Accepted by 11th IEEE LISAT Conference. To be published in *Proceedings of IEEE LISAT*, May 1, **2015**, pp. 1-6

A Wireless Tracking System for At-home Medical Equipment during Natural Disasters

David Li, *Member, IEEE*
Department of Science
Commack High School
1 Scholar Lane, Commack, NY 11725
Davidli27606@gmail.com

***Abstract*—Electricity-operated durable medical equipment (DME), such as ventilators, dialysis machines, and patient monitoring devices, are life-supporting machines used extensively by patients at home. While convenient and economical, at-home use of DME is susceptible to power outages, especially the ones caused by natural disasters that often occur in large area and for a long duration. There is little existing technology allowing hospitals to monitor DME-dependent patients without using the current infrastructure, such as the landlines, the cell towers, Ethernet cable or the Internet. Reported herein is a novel wireless system that utilizes a radio ad hoc network to automatically report the patient's information and location, and the DME information and status to a nearby hospital when a power outage is detected. This system consists of two parts: a hospital-based receiving device, called the Base Station node, and multiple transmitting devices, called User Nodes, each connected to the DME at patients' homes. The Base Station and User Nodes is each built with a Teensy® microcontroller, a GPS receiver module, and an Xbee® radio implementing the Zigbee® protocol. Additionally, each User Node contains a status LED and an internal lithium-ion battery connected by a charge controller. User Nodes are programmed to obtain the GPS location of the patient, monitor the DME status, communicate with nearby nodes, transmit the data and relay information to the Base Station through the radio ad hoc network the nodes form in the case of a power outage. The Base Station device is programmed to receive and convey the information transmitted from the User Nodes to a nearby hospital's patient monitoring computer through a USB connection. This system works without relying on the infrastructure, and allows hospital staff to know the information and locations of DME and their users and provide help needed during power outages.**

Keywords-Ad hoc Network, DME, Durable Medical Equipment, GPS, Radio, Tracker, Tracking System, Wireless, Zigbee, Xbee

I. INTRODUCTION

Durable medical equipment (DME) is any medical device used at home by patients for monitoring and/or treating diseases [1]. There are two types of DME: passive equipment and active equipment, the latter reliant on electricity to operate. Life-supporting active DME include dialysis machines, ventilators, oxygen concentrators, etc. [2]. At-home use of DME is not only convenient and economical, but also leads to a better quality of life for the patient. In a 2013 survey, the World Health Organization (WHO) estimated that in Japan alone, there are 13,000 DME in use, namely 101 DME users per million population [3]. DME are heavily used in the United States although a specific number is not available due to privacy laws [3].

Despite aforementioned benefits, at-home DME are susceptible to power outages, especially those caused by natural disasters. During difficult times like this, the DME-dependent patients had to face the life-threatening situation because their machines had stopped functioning. While most at-home DME are equipped with integrated batteries to keep them functioning during power outages, their rechargeable batteries typically last only 1 hour with lead-acid batteries and 2-3 hours with newer lithium-ion batteries [4]. Thus, there is a critical need for a means of communication between the medical staff at a hospital and patients at home during natural disasters without needing current infrastructure such as landlines or cell towers that are often unavailable during natural disasters. Aware of the severity of this problem, the Assistant Secretary for Preparedness and Response (ASPR) of the U.S. Department of Health & Human Services through its partner, www.innocentive.com, launched a challenge in 2013 to seek ideas that might solve this communication issue [5].

Although there are several commercially available general purpose trackers or locators [6], [7], and some DME even have integrated reporting units, none of these are operable when the infrastructure is disabled, because they all rely on the cell phone services and/or Internet connectivity. Hence, there needs to be a means of communication, which does not use the current infrastructure, between the at-home DME-dependent patients and the hospital staff during natural disasters. Reported herein is a novel DME tracking system utilizing a radio ad hoc network for transmitting data.

II. MATERIALS AND METHODS

A. Materials

The Xbee® shield was purchased from Sparkfun [8]. The Teensy® 3.1 development board was ordered from PJRC [9]. The GPS shield is a generic version and bought from Ebay. All other parts and tools were obtained from RadioShack® and Home Depot®.

The DME tracking system described herein is patent pending.

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

B. Engineering goal

To solve the communication issue when the infrastructure is disabled and help DME-dependent patients, it was proposed to engineer a novel DME tracking and reporting system based on wireless nodes with radios following the Zigbee® (IEEE 802.15 standard) specifications [10], operating at the frequency band of 2.4 GHz and consuming little power, though transmitting at short distances and at low data rates [11]. This system would comprise of two parts: multiple transmitting devices located in patients' homes and connected to patients' DME, called User Nodes, for gathering relevant live data to be transferred to the hospital and one central receiving device located in a local hospital, called the Base Station, for collecting the patient and DME information sent by the User Nodes. The system would have a modular design and scalable implementation, providing flexibility for further optimizations (for example, substituting the current radio module with other radios that have the same interface). Furthermore, each User Node would also have the feature of an internal rechargeable battery, charging when power line voltage is present, at which time the User Nodes are only able to relay other nodes' information. When a break of AC power supply is detected at a certain location, the user node would send information towards the Base Station in the hospital. The information includes patient information (i.e., name, age, disease, type and brand of DME being used, etc.), GPS location of the patient and DME, and the power outage status (i.e. how long the DME has been running using battery power and how much battery life is remaining). All patient data and information would be encrypted with symmetric-key encryption so that only the administering hospital could receive and decrypt the information in compliance of the HIPAA laws [12].

To make it widely accessible to the hospital and to the patient, the tracking system would be inexpensive to produce and maintain, also incurring no monthly fee associated with the cell phone services.

C. Hardware Design and Assembly

The proposed hardware was based on the Teensy® version 3.1 Development board [9], having an MK20DX256 32-bit microprocessor based on ARM Cortex-M4 and 256KB of flash storage and 64KB of RAM. This board has 3 standard asynchronous serial ports (protocol: 8 data bits, 1 stop bit and no parity), in addition to a USB programming ports capable of transmitting data in 4800 Baud increments. These serial ports were used to communicate with the radio, receive GPS information and retrieve information from the DME. This board was chosen due to its easiness of prototyping and relatively low cost. Additionally, it is programmable with the C language, which was used to program all User Nodes and the Base Station.

The radio used to communicate among modules was the Xbee® Pro Series 1 Point-to-Point device for communicating with IEEE protocol 802.15.4. It has a rated power output of 1 mW and can transmit 90 m within line of sight, either broadcasting its information or unicasting to a radio whose serial number is the same as the set destination address [13]. Moreover, the module inputs and outputs data through a standard serial interface at 115.2 kilobits per second.

The GPS module used in this prototype is a generic GPS board that uses the u-blox6 GPS module and interfaces with the microcontroller through serial interface at 9600 bits/second [14]. Its horizontal position accuracy is within 2.5 m and outputs its data with NMEA GPRMC statements, which includes geographic location and UTC time.

Figure 1A shows the hardware design of the User Node and Base Station, both using the same assembly. Figure 1B is a picture of the physical setup of the hardware.

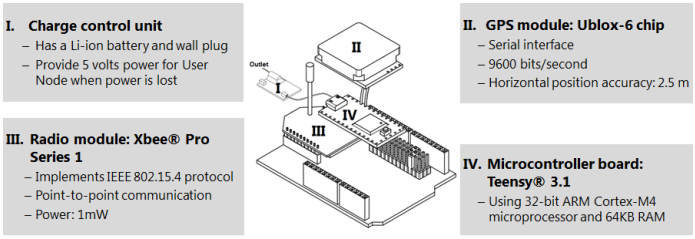


Figure 1A. Hardware design of the User Node and Base Station.

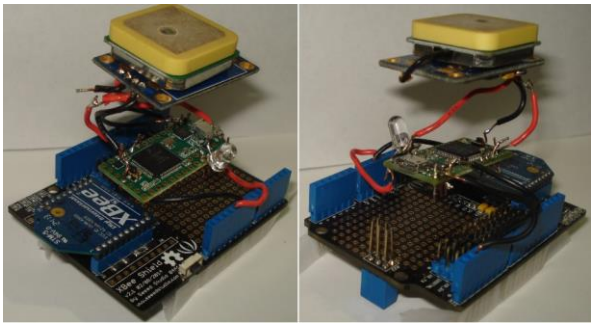


Figure 1B. Physical setup of the hardware.

D. Low Level Software Design

The program for the User Nodes was designed first. In order to have a platform on which to program the algorithm for implementing the routing protocol, low level drivers needed to control the Xbee® radio module and the GPS were written as the manufacturers either did not provide the driver code or/and the code was not well documented. After each driver was created, the driver algorithms were tested for accuracy and bugs, with each of them being debugged after being created. Firstly, a generic serial reader code was created to access the serial interfaces which was able to read the serial ports for data and output the data in variable `stringOut`, containing a maximum amount of characters defined at the beginning of the code by the constant, `MAX_READ`. Since the serial port is asynchronous and the node is without knowledge of the size of the upcoming information, the Serial was read at

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

```
typedef struct
RequestPacket{
  uint32_t packetType;
  uint32_t myHaddress;
  uint32_t myLaddress;
  uint32_t
magicNumber;
}
RequestPacket;
```

```
typedef struct
ReplyPacket{
  uint32_t packetType;
  uint32_t myHaddress;
  uint32_t myLaddress;
  float Latitude;
  float Longitude;
  uint32_t
magicNumber;
}
ReplyPacket;
```

```
typedef struct DataPacket{
  uint32_t packetType;
  uint32_t myHaddress;
  uint32_t myLaddress;
  float Latitude;
  float Longitude;
  float destinationLatitude,
  float destinationLongitude;
  char data[256];
  uint32_t magicNumber;
}
DataPacket;
```

Figure 2. Fundamental data structures used in the routing protocol implementation.

the same rate at which it was being filled, delaying the reading to accommodate a few characters to fill the 64-Byte Serial buffer. This is to ensure that the Serial buffer neither overflows nor becomes prematurely empty, at which point the reading stops and truncates the other information left in the Serial buffer. All codes were written modularly to support customization according to the platform and the usage. The serial reader code was a dependency for both the Xbee® radio driver and the GPS radio driver as it read their outputs.

The Xbee® driver code, based on the AT commands of the Xbee® radio [15], was designed to change the addressing of the radio and obtain information about the radio, such as its unique serial number. This would enable Xbee® module to enter either unicasting or broadcasting mode. The working of the Xbee® driver code was tested by running the commands and assessing their effectiveness through software on the computer released by the Xbee® module's manufacturer, which reads parameters of the Xbee® module *via* the USB port [16].

The GPS driver code detected the NMEA sentences transmitted by the GPS module and separated them into a queue to be processed.

After the driver code were implemented, higher level functions that used these drivers were created; such as a function to send a character array (i.e. `char*`) to a specific address of Xbee® radio modules. This code was verified by attempting to send a message to a second node through using the serial number of the receiving node's Xbee® serial number and verifying the text's integrity. This test was performed at short distances to lower the chance of the message being lost in the transmission process.

The code used to parse the GPS data into latitude, longitude and timestamp was written and it was tested through obtaining the raw data from the GPS module and parsing them using a reputable online NMEA decoder [17], with the results of the written NMEA decoder compared to that online.

E. High Level Software Design and Protocol Implementation

While there is documented, proprietary software to create a mesh network using the Xbee® radios known as DigiMesh® [18], such a protocol for routing was not used for the interest of providing a modular platform on which other, more efficient routing protocols could be easily implemented [19].

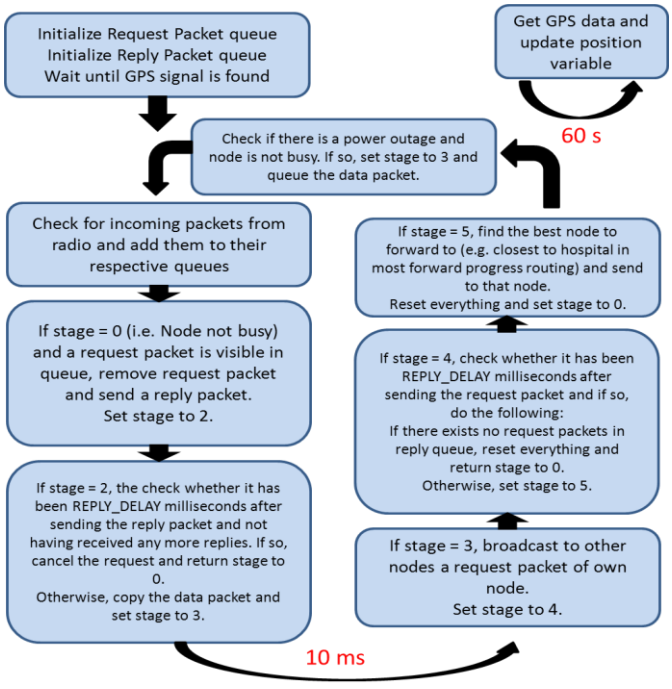
The algorithm implemented is a greedy geographic routing, whereby each node polls nearby nodes and the data is sent to the node closest to the final destination [20]. The algorithm is stateless and reactive, with the nodes not requiring information about previous nodes the packet was routed through and also dynamically routed. However, due to the modularity of the program, the routing algorithm can be easily changed. Following the routing protocol, three types of information packets were used to perform the routing: the Request packet, the Reply packet, and the Data packet; which were in the form of a struct instance that was cast into a string before sending. The data structures of the three packets are shown in Figure 2.

Because some of the parameters in the packets may add null characters (i.e. byte value of 0) to the data, during the conversion of each struct object into a character sequence to be sent to other nodes, the data in the structs alone would not be sent completely because the streaming of data by default ends with the 0 character. Thus, before the sending of any packets, a function is used to scan each struct for 0 values and replace them with another character that is not present anywhere else in the struct. Then, the encoded sequence is wrapped by prepending two characters to it: the character replacing the 0 characters in the original string and an arbitrary character that was decided to be "|", or ASCII character number 124. Thus, the final wrapped packet is in the form of "<r>|<packet>", where <r> is the replacement character and <packet> is the final encoded packet. Similarly, a parsing algorithm will scan from the beginning of the string for the first occurrence of the "|" character and undo the replacement of the 0s, thus recreating the original packet.

A novel feature of this algorithm is the sorting of the packets and the verifying of their integrity. Each struct contains two elements constant across all objects of the type namely, `packetType` and `magicNumber`. The `packetType` shows the type of packet being decoded, being 0 for a Request packet, 1 for a Reply packet and 2 for a Data packet. Additionally, the `magicNumber` is a constant that is set at the beginning of the program and can vary among hospitals. The purpose of the magic number element is not only to create separate channels of communication (i.e. a node will not respond unless the magic number of the received packet is equal to its preset magic number), but it is also used to differentiate any received packet among the three types. The separating of received packets by packet type is achieved by casting the data into each of the 3 packet types and testing whether the `magicNumber` and `packetType` elements match for the given packet type.

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

The algorithm implemented runs continuously through a loop and polls the inputs and outputs. It was planned and implemented through the high level functions. Figure 3 is an overview of the algorithm, including the flags and variables.



Variables and Constants:
stage = 0 when the node is not busy with any routing or sending of data
stage = 2 when the node is waiting for a packet
stage = 3 when there is a data packet to send and the node has sent out a request packet
stage = 4 when node is waiting for replies to its request packet
stage = 5 when the node has received request packets and has found a forwarder.
REPLY_DELAY is the number of milliseconds node waits for replies.

Figure 3. An overview of the algorithm.

F. *Power Consumption Measurement*

To measure the power consumption of the DME tracking device, one node was connected in series with nine 1Ω ± 5% resistors, connected such that they are in 3 parallel groups of 3 series resistors with a final equivalent resistance of 1Ω ± 5%. While not decreasing the error, this arrangement was used to narrow the distribution of possible values towards the mean (i.e. create a higher probability that the resistor is closer to its marked value). Using an ATMEGA328-based microcontroller, the voltage across the resistive load was measured to the nearest millivolt and outputted through the USB serial port at a 5-ms interval. Simultaneously, the node performed either a standard data transmission or a GPS updating, in which it updates the latitude and longitude values, in addition to the UTC time based on NMEA sentences from the GPS module. The node in each of the tests was powered by a regulated 5V source. The wiring diagram of power consumption test was shown in Figure 4.

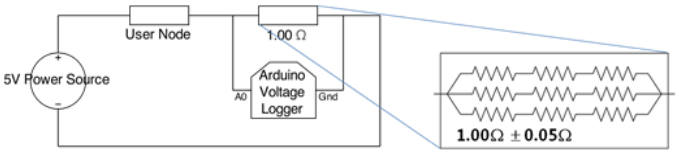


Figure 4. Wiring diagram of power consumption test.

G. *Field Test*

In order to ascertain the functionality of the routing protocol, as well as to determine the time used to relay the data packet, a field test was conducted, in which 5 User Nodes and 1 Base Station were placed linearly at a distance 90 meters apart from each other such that each node could only receive and transmit to adjacent nodes (Figure 5). The distance of 90 meters was used based on the radio range of the Xbee®

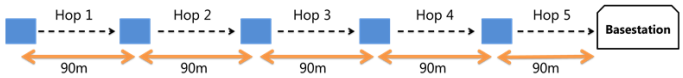


Figure 5. Experimental setup in field test.

module. The nodes were each tested such that they could only transmit to their immediate neighbors.

Additionally, the software of each node was modified by adding in a timestamp parameter into each of the data structures. This timestamp was used to collect information about the length of time taken for each transmission. This timestamp information was collected from the GPS module. The `getGPS` function was slightly modified to also record and update the UTC time variable in the program. The UTC time was found to the nearest second on the start of the second as the GPS module only outputs data on the beginning of every second. Since the GPS is read every 60 seconds, there needed to be a function written and tested to interpolate the exact UTC time, in addition to a milliseconds extension, to accurately record the time taken for information to be transferred. The time interpolation function takes into account the time interval recorded by the Teensy® since the last updating of the UTC time, and adds the time to the UTC time and the milliseconds extension. To test the accuracy of the interpolated time, two nodes were loaded with the same

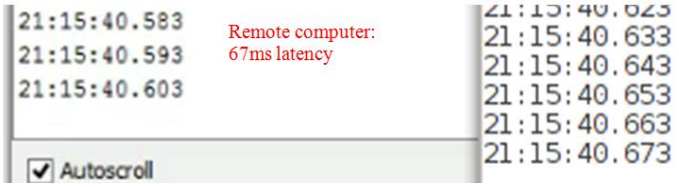


Figure 6. UTC time measurements from 2 nodes to verify the relative accuracy of the time interpolation function.

program, and were programmed to output their exact interpolated time *via* USB serial onto separate computers. Then, the second computer was connected to *via* remote desktop with a known latency such that the two serial

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

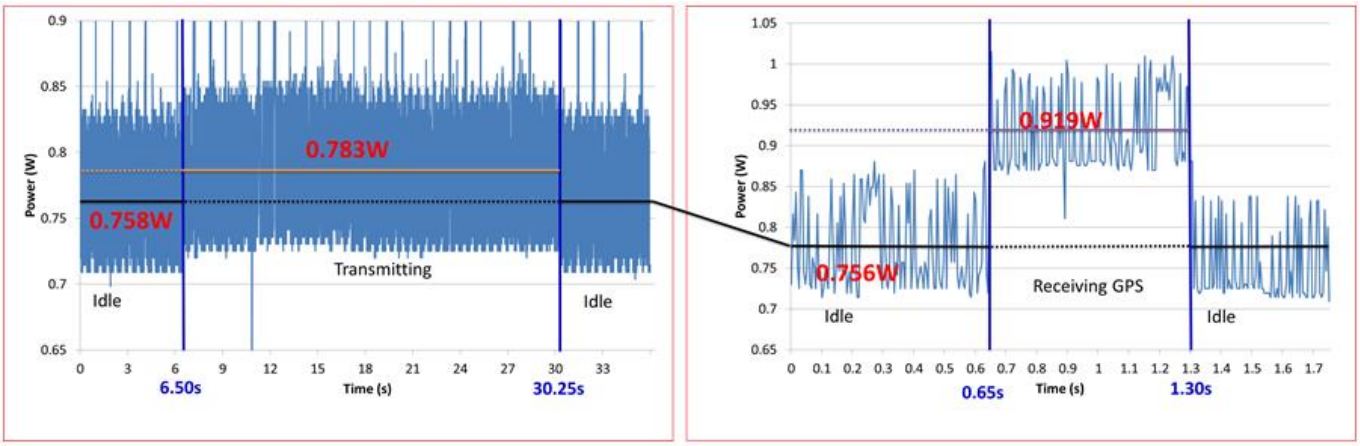


Figure 7. Power drawn from the User Node for idling, relaying packets, and receiving GPS.

windows were adjacent on a screen. Multiple screenshots of the two adjacent windows verified that, compensating for remote desktop latency, the two nodes displayed the same time, shown in Figure 6.

H. Network Test

After determining that the transmissions were reasonable, it was decided to test the implementation on the large scale with many nodes through an independently written simulation of the nodes implemented in the C language. This program was written to calculate the packet delivery ratio of an ad hoc network implementing the Most Forward in Radius protocol given the node density ($1/m^2$), range of each radio (m), and distance from origin node to the Base Station (m). The program calculates the value by simulating the routing of data 5,000 times, and returning the ratio between successes and total transmissions.

The parameters used for testing were as follows:

(a) The distance from origin node to the Base Station is 8 km, greater than the mean distance from a patient to its closest hospital [21].

(b) The range of each radio was varied from 1 km to 8 km, not including 8 km.

(c) The density of User Nodes/km^2 is dependent on demographics, but reasonable estimates for high, medium, and low density situations were $10/km^2$, $1/km^2$ and $0.2/km^2$, respectively.

The testing was performed with a tester function written in PHP script that initialized and called the main network simulator program with the parameters, and incremented the transmission range by 0.1 km. At locations where the values change, the data was collected in more details, with the transmission range being incremented by 0.01 km.

III. Results and Discussion

A. Hardware Power Consumption Test Results

The power drawn over time was calculated through $I_L = \frac{V_L}{R_L}$ and $P_T = I_T * V_T$, with total current equaling load current. While the voltage across the test resistor was not negligible, such difference did not impact the measured results, as the nodes are powered by a 3.3V regulator, and thus would not change current with the input voltage above the 4.5V threshold voltage of the 3.3V regulator.

The approximate power drawn was determined by finding $E = \int P(t)\,dt$ and dividing by Δtime. Figure 7 shows the power drawn from the User Node for idling, relaying packets, and receiving GPS. Table 1 shows the power drawn and time for each operation. The power consumptions for idling, relaying/transmitting data, and receiving data are 0.76 W, 0.78 W, and 0.92 W, respectively. The times needed for relaying/transmitting data and receiving data are 23.75 s and 0.65 s, respectively.

TABLE I. Power drawn and time for each operation

Operation	**Power**	**ΔTime**
Idle	0.76W	-
Relay or Transmit Patient Data	0.78W	23.75s
Receive GPS	0.92W	0.65s

According to the above results, the prototype power consumption was reasonable and these measurements can be used for optimizing the routing algorithm in the future.

B. Field Test Results

The results from the field simulation test were summarized in Table 2. The transmission was successful and time for each hop was averaged at 20.14 seconds.

TABLE II. Times of transmissions

Hop#	1	2	3	4	5	Average
Cumulative time (s)	20.09	40.24	60.38	80.53	100.68	
Time for each hop (s)	20.09	20.15	20.15	20.15	20.15	20.14

The field test not only shows that the routing algorithm could be viably implemented, but also shows the modularity of the software, as the data structures were modified without needing to modify any other parameters.

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

C. *Network Test Results*

Figure 8 shows the results of probability of transmission or packet delivery ratio (%) *vs.* transmission range of User Nodes (km) for high, medium, and low density network situations.

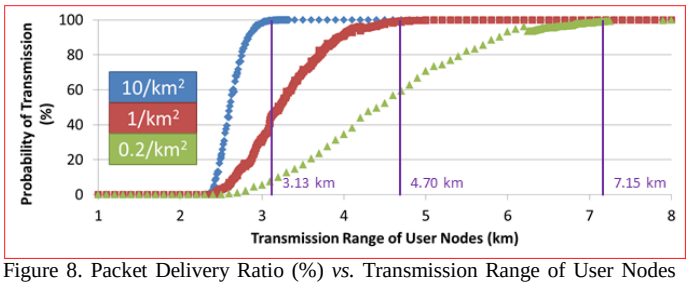


Figure 8. Packet Delivery Ratio (%) *vs.* Transmission Range of User Nodes (km) for high, medium and low density network conditions.

It was found that the routing implementation could work for all demographics, though is much less effective at lower densities, such as 0.2 users/km^2. To find the radio transmission ranges needed for the network, the highest value for transmission range whose packet delivery ratio was below 99% was found and was 3.13 km, 4.70 km, and 7.15 km for densities of 10 users/km^2, 1 user/km^2 and 0.2 users/km^2, respectively. In practice, software used by the hospital staff at the time of User Node registration may be able to determine whether extra auxiliary nodes nearby the User Nodes should be implemented to ensure that the node is connected with the remainder of the network.

It was determined that the model is accurate. Additionally, it is dimensionally valid.

IV. Conclusion

Given testing data, it was found that the prototype design of the DME tracking system was feasible to implement and would meet the requirement for securely transmitting patient data, location information, and the status of DME to a nearby hospital during power outages. Although the maximum radio range for the current pilot prototype was found to be 90 m, the advantage of modular design allows this proof-of-concept system to be easily scalable by simply employing more powerful radio modules or having specially placed forwarding nodes to facilitate the forwarding of information from more distant homes. In a medium patient density situation, for instance, a radio with an indirect (i.e. non line-of-sight) range of >4.70 km could be employed. Additionally, the implementation of this DME tracking system is relatively inexpensive, utilizing commercially available low cost general-purpose microcontrollers and general-purpose radios. When produced in one circuit in mass production, the cost will be even lowered.

This novel DME tracking system provides a critical link between the DME-dependent patients and the hospital during natural disasters. Further research and optimization of this system is underway.

Acknowledgment

Assistance and advice from my research teachers Mr. Richard Kurtz and Dr. Lorraine Solomon are greatly acknowledged. Thanks also go to Drs. Barbara and Frederick Kruger of IEEE, Prof. Xin Wang and Mr. Jose Cordova of Stony Brook University, Mr. Steven Hartman of Sterling Medical Devices, and Mrs. Alison Offerman-Celentano of Commack School District for helpful discussions and their suggestions and support.

References

[1] Durable medical equipment. *http://en.wikipedia.org/wiki/Durable_medical_equipment*

[2] National Coverage Determination (NCD) for Durable Medical Equipment Reference List (280.1). *http://www.cms.gov/medicare-coverage-database/details/ncd-details.aspx?NCDId=190&ncdver=2&NCAId=27&NcaName=Electrostimulation+for+Wounds&IsPopup=y&bc=AAAAAAAAAgAAAA%3D%3D&*

[3] Baseline country survey on medical devices, 2013 update: Medical Equipment–Total density per million population computed tomography. *http://gamapserver.who.int/gho/interactive_charts/health_technologies/medical_equipment/atlas.html*

[4] B. Norman. What Will You Do if the Power Goes Out? *http://alsn.mda.org/article/what-will-you-do-if-power-goes-out*

[5] HHS Press Office. HHS selects winners in idea challenge for emergency response. *http://www.hhs.gov/news/press/2014pres/02/20140220a.html*

[6] GPS Trackers for Today's World. *http://www.pocketfinder.com/*

[7] Zoombak Personal GPS Locators. The next generation Zoombak is eZoom. *http://www.securusgps.com/zoombak.aspx*

[8] SparkFun Xbee shield. *https://www.sparkfun.com/products/12847*

[9] Teensy USB Development Board. *https://www.pjrc.com/teensy/*

[10] ZigBee® Alliance. *http://www.zigbee.org/*

[11] ZigBee. *http://en.wikipedia.org/wiki/ZigBee*

[12] Health Information Privacy. *http://www.hhs.gov/ocr/privacy/*

[13] Xbee® 802.15.4. *http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#specs*

[14] NEO-6 u-blox 6 GPS Modules. *http://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf*

[15] Xbee Command Reference Tables. *http://examples.digi.com/wp-content/uploads/2012/07/XBee_ZB_ZigBee_AT_Commands.pdf*

[16] XCTU. *http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu*

[17] Online NMEA Tools. *http://freenmea.net/*

[18] Xbee® DigiMesh® 2.4. *http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-digimesh-2-4#overview*

[19] X. Xiang, X. Wang, and Z. Zhou, Self-adaptive on-demand geographic routing for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 11(9), Page(s): 1572-1586, Sept, 2012.

[20] Geographic routing. *http://en.wikipedia.org/wiki/Geographic_routing*

[21] J. Nicholl, J. West, S. Goodacre, and J. Turner, The relationship between distance to hospital and patient mortality in emergencies: an observational study. *Emerg. Med. J.*, 24(9), Page(s): 665–668, Sept, 2007.