

A Prototype Home-Based Environmental Monitoring System

Michael Cook¹, Trina Myers^{1,2} and Jarrod Trevathan³

¹*Discipline of IT, School of Business, James Cook University, Queensland, Australia*

²*Centre for Tropical Biodiversity and Climate Change, James Cook University,
Queensland, Australia*

³*School of Information and Communication Technology, Griffith University, Australia*

^{1,2}*(Michael.Cook; Trina.Myers)@jcu.edu.au*

³*j.trevathan@griffith.edu.au*

Abstract

An important step towards reducing humankind's reliance on unsustainable energy is by improving energy efficiency in a home or building. One solution to energy-intensive homes is to take environmental measurements in-house that can alert the owner of what rooms and conditions have high-energy waste (e.g., unnecessary air-conditioning or lighting). This paper describes the development and performance of a low-cost, easy-to-use and easy-to-maintain wireless sensor network suited for home environmental monitoring. Low-cost and ease-of-use are achieved through a single open-source software element developed for use on a home computer and a low-cost sensor node element that can be simply placed and turned on. Ease-of-maintenance is achieved by prolonging the sensor's battery life and minimising user interaction. Data is retrieved from the sensor and aggregated to central location automatically when the home PC/sink node is turned on. There is no further interaction is required by the user beyond initialising the application.

Keywords: *Wireless sensor networks, middleware integration platform, environmental monitoring, energy efficiency models*

1. Introduction

The ability to capture environmental data such as temperature and humidity in homes is an important step towards understanding how to build more environmentally friendly housing in the future. This ability is also imperative to educating the populace on energy efficiency and how to improve their energy usage at home. *Wireless Sensor Networks* (WSN) are emerging as a useful and viable technology that can enable the collection of environmental measurements around the home.

While there has been significant research into WSNs over the last decade, most efforts have been aimed at large or physically challenging applications (e.g., [1-3]). Small-scale applications such as environmental monitoring in the home appear to have been largely overlooked except for preliminary work by Lozano [4] on indoor air quality. Most home-based systems have focussed on outpatient health monitoring and telemedicine applications [4-7]. However, the data collected from a home-based environmental WSN can be used constructively to view energy consumption trends, compare usage with other homes, and provide motivation towards saving energy.

A home-based environmental WSN requires different considerations to large-scale applications, such as being low-cost, easy-to-use, and requiring minimal maintenance. The cost of developing and implementing the system must be low to make home monitoring viable for the energy savings the system is expected to produce. The system should also be simple to manage by a non-specialist (*i.e.*, the homeowner) and unobtrusive or largely unnoticeable as part of the home infrastructure. Beyond the initial installation, the system should be self-sustaining and require minimal maintenance on the user's behalf.

This paper focuses on the technical aspects of implementing a home-based monitoring system from end-to-end for the purposes of gathering environmental measurements. The proposed system aims at a simple setup process for low-cost home monitoring that requires minimal input and maintenance from an average computer user. A *Personal Computer* (PC) is used to collect data from a sensor node via a WSN. The data is stored locally on the sensor node and then automatically sent to a central repository when an active Internet connection is available. This project serves as a proof-of-concept for a home-based environmental WSN.

This paper is organised as follows. Section 2 presents the problem motivation and the project aims. Section 3 discusses the proposed methodology for implementing the system. Section 4 specifies the implementation details for the home-based environmental WSN. Section 5 describes a series of performance tests conducted to assess the viability of the system. Section 6 provides some concluding remarks and avenues for future work.

2. Background and Project Aims

A WSN consists of a collection of sensor nodes that are physically separated [8]. Each of these nodes can be configured with a variety of physical sensors capable of measuring a number of different variables, which might include temperature, light, humidity, wind speed, salinity, water pressure, *etc.* Data from these nodes can then be collected from a central point in the network (also known as a sink node). The wireless property allows for sensor nodes to be deployed across large physical areas without the need for additional infrastructure such as power and data lines. This enables systems to be deployed in remote or challenging environments. Some examples of the variety and range of WSN applications include a deployment in a South East Queensland rainforest [2], an Ecuadorian volcano [3], and more recently, the SEMAT project for marine applications (Trevathan *et al.*, 2012).

The *Smart Environmental Monitoring and Analysis Technologies* (SEMAT) project aims to develop intelligent, low-cost WSNs for use in marine and coastal environments. SEMAT is able to achieve its low-cost approach by allowing the WSN designers to select hardware/sensors from multiple vendors thereby creating a heterogeneous system that best suits the application needs and budget. This heterogeneity property is achieved through the use of a special middleware integration platform.

Middleware generally refers to software or tools used to interface with hardware or software. The main goal of middleware is to hide the complexity and heterogeneity of the underlying hardware and network platforms, provide easier management of resources, and increase the predictability of executions of the application [9]. While many types of WSN middleware exist (*e.g.*, Mires [10] and Impala [11]), SEMAT utilised a middleware integration platform referred to as the *Sensor Abstraction Layer* (SAL).

SAL is an architecture designed to provide a consistent and uniform view of sensor networks regardless of the technologies involved [12]. SAL was designed to run on a sensor gateway and aims to aggregate data from multiple sensing technologies. The driving reason behind the development of SAL is the disparity between accessing and probing the many different hardware devices that are capable of sensing data. There are many different manufacturers of sensor hardware, and there can be great differences between both the

protocols required to access the sensors, and the format of sensed data that is returned. SAL abstracts this process, providing a single, stable and hardware-independent interface to manage an entire sensor network, regardless of the sensing technologies used [13].

While SAL provides an interface to the hardware, it is not a standalone entity. Instead, it is a library of high-level functions for managing and addressing sensors [12]. The full integration into a complete WSN middleware solution is a typical use for SAL, presenting sensor data in a uniform manner to the network. Currently, SAL implementations have been limited to lightweight installations of Linux-based operating systems. Gigan [12] presented a prototype implementation of SAL deployed on the Great Barrier Reef in 2007 consisting of a single board computer and four different sensor technologies. SAL was also successfully used by the SEMAT project with large-scale deployments occurring at Deception Bay and Heron Island Australia in 2011 and 2012 respectively [1].

2.1. The Home Environmental Monitoring System Aims

The proposed home-based environmental WSN aims to capitalise on the low-cost approach taken by SEMAT to:

- Create an in-home environmental sensing platform based on inexpensive hardware;
- Identify the best method to aggregate and store data from up to 2,000 individual WSNs; and
- Develop an easy to use software application for Windows-based personal computers to be deployed into a home environment for collection and transmission of sensed data.

The following four steps have been identified in pursuit of these aims:

1. Port the Java-based SAL from the Linux platform to Windows. SAL will be a key aspect of this project due to its ability to transform various sensed data into a standardised format before transmission to a central repository. SAL will be instrumental in the ability to connect different types of sensing hardware to the system when required.
2. Develop a plug-in for SAL so data can be retrieved via an USB XBee Explorer board [14], which is used to interface with the WSN. This plug-in will allow for the intermittent transferral of data from the WSN to storage on the user's computer.
3. Create a sensor node based on the Arduino hardware platform [15]. The platform will utilise 1-Wire temperature sensors and store data on a local SD card.
4. Develop a centralised database repository for the storage of data collected from the WSN.

The three main contributions arising from this paper include:

1. The further development of the software for the WSN nodes, including the USB interface to a Windows-based personal computer;
2. The porting and development of SAL onto the Windows platform to interface with the WSN; and
3. The storage and transmission of stored data to a centralised database.

These three components should form a package that is simple to setup and configure in a home environment. That is, minimal user input will be required to read, store and send sensed data to the database when an Internet connection is available.

3. Methodology

The objective of this paper was to create a prototype WSN solution for home environmental monitoring to help improve energy efficiency in the home (Figure 1). Ideally, the end-user's PC should act as a sink node with a wireless USB device that when turned on,

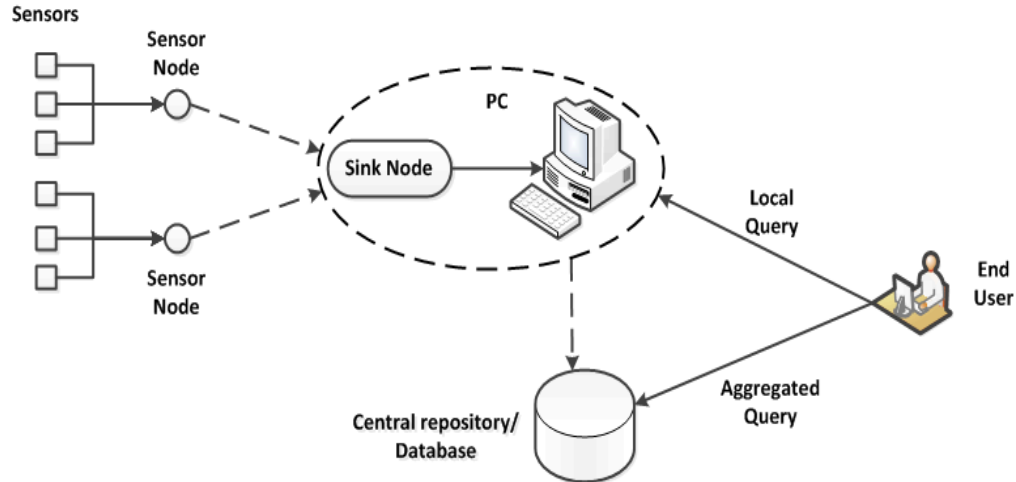


Figure 1. Proposed WSN System, from Sensors to User Query

can connect to the sensor nodes. At this point the PC/sink node aggregates and transmits the data to a central database.

3.1. Features

Three major components were developed within the scope of this project: the *sensor node platform*; the *sink node*; and the *central database*.

3.1.1. Sensor Node:

The sensor node was designed to take environmental measurements, store this data locally, and transmit the data wirelessly when the sink node sends a request. The system takes samples at one-minute intervals. The sensor node design requirements were broken down into the following four features:

1. Ability to sense temperature;
2. Local data storage;
3. Wireless data transmission capability; and
4. Optimised battery life.

While there are many other environmental measurements that could be taken (such as humidity, light and noise levels), this study was restricted to temperature for the purpose of simplicity.

A sensor node must have the ability to locally store sensor readings because an Internet connection will not always be available. The capacity of the sensor node must be sufficient to retain the data until the sink node sends a request to transmit the sensor readings. This functionality follows a query-based information extraction approach as presented by Alsoubi [16].

The proposed system should be able to utilise the existing wireless network for data transmission rather than establishing a separate network on the assumption that most homes now have wireless Internet connections.

Finally, optimised battery usage needed to be considered. This will influence how much interaction and maintenance is required on the user's behalf.

3.1.2. Sink Node:

Data is gathered by the sink node from the sensor nodes so it can be used either locally or transferred to a remote database. The design requirements for the sink node are as follows:

1. Wireless capability for data retrieval;
2. Sufficient storage capacity for data; and
3. Internet connectivity

A desktop PC running Windows was used as the sink node. A desktop PC fulfils the requirements of storage capacity for data. Additionally, most desktop PCs are assumed to have some type of Internet connection to provide connectivity to a central database.

3.1.3. Central Repository:

The central repository aggregates data from multiple sink nodes and contains sensor calibration data. The sink nodes transmit data to the repository in bulk batches. This allows for the database to receive bursts of data at irregular intervals.

4. Implementation

This section presents the details of the hardware and software requirements and issues relating to the final implementation of the proposed system.

4.1. Hardware

The hardware chosen was an Arduino-based platform. Arduino is an open-source electronics prototyping platform utilising easy-to-use hardware and software [15]. These systems are highly configurable, with a range of different manufacturers producing boards with different hardware capabilities. Table 1 lists the hardware that was chosen and the costs for each component.

Table 1. Hardware and Costs

Hardware product	Cost AUD
Seeeduino Stalker v2.3 board [14]	\$39.00
UartSBee board [14]	\$19.50
Synapse RF266PC1 wireless module [17]	\$40.50
Generic SD card – 4GB	\$7.00
DS18B20 digital temperature sensor [18]	\$3.00
XBee Explorer USB [19]	\$25.00

The Seeeduino Stalker v2.3 board serves as the basis of the sensor node platform. This board is compatible with the Arduino programming language. The Stalker board provides many of the capabilities necessary to prototype the WSN, including:

- Onboard real-time clock chip;
- MicroSD card socket; and
- Bee series socket (necessary to add wireless module).

The UartSBee board is used to interface directly with the Seeeduino Stalker v2.3 board via USB. The device acts as a virtual serial port for communication directly between the PC and the Stalker. This interface is used for programming the Stalker, as well as providing power and test capabilities before adding wireless capability to the system.

The Synapse RF266PC1 is a Bee compatible wireless module used to add wireless communication between sink node and sensor node. The module is compatible with the Bee series socket available on the Stalker board and the XBee Explorer USB. This module is fully programmable, which differentiates it from other similar modules.

The DS18B20 is a digital temperature sensor utilising a 1-Wire bus. These sensors can be calibrated to improve their accuracy from the default $\pm 0.5^{\circ}\text{C}$ to $\pm 0.1^{\circ}\text{C}$.

The XBee Explorer USB board is used to connect a Synapse RF266PC1 wireless module to a PC. This device can be used both for programming the wireless modules and as a serial interface to the RF266PC1 for wireless communication to the sensor node.

4.2. Software

Three different software packages were required. The sensor node, sink node and central repository all have different requirements in terms of languages and tools. These software packages are listed below, grouped under their related components:

1. Sensor node:
 - a. Arduino v1.0.1 [15]
 - b. Synapse Portal v2.4.34 [17]
2. Sink node:
 - a. Java SE Development Kit 7u7 (JDK)
 - b. IntelliJ IDEA Community Edition v11.1.2
 - c. Maven
 - d. Synapse Portal v2.4.34
 - e. RXTX [20]
3. Central repository:
 - a. Linux-based server (Linux Mint)
 - b. MySQL

Arduino v1.0.1 is an open source *Integrated Development Environment* (IDE) used for programming Arduino compatible devices. The Arduino language itself is essentially C/C++, providing a familiar programming structure. The IDE is to be used to program the Seeeduino Stalker board. The software works by communicating via a serial port, necessitating the use of the UartSBee board.

Synapse Portal v2.4.34 is a free development environment designed to manage Synapse wireless modules. This tool can be used to program modules, manage networks, and update firmware.

Java is the language required for developing the sink node component. No flexibility in choice was available here, as Java is already used by SAL and is necessary to port the system to the Windows platform. IntelliJ IDEA Community Edition was selected as the IDE for

developing the sink node. IntelliJ is an open source IDEA compatible with the Java language. This was used in conjunction with Maven, a software project management tool.

RXTX is a Java library that provides serial communication for the JDK [20]. This library provides the functions necessary to enable serial communication between the sink node and sensor node components.

The central repository was developed using the MySQL database system on a Linux server. Both MySQL and Linux are open source and provided the necessary features required for the database component. For example, the server must have the capacity to accept remote connections from clients over the Internet and MySQL provides this ability.

4.3. Project Creation

This section details the system's design decisions, implementation, the issues encountered and how they were overcome.

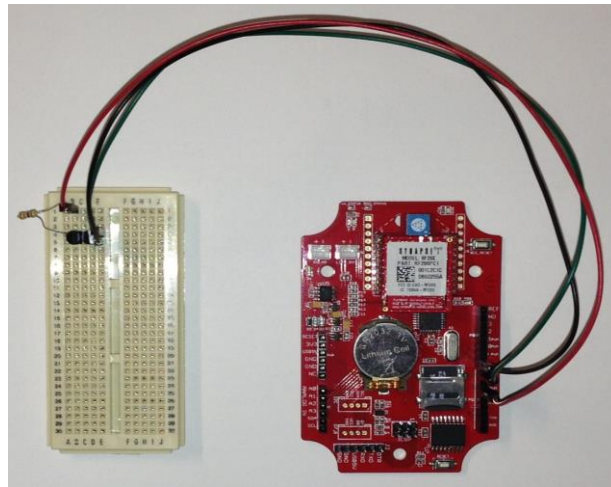


Figure 2. Sensor Node Prototype

4.3.1. Sensor Node Hardware Setup

Decisions on the hardware design needed to ensure all of the physical requirements of the system were covered, including:

1. Wireless capability;
2. Real-Time Clock for synchronisation of data;
3. Temperature sensing;
4. Internal storage; and
5. Independent from external power.

A prototype of the sensor node was created using the aforementioned hardware (Figure 2). The prototype includes on-board storage through the use of the MicroSD memory card, wireless capability with the Synapse RF266PC1 module and temperature sensing from two DS18B20 digital temperature sensors. Independence from external power was achieved by using a 3.7V lithium-ion rechargeable battery.

The Seeeduino Stalker board comes standard with an on-board *real time clock* (RTC). However, to enable the RTC to send interrupts to the system, a jumper had to be manually soldered on the underside of the board as outlined in the manufacturer's online documentation [14].

4.3.2. Sensor Node Software Design

The software design of the system was completed in the Arduino IDE using C/C++ libraries. The libraries available within the Arduino community provided much of the functionality required by the system, while adhering to the open source nature of the project.

Data storage was completed through the use of a MicroSD card. A consideration into local storage was to minimise the amount of data necessary. While further optimisations could be made to the storage format of the data, the chosen format was deemed sufficient for the scope of the project. The format stores date and time, with multiple sensor identifiers and temperature readings in a single file. Values were separated using delimiter characters with each new set of readings on a new line. Data was stored using the following structure for each set of values:

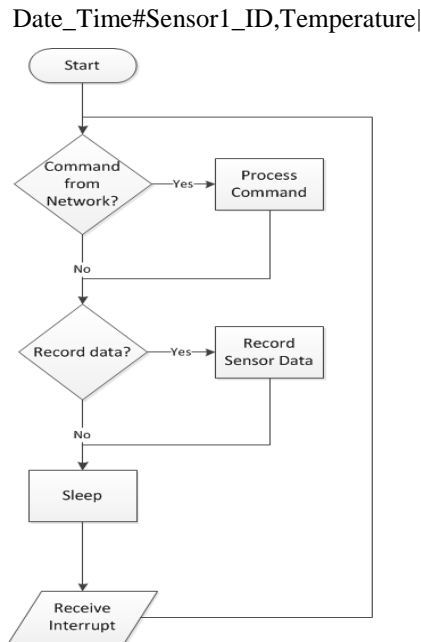


Figure 3. Sensor Node Algorithm

Sensor2_ID,Temperature

The raw data format provided from the DS18B20 sensor is used to record the temperature data. This is a 12-bit integer, and can be converted into degrees Celsius by dividing the number by sixteen. However, because calibration data for each sensor must be applied to the raw reading, no calculations are done at this level. This way, the amount of processing completed by the sensor node is minimised by recording only the raw data. Conversions can be applied at the database and/or sink node level.

Battery life was extended by using a low-power sleep mode on the Seeeduino Stalker board. Several different levels of power saving are available through the sleep function. The level that puts most hardware functions on the board to sleep but leaves the wireless module powered to receive network requests while sleeping was applied.

The sensor node algorithm can be represented as a simple loop containing decisions and awaiting input from system interrupts. Figure 3 illustrates the algorithm and is further detailed in the five steps below:

1. *System Start* – This process is only run when the device initially powers on. The start-up code contains the device initialisation settings and sensor configuration.
2. *Check for Network Commands* – This is the first step within the program loop. At this point, the system checks the internal buffer for any commands received from the sink node via the wireless interface. If no commands have been received, the algorithm continues to step 3, otherwise the system must process the request.

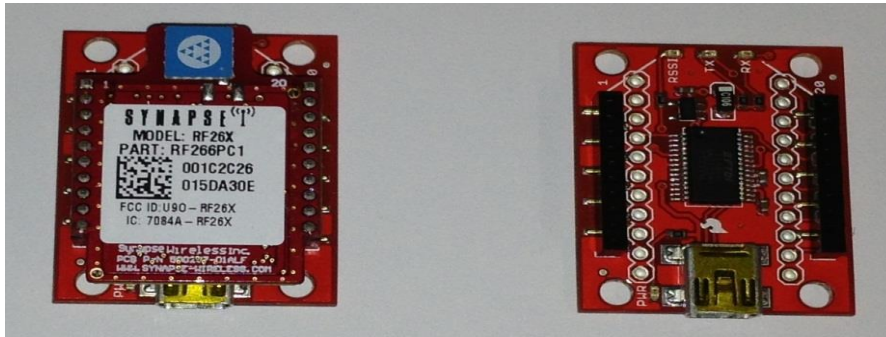


Figure 4. Xbee Explorer USB

Generally, requests from the sink node relate to retrieving all of the currently logged data stored on the SD card.

3. *Record Data* – The system checks specific intervals of time to determine when to record sensor data. If sensor data must be recorded, the system queries the DS18B20 temperature sensors for data, and writes this to the SD card.
4. *Sleep* – The system puts itself into a low-power sleep mode while awaiting a system interrupt. This step helps to preserve system power.
5. *Receive Interrupt* – Interrupts are used as a signal for the system to return to a processing state and perform one or more actions starting from Step 2. These interrupts come in two forms:
 - a. *Network Interrupt* – This interrupt signals that a request for data has been received over the wireless interface from the sink node.
 - b. *RTC Interrupt* – This interrupt originates from the on-board, real-time clock and signals for the system to record sensor data.

4.3.3. Sink Node

The sink node was implemented as software package running on a desktop PC, with a wireless device attached. Wireless capability was implemented via a plug-in device attached to a USB port on the PC. The USB device attached was the Xbee Explorer USB with a Synapse RF266PC1 wireless module. The Xbee Explorer provides access to the wireless system by configuring itself as a virtual serial port in the Windows operating environment. All communications from the sink node to the sensor node are routed through this serial port. Figure 4 shows an Xbee Explorer USB on the right, and with the Synapse RF266PC1 wireless module on the left.

Wireless communication was achieved through configuring the modules as a point-to-point network. The Synapse RF266PC1 wireless module has the capacity for custom wireless configurations to be setup and loaded to each module. This allowed for each device to be configured for communication with the other module in the pair via the last three bytes of their respective hardware MAC addresses, creating a point-to-point network. A sample of the code required to complete this configuration is as follows:

```
from synapse.evalBase import *otherNodeAddr = "\x5D\x25\x5A"  
def startupEvent():  
    initUart(1, 38400)  
    flowControl(1, False)  
    crossConnect(DS_UART1, DS_TRANSPARENT)  
    ucastSerial(otherNodeAddr)  
  
snappyGen.setHook(SnapConstants.HOOK_STARTUP, startupEvent)
```

Communication via the serial connection was limited to a maximum baud rate of 38,400. While both the PC and the sensor node are capable of running at higher baud rates (up to 115,200), transmissions were subject to corruption or one-way communication at any speed above 38,400 baud.

The sink node software was implemented using the middleware solution of SAL [12]. SAL was rewritten from its original Linux platform to be used in the Windows environment. This was accomplished by compiling SAL in Java using the RXTX library for Windows. An application was written to utilise the modified SAL libraries and implement a plug-in to communicate with the sensor node in a Windows environment.

The last feature of the sink node to be implemented was the ability to transmit data to the central repository. The system is able to directly insert data into the database because the sink node creates a direct connection to the remote MySQL database through the use of Java libraries. The structure of the query was designed to minimise the required time for transmission.

As SAL is middleware software, a main program loop was written to enable the system to run. This was programmed in Java using the program structure shown in Figure 5. This algorithm is as follows:

1. *Start* – The program starts and configures the SAL middleware components as necessary.
2. *Detect Serial Port* – The system detects the available serial ports on the PC, including the virtual serial port from the XBee Explorer USB device.
3. *Request Data Log* – The system sends a request to the sensor node via the virtual

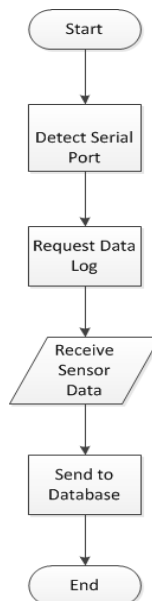


Figure 5. Sink Node Algorithm

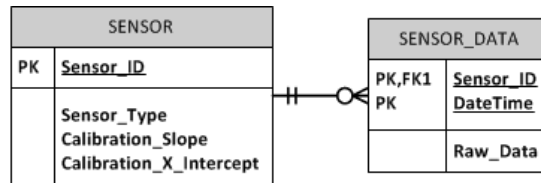


Figure. 6. Database Entity Relationship Diagram

serial port, and awaits the reply from the sensor node.

4. *Receive Sensor Data* – The SAL system writes the data to a storage file on the PC because the data streams back from the sensor node. The data stream is completed upon receipt of an end of file indicator from the sensor node, represented as the string “***EOF***”.
5. *Send to Database* – All data received from the sensor node is then transmitted to the remote database for aggregation purposes. The sink node connects to the remote MySQL server and transmits the data using direct SQL statements.
6. *End* – The program terminates.

4.3.4. Central Repository

The central repository was implemented using a PC running Linux, with a MySQL database to accept remote connections. Figure 6 shows the simple database schema. The SENSOR_DATA table stores the sensor readings and SENSOR table contains sensor calibration information. The accuracy of the sensors was improved from $\pm 0.5^{\circ}\text{C}$ to $\pm 0.1^{\circ}\text{C}$ by applying the following formula:

$$\text{Corrected Value} = \text{Raw_Data} - (\text{Raw_Data} * \text{Calibration_Slope} + \text{Calibration_X_Intercept})$$

5. Results

5.1. Testing Methodology

The following tasks were identified for testing:

1. Sensor node energy efficiency;
2. Data collection at the sensor node;
3. Data transmission to the sink node; and
4. Transmission of collected data to the database.

Testing the energy efficiency of the sensor node involved running the system until the battery completely discharged. The system was tested using a fully charged 2,200 mAh 3.7 V lithium-ion battery. The sensor node was configured to sense data at one-minute intervals and putting the device to sleep between readings. No external commands were executed on the system during this time, so wireless data transmissions did not impact on the lifespan of the device. Two full cycles were performed on the system to gather performance results.

Data collection at the sensor node was undertaken over a 24-hour period. During this time, the amount of data recorded from two temperature sensors was measured in terms of bytes produced by the system.

Data transmission from sensor node to sink node was measured as the time taken to transmit 24-hours worth of two-sensor data wirelessly. A static set of data previously produced by the sensor node was used for this test. The sensor node algorithm was modified

to perform only data transmission functions using this data. Tests were performed at varying distances indoors (2 m, 10 m, 50 m) to determine if range had any impact on transmission speeds achieved. The data was also examined to check for any data corruption caused by distance. Additionally, five tests were performed at each distance to produce an average result.

Finally, the transmission time of the data transferred from the sink node to the central repository was tested. The database used for the central repository was located in a different physical location, with all data transmissions performed over the Internet. In terms of routing, the central repository was nine hops away with a latency of 62 ms. A corresponding transmission test to the database was performed for each 24-hours of recorded data from sensor node to sink node.

5.2. Energy Efficiency

The sensor node was put through two full battery cycles (*e.g.*, from full charge to complete discharge) to determine the approximate running time. Two factors were identified that could have an impact on the results (Table 2):

1. Wireless interference causing the device to wake prematurely; and
2. Variations in the amount of energy stored in the battery for each full charge.

Table 2. Running Duration of Sensor Node

Test Number	Duration of Cycle
1	7.5 days
2	8 days

5.3. Data Collection

The data collected at the sensor node was recorded, and a single file was produced containing the data stored during a 24-hour period. This data consisted of 1,440 records, comprised of a total of 2,880 sensor readings. The file size produced was a total of 99,360 bytes (97 KB). Figure 7 shows a sample of the data produced.

5.4. Data Transmission to Sink Node

Data transmission from sensor node to sink node was tested at three different distances applicable to a standard home, with five tests run at each distance. Table 3 indicates the transmission time in milliseconds (ms) for each of the tests, followed by the average transmission time for each set of tests.

Data corruption during transmission was also tested. At distances of 2 m and 10 m, no data was lost or corrupted. However, at a distance of 50 m, three of the five tests performed resulted in corrupted data. This data corruption was evident by up to six records per test (out of 1,440 total records) missing some characters of data, or simply corrupted.

```
2012/10/09 13:46:00#28EC1A0803000073,390.00|28A233080300001E,389.00
2012/10/09 13:47:00#28EC1A0803000073,389.00|28A233080300001E,389.00
2012/10/09 13:48:00#28EC1A0803000073,390.00|28A233080300001E,389.00
2012/10/09 13:49:00#28EC1A0803000073,389.00|28A233080300001E,389.00
```

Figure 7. Sample of Sensor Node Data

Table 3. Transmission Times to Sink Node

Test Number	Distance		
	2m	10m	50m
1	26580	26610	26681
2	26595	26640	26629
3	26591	26621	26703
4	26581	26591	26747
5	26594	26593	26669
Average Time (ms)	26588.2	26611	26685.8

5.5. Data Transmission to Central Repository

Transmission times to send data from the sink node to the central repository were also tested. These times are dependent mostly on the quality of the Internet connection between the sink node and the database. As data corruption issues occurred at a distance of 50 m, only the transmission times from the first two sets of tests are included here (Figure 8).

5.6. Discussion

The initial cost for all of the sensing hardware totals less than \$150 AUD for one complete sensor node (*i.e.*, Arduino platform, two temperature sensors and related electronics). The homeowner must supply the PC and Internet connection. The cost of the system then scales with the number of sensors desired and how many rooms need to be monitored. The approximate total cost of the system for an entire house is \$500-\$1,000 AUD. Notably, other non-related uses must factor into the cost of the PC and Internet.

The energy efficiency of the sensor node proved to be sufficient for the level of prototyping produced. The ability of the system to run continuously for over one week (7.5 – 8 days) without intervention shows promise for the requirement of being easy-to-maintain. While this level of energy efficiency is sufficient for the scope of the proposed system, improving battery life will be an ongoing challenge.

The data collected during the testing with two temperature sensors resulted in around 97 KB of data per 24-hour period. This amount of data usage is feasible for the system as a whole. For example, 1 year’s worth of sensing data (* 365) equates to a size less than 35 MB. The sensor node utilised a MicroSD card with 4 GB of storage capacity as a prototype, which

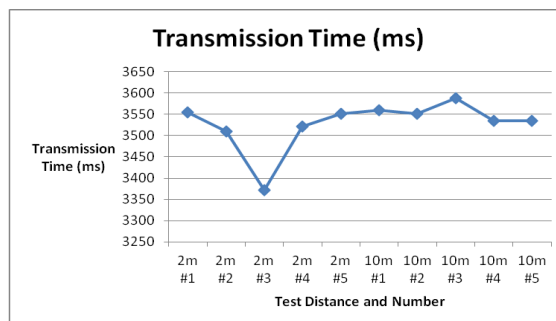


Figure 8. Transmission Times to Database

proves the sufficient feasibility for capturing sensed environmental data in a home environment.

The tests showed the transmission time from the sensor node to the sink node proved stable but transmission distance a problem. The distance of each test did not affect data transmission time; average transmission times were less than 27 seconds. However, the ability of the wireless module to transmit accurately over the various distances became an issue at the 50 m test, with over 50% of the tests resulting in some form of data loss or corruption. The robustness of the wireless communication appears to be an issue and could benefit from alternative programming methods. For example, incorporating some type of packet acknowledgment system to confirm data has been sent or received correctly.

Finally, data was transmitted from the sink node to the database and transmission times were recorded. There were only minor differences across the ten results, most of which can be attributed to variations in latency across the Internet connections. This is a factor outside the control of this project. Notably, the time required to transmit 24-hours worth of data is less than 4 seconds. This result will vary depending on the quality of the user's Internet connection.

6. Conclusion

This paper presented a prototype home environmental monitoring system which aimed to be an inexpensive, easy-to-use, and easy-to-deploy/maintain package. The prototype was designed using off-the-shelf technologies, such as Arduino micro-processors and a Microsoft Windows PC, to collect environmental measurements for the purpose of optimising household energy usage (*e.g.*, air-conditioning, *etc.*). The system serves as a WSN that retrieves data from a sensor node (containing two inexpensive temperature sensors) and aggregates the data in a central database. No interaction is required from a user beyond the initial running of the application on the PC, as the upload to the repository is automated once the PC/sink node is active.

Expenses were minimised through the use of free and open-source elements at all levels of the project with the total cost of the basic hardware for a sensor node being less than \$150. The goal of being easy-to-use was achieved by developing a single automated software element on a home user's PC to act as a sink node and a sensor node element that simply needs to be placed in a location and turned on. The software element consists of a SAL middleware integration layer that was programmatically customised specifically for a Windows environment. The system achieves the goal of minimal maintenance through prolonging battery life on the sensor node and minimised user interaction with the sink node.

Test results indicated that the system was able to retrieve significant amounts of data in a short amount of time. The data derived from this type of system, when collected and compiled from multiple homes, could provide very important information towards improving energy efficiency in the home.

Some aspects of the WSN that could be explored further in the future include:

- Improvements to the wireless data transfer speeds;
- Increasing the number of sensor nodes available in each home, providing data from multiple rooms to give a better picture of environmental conditions throughout the home;
- Improvements or alternatives to SAL;
- Improved user experience through an effective graphical user interface design, including options for the user to view their own data;
- Further development of the database and schema to include other variables and a more secure method of data submission; and

- Enhancements to the system to achieve greater levels of energy efficiency.

Acknowledgements

The authors wish to thank Professor Ian Atkinson, Shirui “Jerry” Yan, Nigel Bajema and Dr Scott Mills. This work was supported by the SEMAT project and preparation of this manuscript was partially supported by the Griffith University New Researcher Grant Scheme.

References

- [1] J. Trevathan, R. Johnstone, T. Chiffings, I. Atkinson, N. Bergmann, W. Read, S. Theiss, T. Myers and T. Stevens, "SEMAT – The next generation of inexpensive marine environmental monitoring and measurement systems", *Sensors*, vol. 12, no. 7, (2012).
- [2] T. Wark, W. Hu, P. Corke, J. Hodge, A. Keto, B. Mackey, G. Foley, P. Sikka and M. Brunig, "Springbrook: Challenges in developing a long-term, rainforest wireless sensor network," presented at the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2008), Sydney, Australia, (2008).
- [3] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees and M. Welsh, "Deploying a wireless sensor network on an active volcano," *Internet Computing, IEEE*, vol. 10, no. 2, (2006).
- [4] J. Lozano, J. I. Suárez, P. Arroyo, J. M. Ordiales and F. Alvarez, "Wireless Sensor Network for Indoor Air Quality Monitoring", *Chemical Engineering Transactions*, vol. 30, 1 (2012).
- [5] I. Korhonen, J. Parkka and M. van Gils, "Health monitoring in the home of the future," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 22, no. 3, (2003).
- [6] W. Y. Chung and S. J. Oh, "Remote monitoring system with wireless sensors module for room environment", *Sensors and Actuators B: Chemical*, vol. 113, no. 1, (2006).
- [7] T. C. Yu, C. C. Lin, C. C. Chen, W. L. Lee, R. G. Lee, C. H. Tseng and S. P. Liu, "Wireless sensor networks for indoor air quality monitoring", *Medical engineering & physics*, vol. 35, no. 2, (2011).
- [8] J. Yick, B. Mukherjee and D. Ghosal, "Wireless sensor network survey", *Computer Networks*, vol. 52, no. 12, (2008).
- [9] M. M. Wang, J. N. Cao, J. Li and S. K. Dasi, "Middleware for wireless sensor networks: A survey", *Journal of computer science and technology*, vol. 23, no. 3, (2008).
- [10] E. Souto, G. Germano, G. Vasconcelos, M. Vieira, R. Nelson, C. Ferraz and J. Kelner, "Mires: a publish/subscribe middleware for sensor networks", *Personal and Ubiquitous Computing*, vol. 10, no. 1, (2006).
- [11] T. Liu and M. Martonosi, "Impala: a middleware system for managing autonomic, parallel sensor systems," presented at the Proceedings of the ninth ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP 2003), (2003), San Diego, California, USA.
- [12] G. Gigan and I. Atkinson, "Sensor Abstraction Layer: a unique software interface to effectively manage sensor networks," presented at the Third International Conference Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 07), Melbourne, Australia, (2007).
- [13] G. Gigan and I. Atkinson, "Towards a uniform software interface to heterogeneous hardware," presented at the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 08), Sydney, Australia, (2008).
- [14] Seeed Studio. (2012, August). *Seeed open hardware facilitator*. Available: http://www.seeedstudio.com/wiki/Seeeduino_Stalker_v2.2.
- [15] Arduino. (2012, November). *Arduino - Software*. Available: <http://www.arduino.cc/>.
- [16] T. Alsboui, A. Abuarqoub, M. Hammoudeh, Z. Bandar and A. Nisbet, "Information Extraction from Wireless Sensor Networks: System and Approaches," *Sensors & Transducers*, vol. 14, no. 2, (2012).
- [17] Synapse Wireless Inc., "Synapse Portal: Wireless Application Development Environment", Available: <http://www.synapse-wireless.com/snap-components-free-developers-IDE-tools/portal>, (2012) October.
- [18] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer - Overview", Available: <http://www.maximintegrated.com/datasheet/index.mvp/id/2812>, (2012).
- [19] SparkFun Electronics, "XBee Explorer USB - SparkFun Electronics", Available: <https://www.sparkfun.com/products/8687>, (2010) July.
- [20] K. Jarvi, "RXTX", Available: http://rxtx.qbang.org/wiki/index.php/Main_Page, (2012) July.

