Remote Sensing System for Cultural Buildings Utilizing ZigBee Technology

Jingcheng ZHANG, Allan HUYNH, Qinzhong YE and Shaofang GONG

Department of Science and Technology, Linköping University,

Norrköping, 60174, Sweden

ABSTRACT

A wireless remote sensing system using the ZigBee standard is presented in this paper. This system is a wireless solution for monitoring purpose in cultural buildings in order to protect cultural heritage. The concept of this system utilizes ZigBee networks to carry and transmit data collected by sensors and store them into both local and remote databases. Thus, users can monitor the measured data locally or remotely. Especially, the power consumption is optimized to extend the lifetime of the battery-driven devices. Moreover, since the system has a modular architecture, it is easy to add extra services into this system.

Keywords: Modular system, power consumption, wireless sensor network, ZigBee

1. INTRODUCTION

Each cultural building is unique. People spend a lot of money every year on maintaining these buildings. However, still many ancient relics like paintings and wooden furniture are damaged due to the fact that there are too many objects to be maintained, which requires a lot of human resource and funding.

Research and effort have been put in this area and come out lots of good ideas [1]-[3]. Meanwhile, many methods are used to support cultural heritage preservation, i.e., mathematic simulation, mechanical reconstruction and data collection. However, according to the survey, most current solutions for data collections in buildings use cables and sensor nodes are mains-powered. This will result in an inevitable modification in the cultural buildings and it is un-recoverable. Especially for some cultural buildings lack of power outlet, it is very difficult to deploy such systems.

A ZigBee wireless data collection system can solve these problems. ZigBee is a global standard for wireless sensor networks [4]. A ZigBee network is designed for low data throughput, long battery lifetime and robust network. These are features suitable for data monitoring and control. The wireless devices ease very much the system installation. The low network throughput ensures a very long battery lifetime. Furthermore, the ZigBee network is considered to be a robust network. Thus, ZigBee not only has its obvious advantage over the wired solution, but also a good choice among wireless solutions in the cultural heritage area.

2. SYSTEM OVERVIEW

Fig.1 shows the system overview of the remote sensing system. ZigBee sensor networks are deployed in different cultural buildings. Sensor modules, i.e., end devices in the network are placed at the monitoring locations. For each sensor module, both temperature and humidity sensors are mounted. The sensor module senses and transmits collected data via the ZigBee network. The data is received by the ZigBee coordinator that communicates with a computer via the serial port. Moreover, a database is installed in the computer and the "local server" software receives the messages from the serial port, parses and stores the information into the database. The database installed in the computer is synchronized with the "main server" via Internet which provides web service to end users.

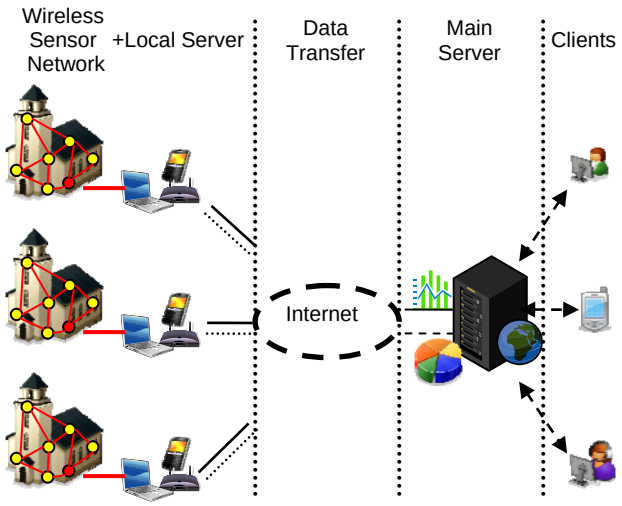

Fig. 1 Remote sensing system network view

For maintenance people working in different buildings, they can monitor the data locally via the local server. The web service provided by the "main server" can be used for a more generic purpose.

3. SYSTEM DESIGN

The system architecture is modularly designed to reduce the system interdependency. The system can be decomposed into the following packages from the software point of view:

- ZigBee network data collection package
- Local server data manipulation package
- Main server web service package

ZigBee network data collection package

Fig. 2 shows the ZigBee stack architecture defined by the ZigBee alliance [5]. It is a complete software system including hardware manipulation functionalities together with network functionality and application related interfaces. Sensor Working and Status Check functionality are implemented as two different endpoints embedded into the ZigBee architecture. The Sensor Working function is only enabled in end devices, since only end devices are mounted with sensors. However, the Simple Descriptor [4] for Sensor Working Endpoint is also registered in the ZigBee architecture for the communication purpose.

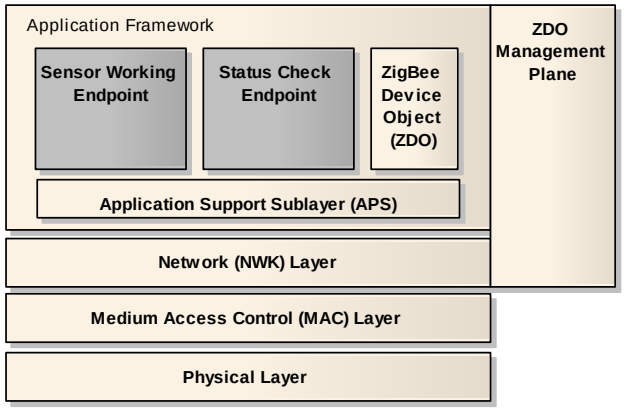

Fig. 2 ZigBee stack architecture

Sensor Working Endpoint:

Sensor modules (end devices) are battery-powered, so it is important to save the power as much as possible. Radio transmit or receive is one of the most power consuming operations in the sensor module, so a state machine is implemented to avoid the data re-transmit and let end device SLEEP in most of time.

As shown in Fig. 3, the state NWK_ESTABLISHED is triggered by the ZigBee stack when an end device successfully joins the ZigBee network. Meanwhile, the SENSOR_WORK_EVENT is also triggered to start state machine. After that, the sensor module tries to collect the temperature and humidity data from the sensor and go into the TEMP_ HUMI_SENSED state. If the return value of sensor reading is correct, the newly collected value is compared with the "old value" stored from last time. If the "new value" is equal to the "old value" (if a threshold of, e.g., 0.3°C is configured, the application will consider the temperature 24.5°C and 24.8°C are "same") the sensor module will not send out any message and go to the POWER_SAVING state. Otherwise a new message is formed and the state machine goes into the MSG_PKT_FORMED state. The end device will send the sensor data message to the coordinator of the ZigBee network. If the message sent successfully, the state machine will go into the PKG_SENT state and automatically jump to the POWER_SAVING state. At the POWER_SAVING state, a timer that triggers the SENSOR_START_WORKING event is set and then the end device goes into the sleep mode to save power. When the timer fires, the SENSOR_WORK_EVENT state is triggered, and the state machine will jump to SENSOR_START_WORKING state for the next round sensor module operation.

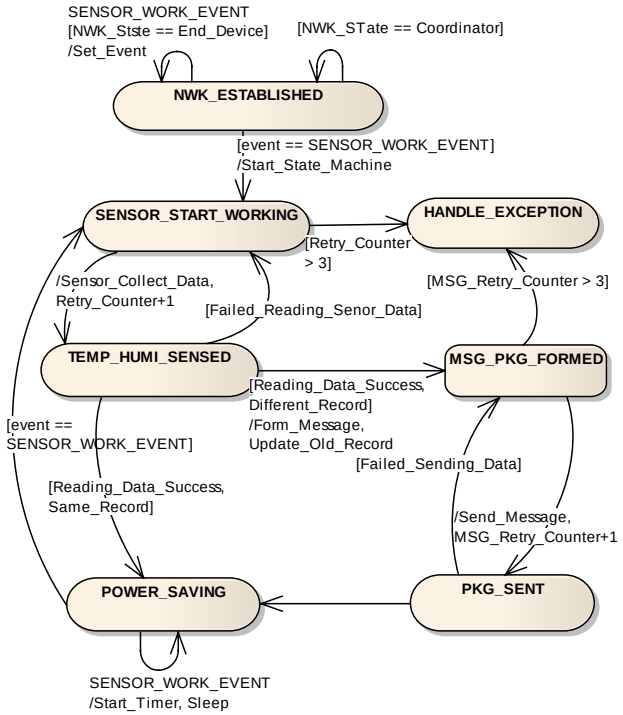

Fig. 3 Sensor working state machine

When assigning value at the state TEMP_HUMI_SENSED, it is important to assign the "old value" only at the state MSG_PKT_FORMED. This is to avoid sensor data from miss-reporting. For instance, the temperature reporting threshold is set to 0.3 °C, and the sensor detects 3 consecutive temperature changes: 11.5, 11.7, 11.9 °C. If the "old value" is updated each time when the temperature is sensed, 11.9 °C will not be reported to the coordinator since 11.9 °C is checked against 11.7 °C and the deviation is 0.2 °C, which is within the threshold. Thus the sensor will lose the sensitivity of detecting the "smooth" temperature deviation. However, if the "old value" is only assigned when it jumps to the MSG_PKG _FORMED state, 11.9 °C will be reported since the old value is 11.5 °C, and there is 0.4 °C temperature deviation which is outside the threshold.

Furthermore, sensor module exceptions are also handled in the state machine, as showed in Fig. 3. The state machine will go into the HANDLE_EXCEPTION state when the sensor fails to collect the sensor information for three times or the sensor module fails to send out data after retry 3 times at the state SENSOR_START_WORKING and MSG_PKG_FORM ED.

Status Check Endpoint:

In some situation, the humidity and temperature are kept stable for a long time, which is hard to identify the working status of end devices. A status check endpoint is implemented for both sensor modules (end devices) and routers for status checking. The principle of this status tracking application is "only talk to the upstream", which means that a device, either end device or router, only reports self-status to its upstream (parent).

For the routers that have children, an address table is implemented in Status Check Endpoint. Once the device registers itself in the network, Status Check Endpoint makes a "status report" to its parent (in this case, the parent is always a router or a coordinator for first level routers.) and the parent receives the record and adds its NWK address together with the current time stamp as a new entry into its address table. If the newly joined device is a router, the address table is also initiated to record the addresses of its potential "children". Since the end devices have no child, there is no address table initiated in the end device.

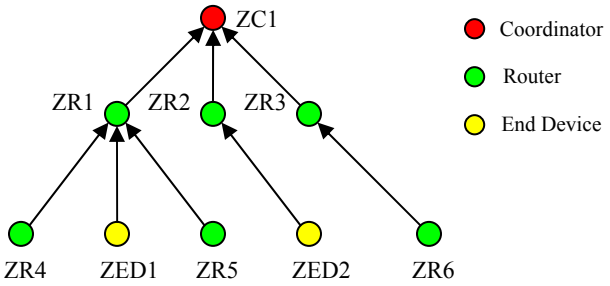

Fig. 4 ZigBee network topology example

Assume that the ZigBee network topology is shown in Fig.4, the arrow position presents the direction of the "status report". When the report is received by the "parent", a time stamp is marked on the report. The "parent" stores the report and the time stamp into the address table, i.e., ZC1 contains the address information of ZR1, ZR2 and ZR3. The ZR1 address table contains the address information of ZR4, ZED1 and ZR5. ZED1 does not have an address table and ZR5 has an empty address table.

"Status report" is repeated within a certain time interval and it can be configured at the initialization of the devices. Once the "parent" receives a report, it scans the address table to find the corresponding NWK_Address entry. If the entry is found, the time stamp field of this entry is updated to the current time stamp. Otherwise, a new entry is created in the address table. For the address table in each device, a "table scanner" is triggered by an internal timer. The "table scanner" will go through the address table when the timer fires. The interval of the timer that triggers the "table scanner" is longer than the interval of the "status report". Once each entry of the "address table" is checked by the "table scanner", following calculation is preformed:

$$D\_TIME = TIMESTAMP_{current} - TIMESTAMP_{record}$$

The value of D_TIME is the difference of entry's time stamp and current time stamp. If the value of D_TIME is larger than the interval of "status report", a warning message will be issued and sent to the coordinator. Once the coordinator receives this message, it will send this message to the local server to warn the maintainer.

Local server data manipulation package

The Local server is a Windows application running in a computer. It monitors the data from the serial port, parsing the message received and storing it into the database via ODBC [6]. It also receives the command from the user interface for data display.

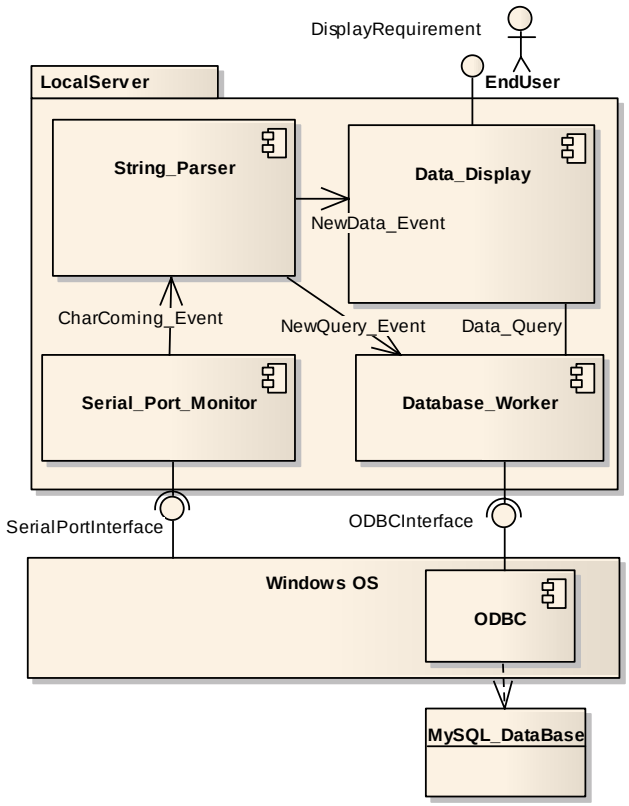

Fig. 5 Local server component view

Fig. 5 presents the component view of the local server architecture. If the local server is considered as a black box, it requires serial port and ODBC module interfaces from operating system and provides user interfaces for data monitoring and command receiving. The entire local server is event-driven. Each component in the Local server is implemented as a single thread. They receive pre-defined events sent from other components and process the data reported together with events.

The Serial_Port_Monitor component is the wrapper of the serial port API provided from Windows operating system. It is implemented for serial port management and communication with the ZigBee coordinator. The Serial_Port_Monitor component reports CharComing_Event once there is new

character received in the serial port. The String_Parser component receives CharComing_Event and stores the character received via the serial port. Once the received characters are successfully parsed, the String_Parser component sends out two events, NewData_Event and NewQuery_Event for Data_Display component and Database_Worker component respectively.

The Data_Display component receivs user command and displays data on demand with various formats. Once received a command from the user, the Data_Display component parses the command and queries the Database_Worker component to fetch the data from the database. The fetched data will be rearranged and displayed to the end user. The Data_Display component also provides "real-time" display function which allows user to browse data change in "real-time". When this "real-time" display mode is entered, the Data_Display component receives NewData_Event sent from String_Parser component and updates the display according to the new coming data. By using this method, the Data_Display component does not have to query the database all the time when displaying the real time data. This increases the local server performance since database invoking via ODBC consumes lots of system recource.

The Database_Worker component is the wrapper layer of the ODBC interface from Windows OS. The Databae_Worker component receives "NewQuery_Event" sent from the String_ Parser component and stores the new data into the database via the ODBC interface.

Main server web service package

The main server provides web service to users who can monitor the sensor data by browsing a webpage. As seen in Fig. 1, data used for web service comes from the database installed in the main server. Each database installed in the local server synchronizes data with the main server, so data in the main server may not be as "real-time" as that in the local server.

4. TEST SETUP

Two tests were so far done on the remote sensing system to test the system performance:

- Power consumption test: Test current consumption of the sensor module and calculate the battery life time.
- Network stability test: Verify the self-healing [7] function of the ZigBee network, and deployment plan in Linköping Cathedral in Sweden [8].

Power consumption test set-up

The test establishment was shown in Fig. 6. The device description list is shown in Table. I.

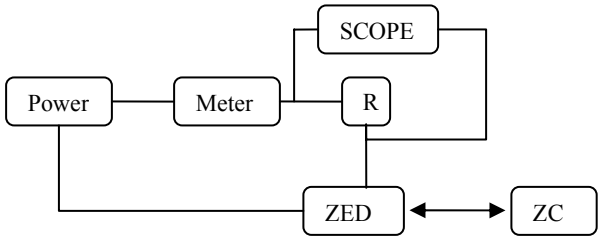

Fig. 6 Power consumption test establishment, see block definitions in Table I.

TABLE I

DEVICE LIST FOR POWER CONSUMPTION

Symbol	Description	Type
Power	Power Supply	Agilent E38833A
Meter	Current meter	HP34401A
R	Resistor	10 Ω
SCOPE	Oscilloscope	Agilent 54612D
ZED	End Device	ZigBee module
ZC	Coordinator	ZigBee module

This test is to verify the current consumption of the ZigBee sensor module (end device) when the sleep mode [10] is enabled. The software running in the End Deive is end point application Sensor_Working Endpoint and Status_Check_endpoint. After joining the network established by the coordinator, the end device performs the following operation:

- Sense temperature and humidity
- Send DataRequest [9]
- Status check
- Send sensor data

The current consumption for each of these operations is observed and calculated separately. Since the interval of the above four operations can be configured, the sensor module battery lifetime is calculated according to different interval configurations.

Network stability test

The ZigBee network is considered to be a robust wireless network because of its self-healing functionality. The network stability test is preformed in order to test the self- healing functionality in the ZigBee stack, a simple self-healing scenario is shown in Fig. 7.

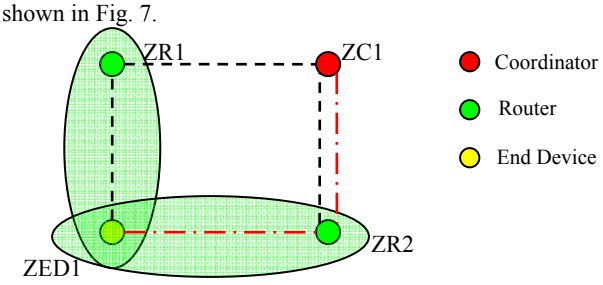

Fig. 7 Self-healing sample

The network was composed of the coordinator, two routers and one end device. The end device is within the propagation range of both routers. At the beginning, ZED1 communicates with ZC1 via ZR1. When ZR1 goes down, ZED1 will announce an "orphan" report to the network. ZR2, in this case, receives the report and takes over ZED1 and performs "rejoin" in the network.

5. TEST RESULT

Current consumption test result

Fig. 8-11 presents the snap-shot of all operations in the sensor module end device. The current consumption for each of these operations is calculated accordingly in Table II - V.

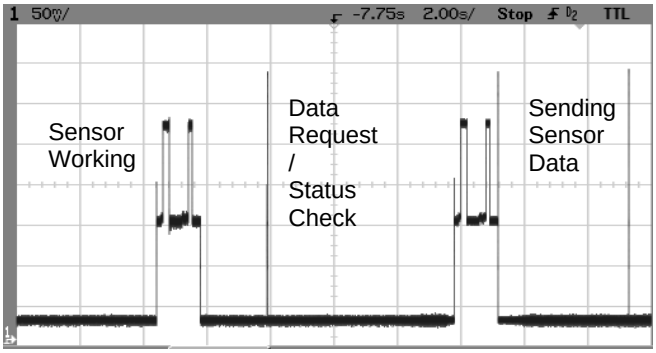

Fig. 8 End device operation oscilloscope snap-shot

Sense temperature and humidity

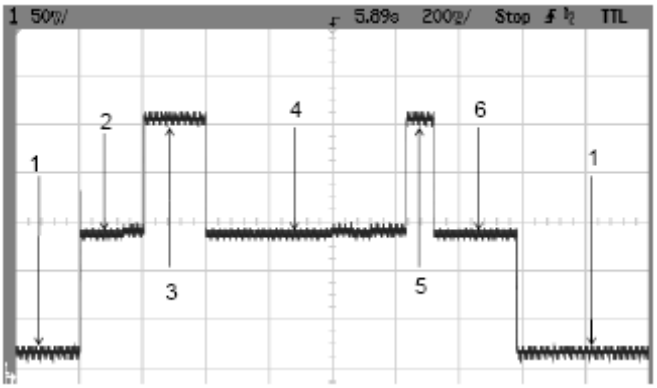

Fig. 9 Sensor data collection oscilloscope snap-shot

TABLE II

Sensor data collection power consumption summary

Interval	Description	Current (mA)	Duration (ms)
1	Power Saving/Sleep	0.0008	-
2	Working @ 32MHz	14.83	204
3	Sensing Humidity	26.95	196
4	Working @ 32MHz	14.83	632
5	Sensing Temperature	26.95	88
6	Working @ 32MHz	14.83	260
	TOTAL	23907mA*mS	1380

Send data request/ status check

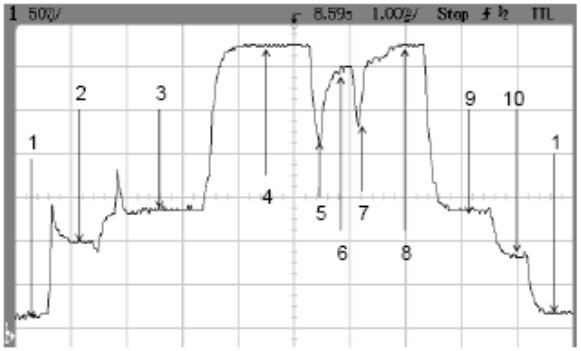

Fig. 10 DataRequest sending oscilloscope snap-shot

TABLE III

DATA REQUEST POWER CONSUMPTION SUMMARY

Interval	Description	Current (mA)	Duration (ms)
1	Power Saving	0.0008	-
2	Start Up @ 16MHz	1.10	0.920
3	MCU @ 32MHz	14.67	1.86
4	RX	33.96	1.92
5	RX->TX	22.48	0.2
6	TX	31.57	0.56
7	TX->RX	24.55	0.12
8	RX	33.96	1.18
9	Packet Processing @ 32MHz	14.67	1.19
10	Processing Sleep @ 16MHz	9.24	0.64
	TOTAL	186.1mA*ms	8.59

The power consumption in this case can vary due to step 4 in table III, since the ZigBee physical layer performs the CSMA-CA algorithm before sending any message. The Status Check endpoint also sends periodic report to its "parent", so the current consumption of this endpoint is the same as sending data

request. This is due to the fact that, from the hardware point of view, both operations involve computation and radio transmit/receive.

Send sensor data

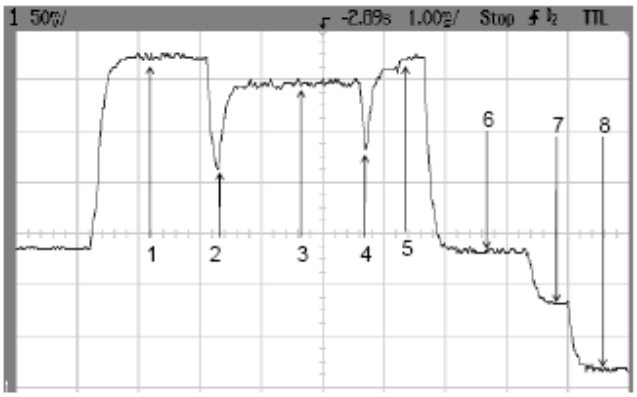

Fig. 11 Send sensor data oscilloscope snap-shot

TABLE IV

Send sensor data power consumption summary

Interval	Description	Current (mA)	Duration (ms)
1	RX	33.81	1.92
2	RX->TX	22.64	0.2
3	TX	31.09	2.3
4	TX->RX	24.71	0.12
5	RX	33.81	0.94
6	Packet processing @ 32 mHz	14.67	1.7
7	Processing Sleep @ 16mHz	9.41	0.62
8	Power Saving	0.0008	-
	TOTAL	206.5mA*ms	7.8ms

The sensor module end device is powered by a 1/2AA lithium battery with 1100 mAh capacity and 3.6 Voltage. By calculation, we have the following battery lifetime, as shown in table V:

TABLE V

Sensor module battery lifetime summary

Sensor module operation	Intervals	Intervals
DataRequest	Every 1 min	Every 20 min
Sensor Sensing	Every 6 min	Every 30 min
Sending Data	Every 30 min	Every 60 min
Status Report	Every 60 min	Every 60 min
Battery Lifetime	**657Days**	**3382Days**

*The calculation result is base on the ideal case. The battery leakage and network exception are not taken into consideration

Network stability test

The network stability test was preformed in Linköping Cathedral, one of the biggest churches in Sweden. By using routers, we established the network topology shown as Fig.12

End device ZED2 is within the propagation range of both ZR2 and ZR1. At the beginning, ZED2 is connected with ZR2. After shutting down ZR2, ZC still receives information sent from ZED2, but lost the connection with ZED3 and ZED4. After checking the address table in ZR1, a new entry for ZED2 is created in ZR1. This means that ZR1 takes over ZED2 as its "new child". However, ZED3 and ZED4 are out of the propagation rage, thus they cannot be taken over by ZR1.

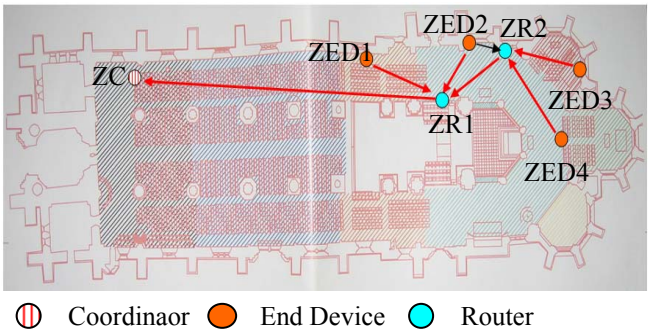

Fig.12. Network stability test network topology

6. DISCUSSION

The remote sensing system architecture is modularly designed, which minimizes the dependency among software packages. From the ZigBee network package point of view, the ZigBee network only routes messages to the correct destination. It is blind of the message content. Thus the network could be shared by all the endpoints for message transmit/receive without any modification. When adding another application, one simply adds another endpoint which is independent of existing sensor working and status tracking endpoints. Additionally, the state machine separates the sensor driver and sensor working logic. It is easy to reuse the state machine for other sensor operations. Moreover, when deploying a ZigBee network, it is good to locate end devices within more than one router's propagation range. This is to increase the system fault tolerance since if one router goes down, another router can take over the end devices.

From the local server point of view, it has the input from the serial port and parses the message it can recognize. The local server itself is blind to the ZigBee network. It only parses the string received from the serial port by the predefined method. For database manipulation in the local server, it is built on ODBC. This method eliminates the dependency of the database. User can change to any other database only if they provide the database driver to the Windows operating system. When there are new services added in the ZigBee network package, the local server only needs to add the corresponding table in the database for data storage. Moreover, introducing database into the local server facilitates people to monitor data locally if the Internet is not available. People can also use other database

monitor/manipulation tool to monitor the local data rather than the local server for database data manipulation, and export data to Excel, etc. For database, the local server is only a database consumer.

Observed from Table II, it is the sensor data collection that consumes most of the power. This is due to the nature of the sensor. In most of cases, using radio is the most power consumption operation. The implementation of the state machine eliminates the data resent and also provides the flexibility that lets the user set the threshold. Users will have the trade-off that the higher the accuracy is required for the humidity and temperature data, the more power is consumed. Also the "status check endpoint" which implements "only talk to upstream" principle dramatically reduces the throughput in the network. It also balances the processing "status report" burden to all parents rather than the single coordinator.

7. CONCLUSION

The remote sensing ZigBee network system is easy to configure and reuse due to the modular system architecture. The system also has a good flexibility for extending new services. The ZigBee network is stable and robust due to the self-healing functionality. End devices have a long battery lifetime which can last up to 10 years.

8. ACKNOWLEDGEMENT

Dr. Tor Broström at Gotland University is acknowledged for valuable inputs to the project. The Swedish Energy Authority (Energimyndigheten) is acknowledged for financial support of the project.

REFERENCEC

[1] R. Natalini, "Mathematical models for damage monitoring and restoration of cultural heritage", GAKUTO international series, Mathematic science and applications, 2005

[2] K. A. Papakonstantinou, C. T. Kiranoudis and N. C. Markatos, "Mathematical modeling of environmental conditions inside historical buildings". Energy and Buildings, Volume 31, Number 3, April 2000 , pp. 211-220(10)

[3] F.Clarelli, A.Fasano, and R.Natalini, "Mathematical Models for the Conservation of Cultural Heritage", 2009

[4] ZigBee Alliance, "ZigBee Specification and PRO Feature Set", 2007

[5] ZigBee Alliance, http://www.zigbee.org/

[6] Microsoft, "Open Database Connectivity (ODBC)", 2006 http://msdn.microsoft.com/ens/library/s9ds2ktb%28VS.80%29.aspx

[7] T.Angskun, G.E.Fagg, G.Bosilca, "self-healing network for scalable fault tolerant environment" Dept. of Computer Science, The University of Tennessee, Knoxville, USA,2006

[8] Linköpings Cathedral http://www.linkopingsdomkyrka.se/

[9] D.Gidlason, *ZigBee Wireless Networking,* Chapter 4, ZigBee Application, page 112-206, 2007

[10] Texas Instrument, "Z-Stack Developer's Guide", "Z-Stack ZCL API", "Application-Level Tuning of Z-Stack", "Method for Discovering Network Topology", "Power Management For The CC2430DB", "CC2430 Datasheet", "Serial Port Interface", 2009

[11] Texas Instrument, "Measuring power consumption with CC2430 & Z-Stack", 2004

[12] A.Denver, *Serial Communications in Win32*, December 11, 1995

[13] MySQL, *MySQL 5.1 Reference Manual* http://dev.mysql.com/doc/refman/5.1/en/

[14] David White, *Kenn Scribner, Eugene Olafsen MFC Programming with Visual C++ 6 Unleashed,* 1999.