Vol.2013-IOT-23 No.2
2013/9/27

Wiki で設定やプログラムを記述可能なセンサネットワークシステム

山之上卓[†1]　小田謙太郎[†1]　下園幸一[†1]

センサやアクチュエータを接続した Arduino ボードと Android 端末を組み合わせた遠隔装置と Wiki ソフトウェアで構成されたセンサネットワークシステムについて述べる．遠隔装置の設定やプログラムは LAN やインターネット上のサーバで動作する Wiki ページで記述できる．遠隔装置は設定やプログラムにより，センサデータを Wiki ページに書き込んだり，Wiki ページ上のデータや設定により遠隔装置のアクチュエータを動かしたり，複数の Wiki ページに書き込まれたセンサデータを集計し，別の Wiki ページに書き込んだりすることができる．

A Sensor Network System which can be Configured and Programmed by Wiki pages at Remote Places

TAKASHI YAMANOUE[†1]　KENTARO ODA[†1]　KOICHI SHIMOZONO[†1]

A sensor network system, which can be configured and programmed by Wiki pages at remote places is discussed. This system consists of mobile terminals and web sites with wiki software. The mobile terminal can be controlled by a series of commands which is written on a wiki page. The mobile terminal may have a data processor and the series of commands may have a program which controls the processor. The mobile terminal can read data from not only the sensors of the terminal but also wiki pages on the Internet. The input data may be processed by the data processor of the terminal. The processed data may be sent to a wiki page. The mobile terminal can control the actuators of the terminal by reading commands on the wiki page or by running the program on the wiki page. In addition, a wiki site can have a program that reads the page and outputs information such as a graph.

1. Introduction

A wiki[17] is a web site that allows the easy creation and editing of any number of interlinked web pages via a web browser and can be used as a means of effective collaboration and information sharing. Wikipedia[16] is a well-known wiki site.

If a wiki is friendly to people, it also must be friendly to machines. If a machine can read and write data on a wiki page automatically, people can obtain much more beneficial information. People also can easily control machines through the wiki page. Not only machine-to-people or people-to-machine, but also machine-to-machine communication, must be achieved easily. If such communication can be achieved, the wiki can be much more useful. For example, if a well-known wiki can be used to connect sensors in a sensor network, building one's own sensor network becomes easier.

To confirm the above presumption regarding the usefulness of wikis, we are developing a sensor network system using wiki software. This system consists of mobile terminals and web sites with wiki software. A mobile terminal of the system consists of an Android[8] terminal and an Arduino[10] board with sensors and actuators. The mobile terminal may have a data processor. This processor is controlled by the program which is written on a wiki page. The mobile terminal can read data from not only the sensors in the Arduino board but also wiki pages on the Internet. The input data may be processed by the data processor of the terminal. The processed data may be sent to a wiki page. The mobile terminal can control the actuators of the Arduino board by reading commands on the wiki page or by running the program on the wiki page. This feature makes this system flexible.

2. Overview of the System

Figure 1 shows an overview of the system. The system consists of mobile terminals and *PukiWiki* sites. PukiWiki is a popular wiki software in Japan. A mobile terminal consists of an Android terminal and an Arduino board with sensors (or an Arduino board with actuators). ADK is used to control the Arduino board. A mobile terminal of the sensor network system also includes the data processor. A mobile terminal reads a series of commands on a wiki page of a PukiWiki site and interprets these commands. The mobile terminal reads sensor data, write acquired data on the wiki page or controls actuators according to these commands. A

†1 鹿児島大学
Kagoshima University

program may be embedded in the series of commands. The program is interpreted by the data processor. The data processor can read data from sensors or wiki pages and it can process such data. The data processor also can write the processed data to a wiki page or can control actuators. The data on the wiki page, which is acquired by a mobile terminal, can be processed by a program with PJC at the PukiWiki site and can be transformed into another data format such as a graph.

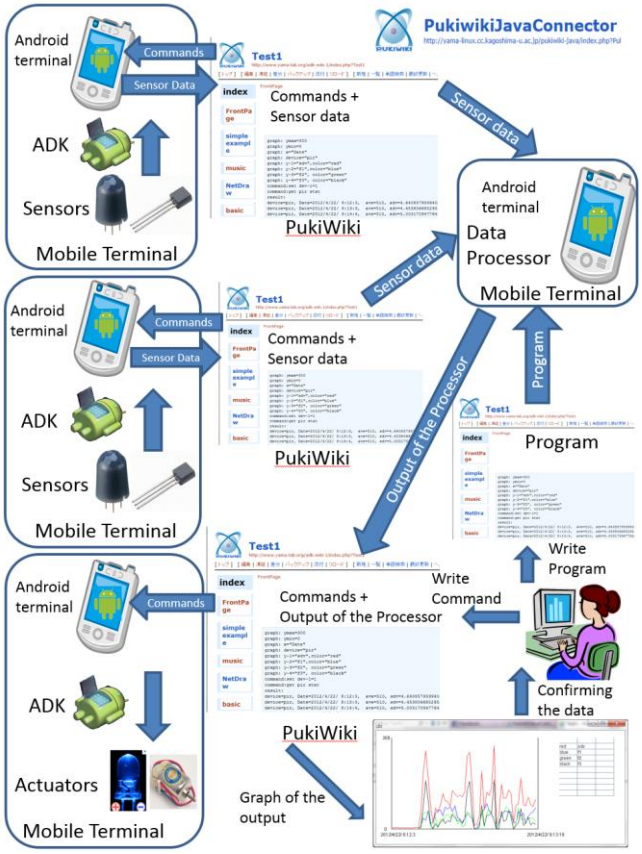


Figure 1. Outline of this system

3. Mobile Terminal

Figure 2 shows the picture of a mobile terminal. Figure 3 shows the structure of the mobile terminal. The terminal consists of an Arduino board with sensors and actuators, and an Android terminal. The Arduino board and the Android terminal are connected with a USB cable.

In the Android terminal, the application program of the system is running. The application program consists of components such as AdkService, PukiwikiJava-ConnectorService (PJC-S), AdkThread, AdkWikiActivity and Basic. The AdkService is the central component of the mobile terminal and this is in charge of controlling other components. The AdkWikiActivity is in charge of GUI of the terminal. The PukiwikiJavaConnectorService is in charge of communicating with wiki sites. The Basic is the data processor of the sensor network system. Packages such as android.jar, usb.jar, and maps.jar are used for the USB communication.

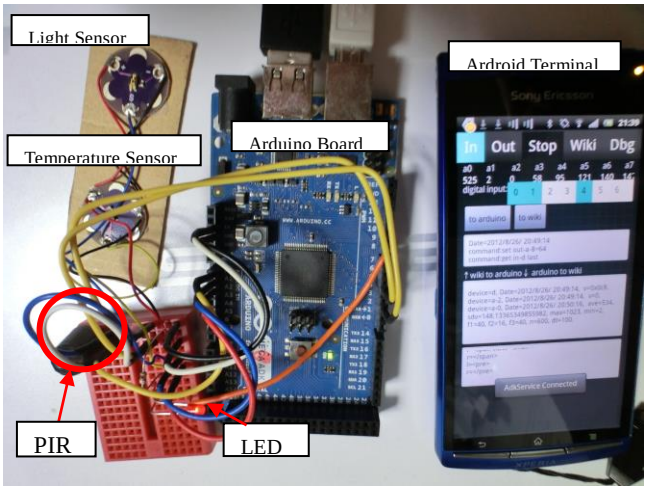


Figure 2. Mobile Terminal

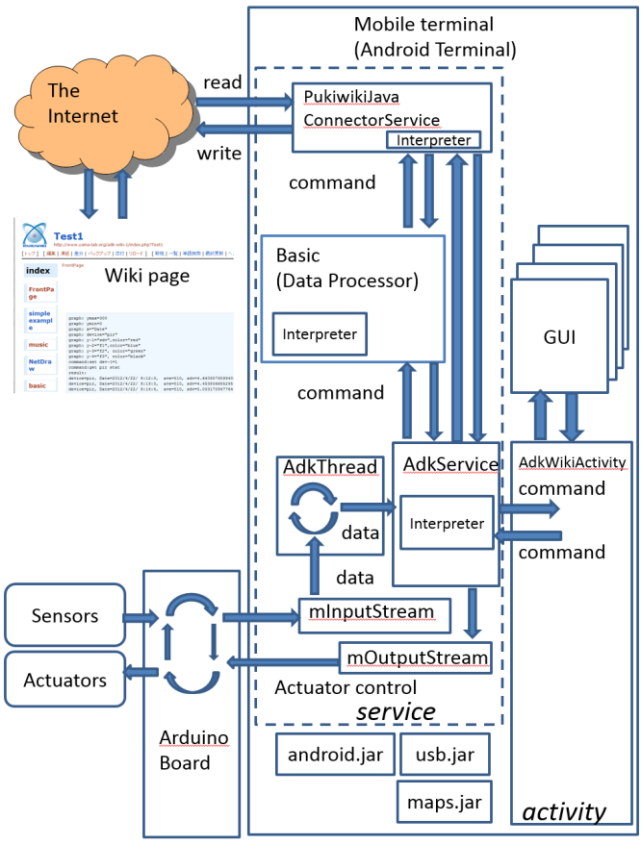


Figure 3. Structure of the Mobile Terminal

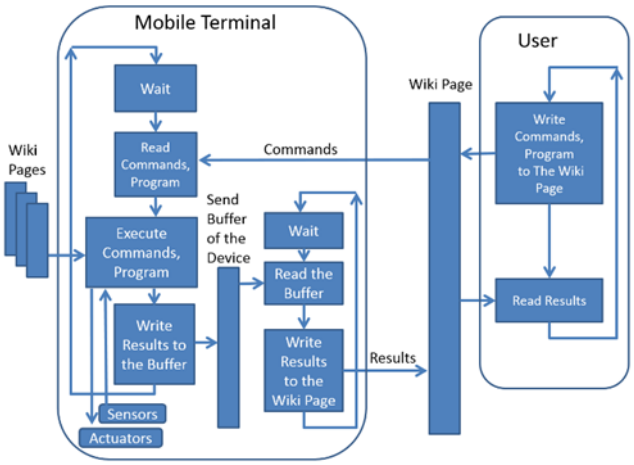


Figure 4. Behavior of the mobile terminal

Figure 4 shows behavior of the mobile terminal. The mobile terminal repeats to read the source text of the wiki page whose URL was previously given by the GUI or another command, and looks for data for communicating between the mobile terminal and the wiki site, which includes a series of commands for the mobile terminal by parsing the page. The series of commands may include a program. Some results of commands and some results of the program are written on the send buffer. The mobile terminal also repeats to write the contents of the send buffer to the wiki page.

4. Wiki page of the Sensor Network System

The series of commands and its result data are in the pre-formatted text of the wiki page. The series of commands for the mobile terminal are extracted from the page and executed. In this sensor network system, a command for a mobile terminal is represented by the following line.

command: <command>

A line of a program is represented by the following line.

program: <program line>

The series of commands may have comments. A line which has the letter "#" at its head shows a comment. The data, which is acquired from sensors or results from the program is in the wiki page after the line of "result:".

[Edit | Freeze | Diff | Backup | Upload | Reload] [New | List of

wiki-basic-ex1

```
# example.
command: set readInterval=300000
command: set sendInterval=600000
command: set out-a-8=127
command: get in-a-1 last
command: program ex
program:   in0=ex("service","get in-a-2")
program:   in2=stoi(in0)
program:   ex("service","println in-a-2="+in2)
program:   if in2>150 then ex("service","set out-a-8=50")
program:   if in2<80 then ex("service","set out-a-8=100")
command: end ex
command: run ex
command: set pageName="wiki-basic-ex2"
result:
device=a-1, Date=2013/5/21/ 17:38:28,  v=0.
device=a-1, Date=2013/5/21/ 17:43:20,  v=0.
device=a-1, Date=2013/5/21/ 17:48:20,  v=0.
device=a-1, Date=2013/5/21/ 17:53:20,  v=0.
device=a-1, Date=2013/5/21/ 17:58:20,  v=0.
```

Figure 5. Example of the wiki page which has the series of commands and saved results

a. Example

Figure 5 shows an example of the wiki page which has the series of commands and saved results. In this figure, the command "set readInterval=300000" directs the mobile terminal to repeat to read the page of the URL at 300000 milliseconds intervals. The command "set sendInterval=600000" directs the mobile terminal to repeat to write the contents of the send buffer to the page of the URL at 6000000 milliseconds intervals. The command "set out-a-8=127" directs the mobile terminal to output the analog value 127 to the output port 8 of the Arduino board. The command "get in-a-1 last" directs the mobile terminal to input the latest analog value at the analog input port 1 of the Arduino board. The command "program ex" and the command "end ex" shows that there is the program between the two commands and the name of the program is "ex". The command "run ex" runs the program ex. The command "set pageName="wiki-basic-ex2"" directs the mobile terminal to read the wiki page from the page of "wiki-basic-ex2" of the current wiki site next time.

The program of this figure inputs the analog value at the analog input port 2, outputs the value to the message area of the GUI of the mobile terminal. If the value is greater than 150, the program outputs the value 50 to the output port 8 of the Arduino board. If the value is less

than 80, the program outputs the value 100 to the output port 8 of the Arduino board.

The lines after the "result:" show the series of sensor values by the "get in-a-1 last" command. Each line consists of the device name, date and time, and the value.

b. Embeded Functions

A program of the data processor can use following embedded functions.

- *ex([object], [command])*

This function sends the *[command]* in a string to the name of the *[object]*. Currently objects are "service" for the AdkService and "connector" for the PukiwikiJava-ConnectorService. This function may have a return value.

- *getResultPart([page])*

This function extracts the result part of the string *[page]*. It is assumed that the page is in the format of the PukiWiki page of the sensor network system which includes a sequence of commands and result.

- *parseCsv([csv], [dataTable],[rowLabel],[columnLabel])*

This function translates the string *[csv]* into the two dimensional array *[dataTable]*. It is assumed that the *[csv]* is lines of equations separated by a comma as following.

$[col\text{-}label_{-1}]=[val_{-1\text{-}1}],\ldots,[col\text{-}label_{-1\text{-}1n}]=[val_{-1\text{-}1n}]$.
$[col\text{-}label_{-2}]=[val_{-2\text{-}1}],\ldots,[col\text{-}label_{-2\text{-}2n}]=[val_{-2\text{-}2n}]$.
…
$[col\text{-}label_{-m}]=[val_{-1\text{-}m}],\ \ldots,\ [col\text{-}label_{-m\text{-}mn}]=[val_{-m\text{-}mn}]$.

Each line should not have the same number of equations. Following is an example of the *[csv]*.

device=d, Date=2013/5/5/ 17:6:18, v=0x0c0.
device=a-2, Date=2013/5/5/ 17:6:18, v=155.
device=a-1, Date=2013/5/5/ 17:6:18, v=53.
device=a-0, Date=2013/5/5/ 17:6:45, ave=242, ..., dt=100.
device=a-0, Date=2013/5/5/ 17:7:53, ave=242, …, dt=100.
…

[rowLabel] is the hash table which has key-value pairs of ("rowcol", "row") and ("maxIndex", maximum row index of the table). The ("rowcol","row") shows that this hash table includes row information. *[columnLabel]* is the hash table which has key-value pairs of ("rowcol","col"), ("maxIndex", maximum column index of the table), ($[col\text{-}label_{-1}]$, column index of the label), …, ($[col\text{-}label_{-max}]$, column index of the label whose index is the maximum column value in the table). The ("rowcol","col") shows that this hash table includes column information.

- *sumif([dataTable], [hash table of row or column], [index-1], [operator], [operand], [index-2])*

This function sums values at $[index_{-2}]$ of rows or columns if the condition, which is represented by the $[index_{-1}]$, *[operator]* and *[operand]*, is satisfied. If the *[hash table of row or column]* is the hash table of *[rowLabel]*, the value of the *sumif* will be the following.

$$\sum_{i=0}^{rmax-1} \left\{ \begin{matrix} dataTable[i, index_2], \\ if\ operator(dataTabe[i, index_1], operand) \end{matrix} \right\}$$

In this expression, *rmax* is the maximum index of the row. If the *[hash table of row or column]* is the hash table of *[colLabel*], the value of the *sumif* will be the following.

$$\sum_{j=0}^{cmax-1} \left\{ \begin{matrix} dataTable[index_2, j], \\ if\ operator(dataTabe[index_1, j], operand) \end{matrix} \right\}$$

In this expression, *cmax* is the maximum index of the column.

- *countif([dataTable], [hash table of row or column], [index-1], [operator],[operand])*

This function counts the number of row or column if the condition, which is represented by the $[index_{-1}]$, *[operator]* and *[operand]*, is satisfied. If the *[hash table of row or column]* is the hash table of *[rowLabel]*, the value of the *countif* will be the following.

$$\sum_{i=0}^{rmax-1} \left\{ \begin{matrix} 1, \\ if\ operator(dataTabe[i, index_1], operand) \end{matrix} \right\}$$

If the *[hash table of row or column]* is the hash table of *[colLabel]*, the value of the *countif* will be the following.

$$\sum_{j=0}^{cmax-1} \left\{ \begin{matrix} 1, \\ if\ operator(dataTabe[index_1, j], operand) \end{matrix} \right\}$$

5. Usage Example

We are developing a remote room sensor system that uses this sensor network system. We successfully acquired room data at a remote place for each hour of a

day and each day of a month, and showed these data on wiki pages using the sensor system. The room data is shown as a graph by the PukiWiki plug-in using the PukiWiki Java Connector [4]. We also successfully controlled LED lighting by the collected data from sensors.

Figure 6 shows the outline of the remote room sensor system. This system consists of a mobile terminal for acquiring sensor data at the remote room (MT-1), 24 wiki pages for saving the sensor data of each hour of a day (H-WIKIs), another mobile terminal for totaling up these data on 24 wiki pages (MT-2), and 31 wiki pages for totaling up and saving the data of each day of a month (D-WIKIs). The MT-1 repeats to read the page of H-WIKI which is corresponding to the current hour, and interprets commands on the page. Every page of H-WIKI has the series of commands which acquires data from sensors of the terminal and writes the data to the wiki page after the series of commands. MT-2 repeats to read the page of D-WIKI which is corresponding to the current day, and interprets commands and the program on the page. Every wiki page of D-WIKI has the program which reads 24 pages of H-WIKI, calculates average values for each hour, arranges them according to hours of the day, and writes the arranged data of the day to the page of H-WIKI after the series of commands and the program. The program also controls brightness of LEDs of the terminal.

Figure 7 shows a part of the page of H-WIKI and the graph of sensors data of the page. In this figure, the line starting with “graph:” indicates the command line for controlling the graph[6]. “set out-a-8=0” sets value 0 for the output device labeled a8. The device a8 is the LED in this example, “get in-d last” gets the latest value of the digital input port, d0,d1,…,d7. “get in-a-2 last” gets the latest value of the analog input port a2, which is connected to the temperature sensor. “get in-a-1 last” gets the latest value of the analog input port a1, which is connected to the light sensor. “get in-a-0 stat” gets the statistics value of analog input port a0. This command acquires a series of PIR values, which are sampled approximately ten times per one second for one minute. Then, the command calculates the average, the standard deviation, the maximum value, the minimum value of the series of values, and the frequencies of a range of values of the series. These values are then sent to the wiki page. “set pageName=”pir-1-<hour>”” sets the next page name of the wiki site to “pir-1-<hour>”. <hour> is replaced by the current hour when this command is executed. This command line realizes page traversing between wiki pages according to the hour.

The “result:” line means that results are written after this line. A line of the result consists of elements separated by a comma. One element is an equation, whose left-hand side is the label of a value and whose right-hand side is the value. Moreover, “device=<device-name>” shows the device name of the data of this line. “Date=<date>” shows the acquired date, “ave=<value>” shows the average of the series of values, “v=<value>” shows the (latest) value of the device, “sdv=<value>” shows the standard derivation of the series of values, “max=<value>” shows the maximum value of the series. “min=<value>” shows the minimum value of the series, “f1=<value>” shows the frequency of values whose range is between 40 to 80 plus the average, “f2=<value>” shows the frequency of values whose ranges are between 80 to 160 plus the average, and “f3=<value>” shows the frequency of values whose ranges are more than 160 plus the average. The value of the device “d” shows the digital values of d7, d6, …, d0 in a hexadecimal value. The graph in this figure is shown by a plug-in of the PukiWiki site. The plug-in is developed by using the Pukiwiki-Java Connector.

Figure 8 shows a part of a page of D-WIKI. The program of this page computes average values of sensors of every hour and writes them after the line of “result:” of this page. This program also controls LEDs of the mobile terminal if some conditions were satisfied. In the Figure 8, page=ex(“connector”, “getpage “+url+i) gets the web page which has the sensor data of the hour of i. The URL, which is the concatenation of url and i, represents the wiki page. y0=sumif(dataTable,rowLabel, columnLabel ("device "), "=", "a-0", columnLabel("sdv")) sums the values whose label is “sdv” in the CSV if the device of the sensor was “a-0”. c0=countif(dataTable, rowLabel, columnLabel ("device"), "=", "a-0") counts the lines of the device “a-0”. ex(“service”, “putSendBuffer “+dataline) adds the dataline to the send buffer of Figure 5 and ex(“service”, “sendResults.”) writes the contents in the send buffer to the current page. ex(“service”, “set out-8-<val>”) sets the value <val> to the actuator of “out-8”. set pageName =” daily-1-<day>” changes the URL of the web page for next time reading. The <day> shows the current day.

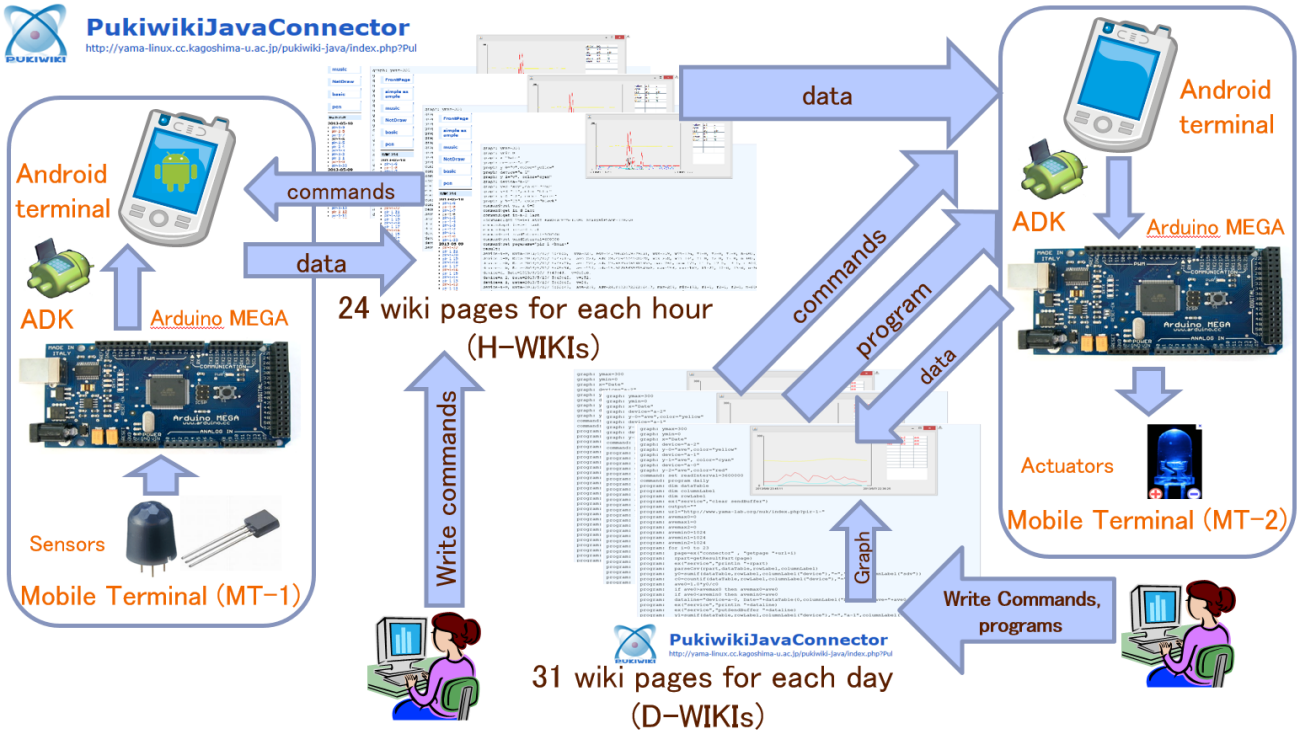


Figure 6. Outline of the remote room sensor system.

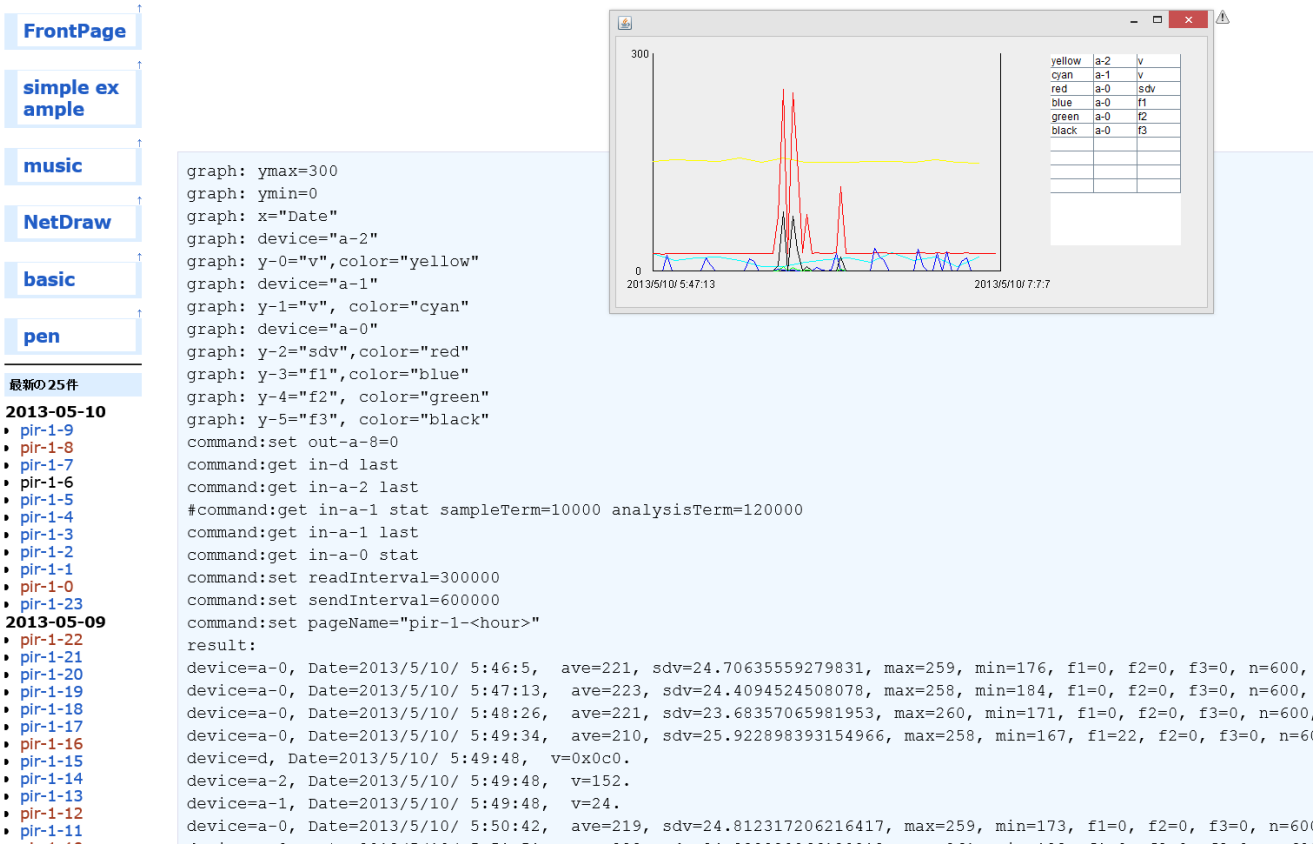
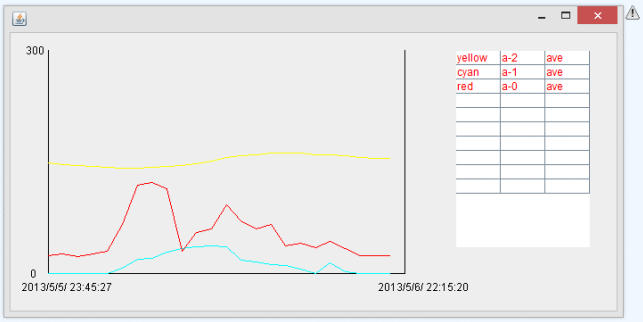


Figure 7. A part of a H-WIKI page

```
graph: ymax=300
graph: ymin=0
graph: x="Date"
graph: device="a-2"
graph: y-0="ave",color="yellow"
graph: device="a-1"
graph: y-1="ave", color="cyan"
graph: device="a-0"
graph: y-2="ave",color="red"
command: set readInterval=3600000
command: program daily
program: dim dataTable
program: dim columnLabel
program: dim rowLabel
program: ex("service","clear sendBuffer")
program: output=""
program: url="http://www.[redacted]/index.php?pir-1-"
program: avemax0=0
program: avemax1=0
program: avemax2=0
program: avemin0=1024
program: avemin1=1024
program: avemin2=1024
program: for i=0 to 23
program:   page=ex("connector" , "getpage "+url+i)
program:   rpart=getResultPart(page)
program:   ex("service","println "+rpart)
program:   parseCsv(rpart,dataTable,rowLabel,columnLabel)
program:   y0=sumif(dataTable,rowLabel,columnLabel("device"),"=","a-0",columnLabel("sdv"))
program:   c0=countif(dataTable,rowLabel,columnLabel("device"),"=","a-0")
program:   ave0=1.0*y0/c0
program:   if ave0>avemax0 then avemax0=ave0
program:   if ave0<avemin0 then avemin0=ave0
program:   dataline="device=a-0, Date="+dataTable(0,columnLabel("Date"))+",ave="+ave0
program:   ex("service","println "+dataline)
program:   ex("service","putSendBuffer "+dataline)
program:   y1=sumif(dataTable,rowLabel,columnLabel("device"),"=","a-1",columnLabel("v"))
program:   c1=countif(dataTable,rowLabel,columnLabel("device"),"=","a-1")
```




```
program: next i
program: dataline="device=maxvalues, Data="+dataTable(0,columnLabel("Date"))
program: dataline=dataline+",avemax0="+avemax0+",avemax1="+avemax1+",avemax2="+avemax2
program: ex("service","println "+dataline)
program: ex("service","putSendBuffer "+dataline)
program: ex("service","sendResults.")
program: if avemax0 < 7.0 then ex("service","set out-a-8=255") else ex("service","set out-d-8=0")
program: if avemax1 < 100.0 then ex("service","set out-a-9=255") else ex("service","set out-d-9=0")
program: if avemax2 > 160.0 then ex("service","set out-a-10=255") else ex("service","set out-d-10=0")
command: end daily
command: set pageName="daily-1-<day>"
command: set result=""
command: run daily
result:
device=a-0, Date=2013/5/6/ 23:45:11,ave=25.07021401548858
device=a-1, Date=2013/5/6/ 23:45:11,ave=0.0625
device=a-2, Date=2013/5/6/ 23:45:11,ave=152.8125
device=a-0, Date=2013/5/7/ 0:44:7,ave=25.00743801302449
device=a-1, Date=2013/5/7/ 0:44:7,ave=0.3125
device=a-2, Date=2013/5/7/ 0:44:7,ave=151.6875
```

Figure 8. A part of a D-WIKI page

6. Comparisons with Related work

A. *IEEE 1888*

IEEE 1888[12] is a HTTP-based SOAP/XML over IP communication protocol among facilities, databases and information system for facility information management and control. Both of our system and IEEE 1888 use web servers for data storage. A mobile terminal of our sensor network system corresponding to the combination of the GW and the APP of IEEE 1888 system architecture. A Pukiwiki site of our sensor network system is corresponding to the Storage of IEEE 1888 system architecture. Our sensor network system does not have the Registry of IEEE 1888 system architecture. The program of a data processor of our sensor network system is written on the wiki page of a wiki site whereas IEEE 1888 does not define where is the program of an APP.

B. Xively

Xively[14] is a real-time open data web service for the Internet of Things. As one of the most popular sensor data sharing services. Xively has open APIs for uploading and manipulating data, and so it is easy to enable a sensor device to upload data to Xively and also easy to write a program for processing the uploaded data. Many devices and applications can read and write data on Xively, and many users are currently using Pachube.

Our sensor network system has functions similar to those of Xively. However, the APIs of Xively are used for the Xively site only. In contrast, our system can be used for any PukiWiki site, not only a specific site. Our sensor network system can also be used for building one's own sensor/actuator network.

C. Scripting Layer for Android (SL4A)

Scripting Layer for Android (SL4A)[13] brings popular scripting languages to Android. We uses our original language processor for the data processor now. However the original language is not familiar with potential users. We should use SL4A instead of using our original language processor for this potential users of this sensor network system.

D. Broadcast

Broadcast[11] is an embedded web application for remote Android device management. Broadcast uses the SL4A. However the Broadcast does not have the function for data exchanging between web pages.

E. Message Oriented Middleware (MOM)

Message-oriented middleware (MOM)[1] is software or hardware infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocols. This data processor can be viewed as a MOM.

7. Concluding Remarks

We have shown that a wiki site can be used for not only people-to-people collaboration but also people-to-machine, machine-to-people, and machine-to-machine collaboration using the proposed our sensor network system. We successfully collected remote room data and uploaded the data to a wiki page using the sensor network system. We also successfully controlled a device at a remote place that writes a command on a wiki page using the sensor network system. We are improving this system to be more general purpose and to be easier to use. Reliability and security of this sensor network system were not discussed in this paper. We will show them in later reports. We welcome the help of others who would like one to participate in improving and making his or her own sensor/actuator network.

References

1) Sushant Goel, Hema Sharda, David Taniar: Message-Oriented-Middleware in a Distributed Environment, proceeding of: Innovative Internet Community Systems, Third International Workshop, IICS 2003, Leipzig, Germany, June 19-21, 2003.
2) Takashi Yamanoue: A Draw Plug-in for a Wiki Software, saint, 10th IEEE/IPSJ International Symposium on Applications and the Internet, pp.229-232, 2010.
3) Takashi Yamanoue, Kentaro Oda, Koichi Shimozono: PukiWiki-Java Connector, a Simple API for Saving Data of Java Programs on a Wiki, ACM WikiSym '11, Proceedings of the 2011 international symposium on Wikis, Mountain View, CA, USA, 3-5 Oct., 2011.
4) Takashi Yamanoue, Kentaro Oda, Koichi Shimozono: A Simple Application Program Interface for Saving Java Program Data on a Wiki , Advances in Software Engineering, Hindawi Publishing Corporation, 2012 .
5) Takashi Yamanoue, Kentaro Oda and Koichi Shimozono: A Casual Network Security Using a Portable Sensor Device and Wiki Software, 12th IEEE/IPSJ International Symposium on Applications and the Internet – C3NET workshop, 2012.
6) Takashi Yamanoue, Kentaro Oda and Koichi Shimozono: A M2M system using Arduino, Android and Wiki Software, Proceedings of the 3rd IIAI International Conference on e-Services and Knowledge Managerment (IIAI ESKM 2012), pp.123-128, Fukuoka, Japan, 20-22 Sep. 2012.
7) Takashi Yamanoue, Kentaro Oda, Koichi Shimozono: A Malicious Bot Capturing System using a Beneficial Bot and Wiki, Journal of Information Processing(JIP), vol.21, No.2, pp.237-245, 2013.
8) Android: http://www.android.com/ as of May 23, 2013.
9) Android Open Accessory Development Kit: http://developer.android.com/guide/topics/usb/adk.html as of May 23, 2013.
10) Arduino: http://www.Arduino.cc/ as of May 23, 2013.
11) Broadcast: https://github.com/mleone/broadcast/ as of May 7, 2013.
12) IEEE 1888: http://standardsinsight.com/engineering_standard_development/ieee1888 as of May 23, 2013.
13) SL4A: http://code.google.com/p/android-scripting/ as of May 7, 2013.
14) Xively: https://xively.com/ as of May 23, 2013.
15) PukiWiki: http://pukiwiki.sourceforge.jp/ as of as of May 23, 2013.
16) Wikipedia: http://www.wikipedia.org/ as of May 23, 2013.
17) Ward Cunningham: Wiki Wiki Web, http://c2.com/cgi/wiki?WikiWikiWeb as of May 23, 2013.