An Android Terminal in TelosB Wireless Sensor Networks

Siquan Hu, Xiaoli Zhang
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China
husiquan@ustb.edu.cn

Hui Yao, Chundong She
Ruijie Networks Co., Ltd.
Beijing, China
yaohui@ruijie.com.cn

***Abstract*—In the deployment of a wireless sensor network, a portable device can be used on protocol analysis or in-situ network diagnose, which is more convenient than the traditional gateway solution. In this paper, based on the Android devolvement board and TelosB mote, the Android terminal was built to collect data from a wireless sensor network composed of TelosB motes. The middleware was implemented on this Android terminal to parse the sensor data and store the data into databases. The evaluation process verified the deign feasibility.**

Keywords-Android terminal, wireless sensor networks;TelosB; data gathering

I. INTRODUCTION

Internet of things (IOT) is an emerging trend in industry in recent years especially after economic depression. As one of the key technologies in IOT, wireless sensor networks can be used to monitor environment, industry devices, vehicles and even human bodies. In a typically deployment, a wireless sensor network is composed of many sensor nodes, which are connected via wireless star or mesh topology. Data are gathered by the sensor, and routed to a base station. Traditionally, a base station is connected to a computer, which may have three software components: a database to store the sensor data, a middleware to accept, interpret and store sensor data from the base station, GUI software to visualize the sensor data. In such kind of deployment, the computer is attached to the base station via serial or USB link, and it acts as a data center.

However, a portable device such as a smart phone or a tablet is more convenient in some circumstance. For instance, in a large-scale deployment of wireless sensor network, a portable device can display the in-net data and can be used a handheld diagnose equipment or protocol analyzer. Wireless body sensor network is another example showing the benefit of a portable device as a data gathering terminal for its small footprint. In such case, the vital body parameters are collected by the portable devices and alarm can be made when abnormality is detected.

Using portable device as a data gathering terminal is not a novel design in wireless sensor networks, Crossbow Stargate [1] and Linksys NSLU2 [2] have been popular gateways in wireless sensor network community. However, due to lack of display, this portable device can only be used as gateway, users can only view the gathered data on a computer connected to the gateway via IP network. So a portable device with a display will benefit the WSN diagnose or WBAN information visualization. Android tablets or Android mobile phones are a very efficient option to meet such demand.

Currently, Android is one of the most popular smart phone and tablet platforms [3]. Additionally, because of Android's open source and high flexibility, software developers can access to hardware and rich software easily. Some developers begin to use Android platform in wireless sensor networks field. Android smart phones were used to monitor the road surface in [4] in vehicle sensor networks. An android device is used to track heartbeat data in the body sensor network [5]. However, due to lack of WSN radio such as Zigbee [6] or other short range wireless in off-the-shelf android devices, above cases used Wi-Fi or Bluetooth and cannot be used directly as a data gathering terminal in wireless sensor networks.

In this paper, we present our work on building an android device to collect data from a wireless sensor network composed of many TelosB [7] motes. The application architecture and hardware issues are presented in part II; the middleware software on Android is presented in part III; Part IV is the evaluation result and Part V is the conclusion.

II. ARCHITECTURE AND HARDWARE

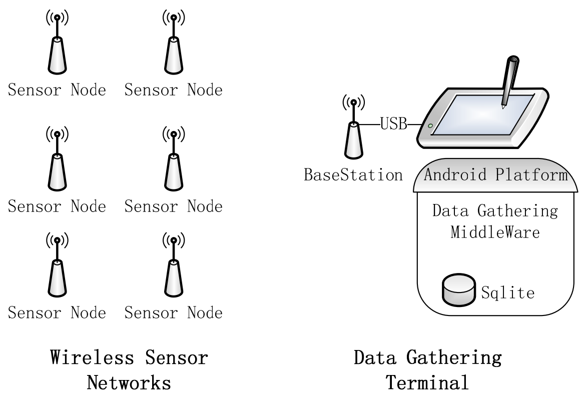


Figure 1. The deloyment Architecture of Android Terminal in WSN

An Android terminal in wireless sensor network is a special portable device deployed in the range of network to collect the sensor data or overhear the protocol data. The typical deployment architecture is illustrated in Fig. 1.

In our experimental wireless sensor networks, all nodes are TelosB motes. The nodes get local temperature, humidity and light readings periodically. The network is multi-hop and ad-hoc. The sensor data are sent to next hop by CC2420 radio. Any node who has received sensor data from its child will relay the data to next hop. Ultimately, all the sensor data are routed to the Android Terminal who has a attached CC2420 radio.

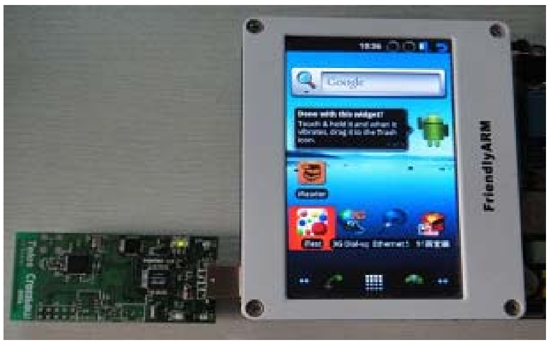


Figure 2. The hardare of Android Terminal

To simplify hardware set up, the Android Terminal is built up based on an Android development board and a TelosB mote illustrated in Fig.2. The Android development board is based on Samsung S3C6410 MCU. The TelosB mote is connected to the USB host port of the Android board. TelosB uses a FTDI chip to achieve USB-to-serial conversion. The default kernel of the Android system doesn't support it. So we rebuild the Linux kernel of the Android system after verifying the configuration in Fig.3. is assured.

```
CONFIG_USB_SUPPORT=y
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_FTDI_SIO=y
```

Figure 3. The kernel configuration for FTDI driver

To store the sensor data, a database is needed. There are two options for the designed Android terminal to use. One is use a traditional DBMS such as PostgreSql or Oracle on a server which can be accessed via Wi-Fi, the other is use local embedded database SQLite [8] on Android. The second option is preferred when a Android terminal is acting as a protocol analyzer or network diagnosis tool. Most Android system is shipped with SQLite. When SQLite is not available, it can be compiled on a Linux machine and installed easily on Android by copying the executable and library to android platform.

III. MIDDLEWARE DESIGN

After the virtual serial port received a sensor data packet, the Android terminal needs a software component to process the data. This could be a serial port demon or an in-memory process. It is a middleware as a bridge between the sensor network and the backend database. The main function of the middleware is receiving the sensor data, parsing the data and storing the data to the local or remote database.

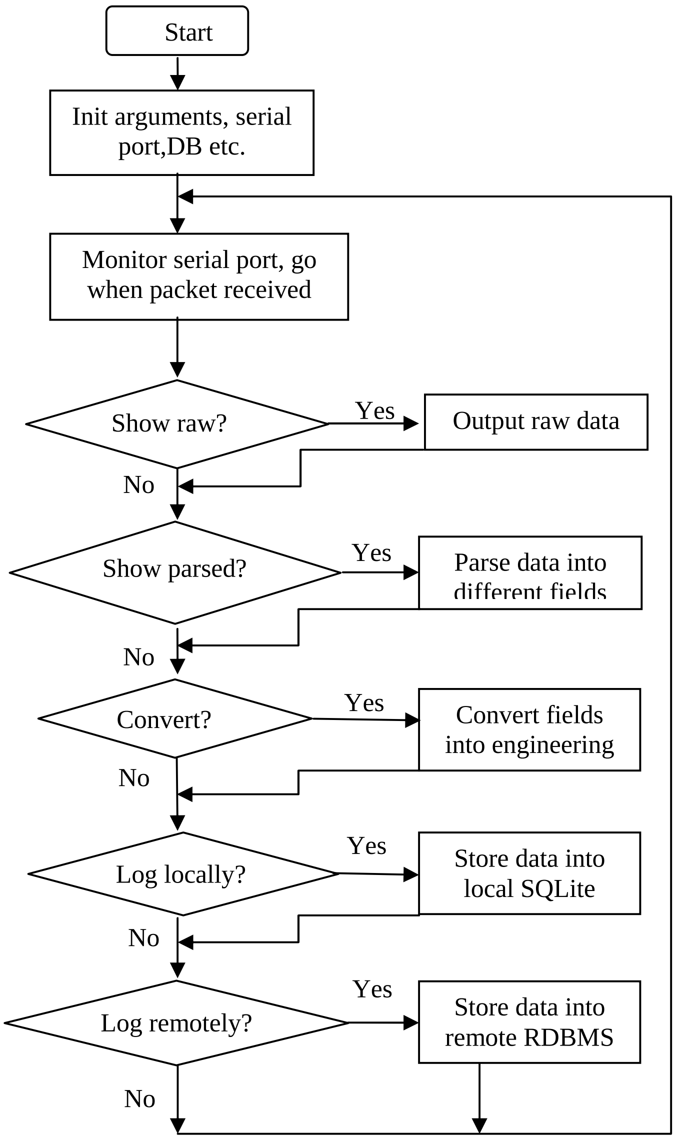

Start
Init arguments, serial port,DB etc.
Monitor serial port, go when packet received
Show raw?
Yes
Output raw data
No
Show parsed?
Yes
Parse data into different fields
No
Convert?
Yes
Convert fields into engineering
No
Log locally?
Yes
Store data into local SQLite
No
Log remotely?
Yes
Store data into remote RDBMS
No

Figure 4. The flowchart of the middleware

We have implemented the middleware in C language. The middleware will run in the Android system, which make the compilation different from that on standard Linux platform, although Android is based on the Linux kernel. This is because that Android uses its own C library – "Bionic Libc" rather than traditional glibc in Linux. To simplify the process to make a C program run on Android, we use static compilation to include all needed elements in the C executable. In this way, we can ignore the difference between Linux and Android, C program can run in Android without dependence on the dynamic link libraries in Android. The simplest way to achieve static compilation is use *–static* compilation flag. In our implementation, it is specified in the MAKEFILE of the middleware with a line as: *CFLAGS+=–static.*

The main flowchart is showed in Fig.4. The middleware is configured to run with some arguments. The arguments specify the requirement of serial port number and baud rate, display options, database options. Thereafter, the middleware will open the serial port with the specified arguments and listen to it. When a legal packet is received, the packet processing begins.

If the raw packet display option is enabled, the raw hexadecimal format will be print in command line. If the parsed display option is enabled, the packet will be displayed field by field, but each field is still in hex format, e.g. "temperature=0x1739", mostly showing the ADC results. If the option to convert the data is enabled, the hex data will be convert into engineering units, e.g. "temperature=25 degC". The conversion formula can be found in the sensor datasheet or the mote board user guide.

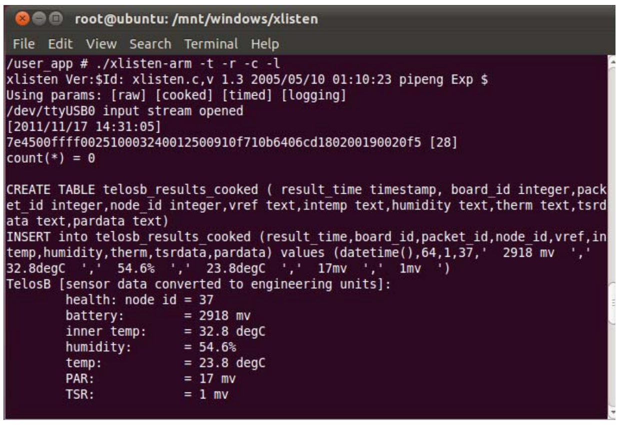
```
int xdb_execute(char *command) {
     char rc;  sqlite3 *db;     char *zErrMsg = 0;
     printf("%s\n",  command);
     //open the database
     rc = sqlite3_open(g_dbname,&db);
     if( rc ){
        fprintf(stderr, \
           "Can't open db: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        exit(1); }
     //execute the log comand
     rc = sqlite3_exec(db,\
        command,callback,0,&zErrMsg);
     if( rc!=SQLITE_OK ){
        fprintf(stderr,  "SQL error: %s\n", zErrMsg);
        sqlite3_free(zErrMsg);}
     //close the database
     sqlite3_close(db);
     return 0;
}

sprintf (command,"INSERT into %s (result_time,
   node_id,temperature,humidity)  values
   ( datetime(),%u,%u,%u)",table,data->nodeid, data-
   >temp, data->humidity,    data->therm);

xdb_execute(command);
```

Figure 5. Middleware code for storing data into local databse

There are two options related to storing the data to database. One is to store the data into local SQLite database; the other is to log the data into remote relational DBMS. The two options are similar in process logic except that remote database need be specified the database host name or IP address. Fig.5. gives the main process code for storing the sensor data into local SQLite database. Of course a table is needed beforehand; this table has fields for all the sensor readings and other helpful information such as timestamp for the packet, the sensor node identification, etc. When logging a coming packet into local SQLite, *xdb_execute()* will be called with a proper SQL stream - *command*. The SQL stream could be assembled by *sprinf()* function as illustrated in Fig.5.

IV. EVALUATION

To evaluate the development of the Android terminal meets the requirement of data gathering for Telos wireless sensor networks, a simple test deployment as Fig.1 is set up in the lab environment. The test procedure is carried out on a Ubuntu machine through ADB (Android Debug Bridge). ADB is a debug tool used on the development host to send shell command to the target Android board. The command will be executed on Android and the result is showed in adb shell of the development host.


root@ubuntu: /mnt/windows/xlisten
File Edit View Search Terminal Help

```
/user_app # ./xlisten-arm -t -r -c -l
xlisten Ver:$Id: xlisten.c,v 1.3 2005/05/10 01:10:23 pipeng Exp $
Using params: [raw] [cooked] [timed] [logging]
/dev/ttyUSB0 input stream opened
[2011/11/17 14:31:05]
7e4500ffff00251000324001250091 0f710b6406cd180200190020f5 [28]
count(*) = 0

CREATE TABLE telosb_results_cooked ( result_time timestamp, board_id integer,pack
et_id integer,node_id integer,vref text,intemp text,humidity text,therm text,tsrd
ata text,pardata text)
INSERT into telosb_results_cooked (result_time,board_id,packet_id,node_id,vref,in
temp,humidity,therm,tsrdata,pardata) values (datetime(),64,1,37,'  2918 mv  ','
32.8degC  ','  54.6%  ','  23.8degC  ','  17mv  ','  1mv  ')
TelosB [sensor data converted to engineering units]:
        health: node id = 37
        battery:        = 2918 mv
        inner temp:     = 32.8 degC
        humidity:       = 54.6%
        temp:           = 23.8 degC
        PAR:            = 17 mv
        TSR:            = 1 mv
```

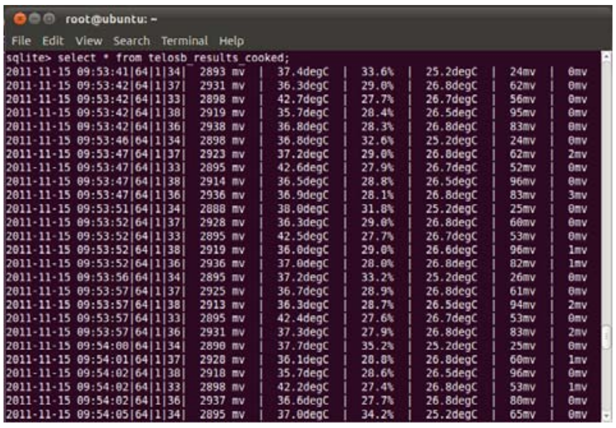
Figure 6. The running middleware displays the sensor data

Firstly, run "*adb devices*" to check USB OTG connection is ok and the adb service is running on the Android terminal. Then upload the middleware to the target Android terminal by "*adb push ./xlisten-arm /user_app*". Start adb shell and run the middleware. When the packets come from the TelosB sensor network, the middleware will process the data, display the data as Fig.6 and store them into database if needed. The screenshot shows that the middleware parse and convert the sensor data correctly

To verify the middleware store the data correctly, we let the middleware in android terminal run for a hour to collect enough quantity of sensor data in SQLite. Then start sqlite client in Android from adb shell, and execute sql statement to check whether the database table has correct data stored. The screenshot in Fig.7 showed the sensor data had been stored in the local SQLite correctly.

V. CONCLUSION AND FUTURE WORK

In the scenario of protocol analysis or in-situ diagnose of wireless sensor network, a portable Android terminal has advantage over traditional gateway solution. Based on the Android devolvement board and TelosB mote, the Android terminal is built to collect data from a wireless sensor network composed of TelosB motes in this paper. The middleware are implemented on this Android terminal to parse the sensor data and store the data into databases. The evaluation process verified the deign feasibility. Future work will expand current work into an Android smart phone or tablet and use native Android GUI to display the sensor data. The protocol analyzer or other application will be also implemented.



```
sqlite> select * from telosb_results_cooked;
2011-11-15 09:53:41|64|1|34|   2893 mv   |   37.4degC   |   33.6%   |   25.2degC   |   24mv   |   0mv
2011-11-15 09:53:42|64|1|37|   2931 mv   |   36.3degC   |   29.0%   |   26.8degC   |   62mv   |   0mv
2011-11-15 09:53:42|64|1|33|   2898 mv   |   42.7degC   |   27.7%   |   26.7degC   |   56mv   |   0mv
2011-11-15 09:53:42|64|1|38|   2919 mv   |   35.7degC   |   28.4%   |   26.5degC   |   95mv   |   0mv
2011-11-15 09:53:42|64|1|36|   2938 mv   |   36.8degC   |   28.3%   |   26.8degC   |   83mv   |   0mv
2011-11-15 09:53:46|64|1|34|   2898 mv   |   36.8degC   |   32.6%   |   25.2degC   |   24mv   |   0mv
2011-11-15 09:53:47|64|1|37|   2923 mv   |   37.2degC   |   29.0%   |   26.8degC   |   62mv   |   2mv
2011-11-15 09:53:47|64|1|33|   2895 mv   |   42.6degC   |   27.9%   |   26.7degC   |   52mv   |   0mv
2011-11-15 09:53:47|64|1|38|   2914 mv   |   36.5degC   |   28.8%   |   26.5degC   |   96mv   |   0mv
2011-11-15 09:53:47|64|1|36|   2936 mv   |   36.9degC   |   28.1%   |   26.8degC   |   83mv   |   3mv
2011-11-15 09:53:51|64|1|34|   2888 mv   |   38.0degC   |   31.8%   |   25.2degC   |   25mv   |   0mv
2011-11-15 09:53:52|64|1|37|   2928 mv   |   36.3degC   |   29.0%   |   26.8degC   |   60mv   |   0mv
2011-11-15 09:53:52|64|1|33|   2895 mv   |   42.5degC   |   27.7%   |   26.7degC   |   53mv   |   0mv
2011-11-15 09:53:52|64|1|38|   2919 mv   |   36.0degC   |   29.0%   |   26.6degC   |   96mv   |   1mv
2011-11-15 09:53:52|64|1|36|   2936 mv   |   37.0degC   |   28.0%   |   26.8degC   |   82mv   |   1mv
2011-11-15 09:53:56|64|1|34|   2895 mv   |   37.2degC   |   33.2%   |   25.2degC   |   26mv   |   0mv
2011-11-15 09:53:57|64|1|37|   2925 mv   |   36.7degC   |   28.9%   |   26.8degC   |   61mv   |   0mv
2011-11-15 09:53:57|64|1|38|   2913 mv   |   36.3degC   |   28.7%   |   26.5degC   |   94mv   |   2mv
2011-11-15 09:53:57|64|1|33|   2895 mv   |   42.4degC   |   27.6%   |   26.7degC   |   53mv   |   0mv
2011-11-15 09:53:57|64|1|36|   2931 mv   |   37.3degC   |   27.9%   |   26.8degC   |   83mv   |   2mv
2011-11-15 09:54:00|64|1|34|   2890 mv   |   37.7degC   |   35.2%   |   25.2degC   |   25mv   |   0mv
2011-11-15 09:54:01|64|1|37|   2928 mv   |   36.1degC   |   28.8%   |   26.8degC   |   60mv   |   1mv
2011-11-15 09:54:02|64|1|38|   2918 mv   |   35.7degC   |   28.6%   |   26.5degC   |   96mv   |   0mv
2011-11-15 09:54:02|64|1|33|   2898 mv   |   42.2degC   |   27.4%   |   26.8degC   |   53mv   |   1mv
2011-11-15 09:54:02|64|1|36|   2937 mv   |   36.6degC   |   27.7%   |   26.8degC   |   80mv   |   0mv
2011-11-15 09:54:05|64|1|34|   2895 mv   |   37.0degC   |   34.2%   |   25.2degC   |   65mv   |   0mv
```

Figure 7. The sensor data are stored in the SQLite by the middleware

ACKNOWLEDGMENT

This project is sponsored by National Key Technology R&D Program Grant #2012BAH25B02, and the Guangdong Province University-Industry Cooperation Grant #2011A090200008, and the Scientific Research Foundation, Returned Overseas Chinese Scholars, State Education Ministry.

REFERENCES

[1] Stargate: A platform X project, http://platformx.sourceforge.net/.

[2] K.Laufer, G.K.Thiruvathuka and C.R.Martinez-Eiroa,"Putting a Slug to Work", Computing in Science Engineering,vol.11, no.2,pp62-68, 2009.

[3] Android, http://www.android.com

[4] G. Strazdins, A. Mednis, G. Kanonirs, R. Zviedris, and L. Selavo, “Towards Vehicular Sensor Networks with Android Smartphones for Road Surface Monitoring,” 2nd International Workshop on Networks of Cooperating Objects (CONET’11), Electronic Proceedings of CPSWeek'11, 2011.

[5] M. Mitchell, F. Sposaro, A. A. Wang, G. Tyson, “BEAT: Bio-Environmental Android Tracking”, 2011 IEEE Radio and Wireless Symposium (RWS), pp.402-405,2011.

[6] S.S.Riaz Ahamed, "The role of zigbee technology in future data communication system", Journal of Theoretical and Applied Information Technology,Vol.5, No.2, pp.129-135. 2009.

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research", IEEE IPSN 2005 , pp. 364-369, 2005

[8] SQLite, http://www.sqlite.org.