93.90 <-- data sent
1072
187.80
2

Sensor# 3
261.48 <-- data sent
1072
130.74
3

Sensor# 0
29.97 <-- data sent
0537
119.00
0

Sensor# 1
98235.00 <-- data sent
1072
205.00
1

Sensor# 2
93.92 <-- data sent
1072
187.84
2

Sensor# 3
260.46 <-- data sent
1072
130.23
3

Sensor# 0
```

```
13     if (b=0)sensors(0)=asensors(3)= 13025
sensors(
001 312.51  278.7  252.72  0  0  0  96  205  79  130  21  48  19  1  0 jump 7500
s
s
s
```

```
001 310.53  285.6  257.04  0  0  0  96  205  78  130  21  48  19  1  0
001 310.2  283.8  257.58  0  0  0  96  206  79  129  21  48  19  1  0
001 311.85  280.8  256.5  0  0  0  96  205  78  130  21  48  19  1  0
001 317.13  287.4  257.31  0  0  0  96  205  78  130  21  48  19  1  0
001 311.19  282.3  252.45  0  0  0  96  205  78  130  21  49  19  1  0
001 312.51  287.4  256.23  0  0  0  96  205  78  130  21  49  19  1  0
001 303.27  282.9  250.29  0  0  0  96  205  78  130  21  49  19  1  0
001 307.89  279.3  256.5  0  0  0  96  206  78  130  21  49  19  1  0
001 315.48  280.8  258.39  0  0  0  96  205  78  130  21  49  19  1  0
001 312.84  283.5  251.64  0  0  0  96  205  78  130  21  49  19  1  0
001 317.13  288.9  256.5  0  0  0  96  206  78  129  21  49  19  1  0
001 309.54  286.2  252.18  0  0  0  96  205  78  130  21  49  19  1  0
001 312.51  286.2  262.71  0  0  0  96  205  78  130  21  49  19  1  0
001 306.57  280.2  256.5  0  0  0  96  205  78  130  21  49  19  1  0
001 306.57  288.6  255.69  0  0  0  105  205  149  130  21  49  19  1  0
001 311.52  283.5  255.96  0  0  0  113  205  181  130  21  50  19  1  0
001 317.79  279.9  253.8  0  0  0  117  205  185  130  21  50  19  1  0
001 308.88  283.8  259.47  0  0  0  119  205  186  130  21  50  19  1  0
001 310.53  279.3  259.74  0  0  0  117  205  186  130  21  50  19  1  0
001 313.5  280.8  259.47  0  0  0  119  205  187  130  21  50  19  1  0
001 312.51  288.3  254.88  0  0  0  118  205  187  130  21  50  19  1  0
001 308.22  285  255.15  0  0  0  119  205  187  130  21  50  19  1  0
001 308.55  283.5  257.04  0  0  0  119  205  187  130  21  50  19  1  0
```

Figure 5: Data received from Arduino (left window) and Spectrum (right window)

Also, a user-touch on the screen was programmed to change the graph display to an all-text display. The text displayed average values for each sensor over the last 30 minutes. This calculation was performed manually with data from the values displayed on the computer terminal interface. Once a user touched the screen again, the text display toggled back to the graph display.

PC GUI

The PC graphic user interface was programmed to read data in real time from the main unit. This data was transmitted from the Spectrum 2a's serial port to the PC at the same time as displayed on the touch screen GUI. Data was verified on the computer GUI based on the touch screen GUI display.

Battery Backup

The battery backup was connected to take over if the AC power became unavailable. Operating the device and abruptly disconnecting the power tested backup capability. Then re-connecting the AC power while the device ran in backup made verified the full function of the battery backup. Lastly, the device was operated on backup power until the battery completely drained. The device operated uninterruptedly during all of these tests, until the batteries were completely drained during an endurance test.

CONCLUSION

As the use of telemetry and monitoring systems grows, many approaches to system design will result. This project demonstrated the marriage of four sensor communication protocols, seven sensors, two graphic user interfaces, and an analysis and alert system. The hardware and varying technologies used for this system may be configured for many applications, which will be crucial in the exploration of connected device roles of the future. Rapidly growing demand will lead increased innovation and use of any number of protocols, of which our system has demonstrated the successful use of many.

REFERENCES

[1] Ericsson. "More that 50 billion connected devices." *Ericcson.com.* February 2011. April 25, 2012. http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf

[2] Arduino Open Source Platform, [Online] http://www.arduino.cc

[3] UM10204 I^{2}C-Bus Specification and User Manual, Rev. 4, 13 February 2010, NXP B.V., [Online] http://www.nxp.com/documents/user_manual/UM10204.pdf

[4] SPI Block Guide V03.06, Motorola Inc., [Online] http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[5] Arduino Uno Board, [Online] http://arduino.cc/en/Main/ArduinoBoardUno

[6] Bosch BMP085 Digital Barometric Pressure Sensor, Version 0.2 022008, Bosch Sensortec GmbH, [Online] http://www.bosch-sensortec.com/content/language1/downloads/BMP085_Flyer_Rev.0.2_March2008.pdf

[7] Datasheet SHT1x (SHT10, SHT11, SHT15), Sensirion AG, Ver 5.0, Dec. 2011, [Online] http://www.sensirion.com.cn/down/downimg/Datasheet-SHT1x%20V5.pdf

[8] Single Chip 2.4 GHz Transceiver, Nordic Semiconductor, [Online] http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01#Downloads