Electronic Preprint for Journal of Information Processing Vol.28

Regular Paper

Monitoring of Servers and Server Rooms by IoT System that Can Configure and Control its Terminal Sensors Behind a NAT Using a Wiki Page on the Internet

Takashi Yamanoue[1,a)]

Received: June 17, 2019, Accepted: November 29, 2019

Abstract: This paper describes a method of monitoring servers or server rooms by an Internet of Things (IoT) system that can configure and control terminal sensors behind a network address translation (NAT) router through a Wiki page on the Internet. This IoT system consists of Wiki pages and a bot (Wiki Bot) that runs on Raspberry Pi with sensors. A Wiki Bot can be placed behind the NAT router to resist various online attacks. The IoT system can monitor servers behind a NAT router over the Internet. A Wiki Bot is controlled by sending commands from the Wiki page. It acquires data from its sensors and processes the data via a command sequence of commands. The sensors settings and the data sampling rate can be remotely changed by changing the commands on the Wiki page.

Keywords: monitoring, network management, Wiki, Bot, IoT

1. Introduction

Information and communication technology (ICT) infrastructure administrators at universities monitor servers and server rooms on campus [1], [2]. Such monitoring is deceptively difficult. If the administrators fail to identify a potential problem or if the staff cannot solve an identified problem, users will not be able to use the network. Some server room problems can even lead to fire.

ICT infrastructure administrators thus use various methods to identify problems with a server or the server room.

In this paper, we describe the monitoring of servers and server rooms using bots, which are remote-controlled computers or programs. Bots are often malicious programs that form a botnet [3], but they can also be used for beneficial tasks [4], [5], [6], [7].

We use a bot to monitor a Web server on our campus. This bot uses Twitter to periodically update the server status. We also use bots to monitor a server room on our campus. These bots notify administrators of changes in room temperature, room brightness and room air pollution.

The bots are part of an Internet of Things (IoT) system that connects them over the Internet.

2. Outline of Wiki IoT System and the Wiki Bot

The bots, called Wiki Bots, are Raspberry Pis that run the bot software. Some of them are equipped with sensors and some are equipped with a wireless sensor network (WSN) transmitter. Wiki Bots are controlled by commands and programs on Wiki pages on web servers. Wiki Bots equipped with a WSN transmitter are gateways to the WSN. This IoT system consists of IoT devices (bots) that communicate with each other and Wiki software on the Internet. We call this IoT system the *Wiki IoT system* (**Fig. 1**).

Administrators can control bots in a local area network (LAN) protected by network address translation (NAT) routers from outside the LAN by writing commands and programs on Wiki pages hosted on web servers located outside the LAN or that can be accessed from outside the LAN. A WSN is not considered in this paper. A Wiki IoT system with a WSN is described in other papers [8], [9].

We adopt PukiWiki software [10] for the Wiki IoT because PukiWiki is simple to deploy than many other wikis. PukiWiki needs minimized requirements, even requires no data base engines such like MySQL. Extracting PukiWiki tar-ball to a web server root directory, it just works [11].

When a server is connected to a campus LAN that cannot be accessed from the Internet directly, it is impossible to directly monitor the server from outside of campus. However, such monitoring would be convenient when administrators are not on campus. A Wiki Bot can help administrators monitor the server in such cases. A server room is usually physically isolated from the outside; there are no people in the server room. A Wiki Bot can help the administrators monitor server rooms. A bot can help administrators monitor servers that are maintained by a third-party company and onto which monitoring software cannot be installed.

2.1 Behavior of a Wiki Bot

Figure 2 shows the behavior of a Wiki Bot. A Wiki Bot repeats the following steps:

1 Fukuyama University, Fukuyama, Hiroshima 729–0292, Japan
a) yamanoue@fukuyama-u.ac.jp

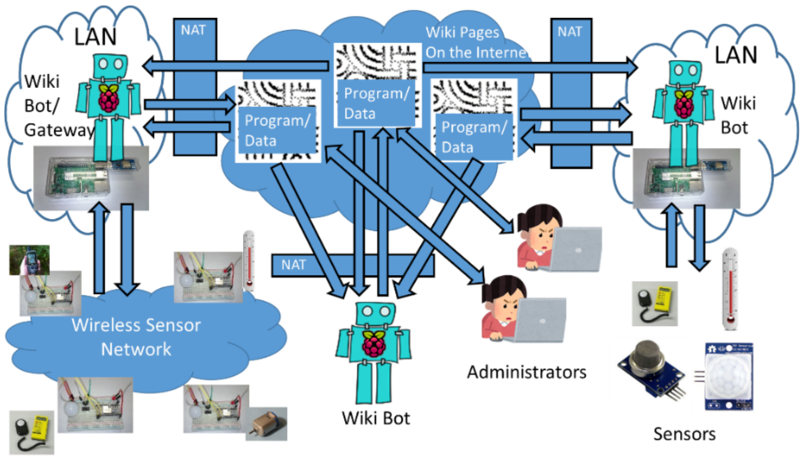


Fig. 1 Outline of a Wiki IoT system.

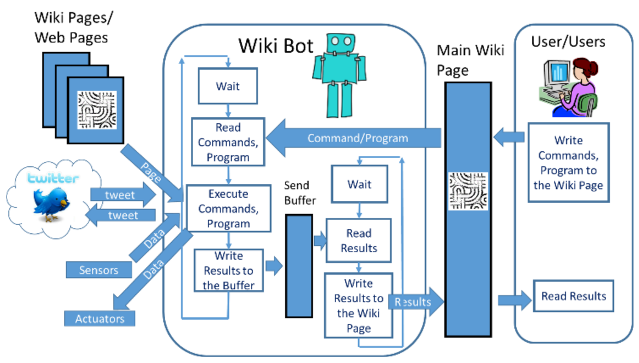


Fig. 2 Outline of behavior a Wiki Bot.

1) Wait for a specified time.
2) Read commands from the specific Wiki page assigned to the bot. The source code of a program and the command for running the program can be used as commands
3) Execute these commands.
4) Data in the send buffer are written back to the Wiki page that contains the commands, after the line “result:”.

If the source code of a program is embedded in the series of commands, then the program is transferred to the language processor of the bot. The program is translated into an internal representation and run by the interpreter when the run command is executed.

The program can read other Wiki pages and Web pages. It can read and send tweets on Twitter. Bots with sensors, can also read sensor data. Bots with actuators can send data to the actuators. A bot can write data to the send buffer. If the data are spilt from the send buffer, old data are deleted. These functions are realized using the embedded functions of the programming language.

We call the specific Wiki page that contains the commands, the program, and data which are written back from the Wiki Bot, the *main Wiki page*.

Actuators are not considered in this paper. A Wiki IoT system with actuators is described in another paper [12].

2.2 Commands and the Program of a Wiki Bot

Figure 3 shows an example of a program with a series of commands. In this example, lines that start with “*command:*” are the commands. The first line,

command: set readInterval = 60000

tells the Wiki Bot to read the page at the given URL every minute. The time interval is given in milliseconds.

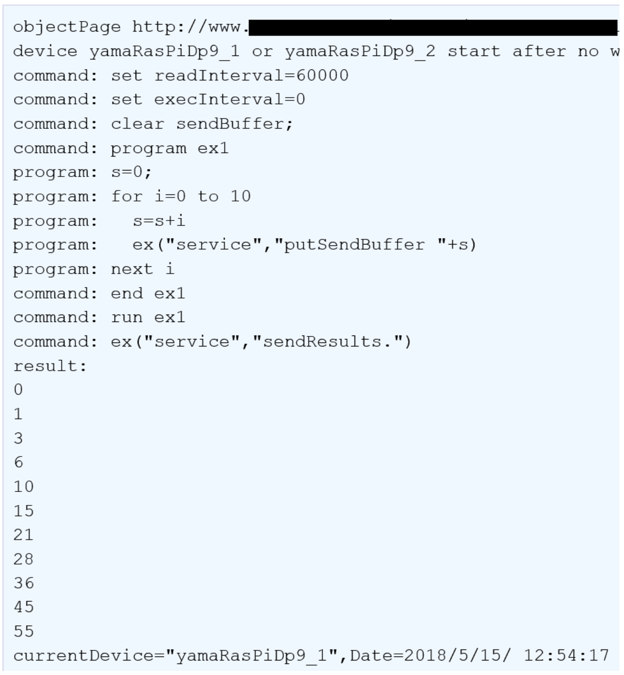

```
objectPage http://www.[redacted]
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no w
command: set readInterval=60000
command: set execInterval=0
command: clear sendBuffer;
command: program ex1
program: s=0;
program: for i=0 to 10
program:   s=s+i
program:   ex("service","putSendBuffer "+s)
program: next i
command: end ex1
command: run ex1
command: ex("service","sendResults.")
result:
0
1
3
6
10
15
21
28
36
45
55
currentDevice="yamaRasPiDp9_1",Date=2018/5/15/ 12:54:17
```

Fig. 3 An example of the program of a Wiki Bot and its output after the execution of the program.

The lines that start with “*program:*” are the program. A program is enclosed by commands “*command: program <name>*” and “*command: end <name>*”, where *<name>* is the name of the program. In this example, the program is named “ex”. The last command line, “*command: run ex*” translates the program into its internal representation and executes.

A Wiki page for a Wiki Bot can also contain the “*set pageName*” command and the “*include*” command.

When the “*set pageName*” command is interpreted in the bot, the Wiki Bot will use the Wiki page designated by this command as the main Wiki page. The name of the designated Wiki page can include the current time or date. For example, when the following command is interpreted at eight o'clock, the Wiki Bot uses the page "pir-1-8", on the same server used for the current Wiki page, as the main Wiki page.

set pageName = "pir-1-<hour>"

When the "*include*" command is interpreted in a Wiki Bot, the bot inserts the Wiki page designated by the include command into the place of the include command of the original Wiki page. This command is useful when there are identical commands or programs on many Wiki pages. It can also be used for object-oriented programming.

The commands and the program on the main Wiki page can be modified to change the behavior of the bot without stopping the bot.

A Wiki Bot can be connected to a LAN protected by a NAT or network address port translation router. The bot can be controlled from outside the LAN.

2.3 Embedded functions of a program

The program of a Wiki Bot can use the following embedded functions.

- *ex(<object>, <command>)*

This function sends the *<command>* in a string to *<object>*. Currently, objects are a "service" for interacting with the bot's functions, a "connector" for interacting with a web page and a "pi4j" for interacting with sensors and actuators connected to the Wiki Bot. This function can have a return value. The following is an example of the statement that reads the page http://www.page.ex/ and assigns the page to the variable *page* as a string value.

page = ex("connector", "getpage http://www.page.ex/")

- *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 a Wiki Bot, which includes a sequence of commands, a program, and the result part.

- *parseCsv(<csv>, <dataTable>, <rowLabel>, <columnLabel>)>)*

This function transforms the string *<csv>* into a two-dimensional array *<dataTable>*. It is assumed that *<csv>* is in the following format.

<col-label-1>=<val-1-1>,...,<col-label-1-1n>=<val-1-1n>.

<col-label-2>=<val-2-1>,...,<col-label-2-2n>=<val-2-2n>.

...

<col-label-m>=<val-1-m>,...,

<col-label-m-mn>=<val-m-mn>.

Each line should not have the same number of values.

The following is an example of *<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 a hash table with key-value pairs ("rowcol", "row") and ("maxIndex", maximum row index of the table). This hash table includes row information such as the maximum index of the row. *<columnLabel>* is a hash table with key-value pairs ("rowcol","col"), ("maxIndex", maximum column index of the table), (*<col-label$_{-1}$>*, column index of the label), ..., (*<col-label$_{-max}$>*, column index of the label whose index is the maximum column value in the table).

There are functions for accessing a table such as *sumif* and *countif,* these functions are equivalent to those in Microsoft Excel. There is also a function called *getindex* that finds the minimum index of the table that satisfies the specified condition which is the argument of the *getindex* function.

2.4 Class Pages and Object Pages

Wiki IoT is an object-oriented computing system [9]. In our Wiki IoT system, an object is the combination of a Wiki page and a Wiki bot.

Some bots in a Wiki IoT system may use the same commands. To reduce duplication on Wiki pages, the Wiki IoT system might have a *class* page for sharing common commands among the Wiki pages of such objects. We call a main Wiki page of objects an *object* page. An object page uses the "*include*" command for a class page when sharing a common class among object pages.

If class pages have common commands, they can share another page of the same class page using the "*include*" command, similar to inheritance in object-oriented programming.

The override function in object-oriented programming is also realized by the "*include*" command. If Wiki page B includes Wiki page A, then the program on Wiki page A becomes the super-class of the program on Wiki page B, which is the sub-class. The programming language for our Wiki IoT system is similar to BASIC. A program is translated into an S-expression, which is evaluated by a LISP interpreter. If functions with the same name exist in the super-class program and the sub-class program, the func-

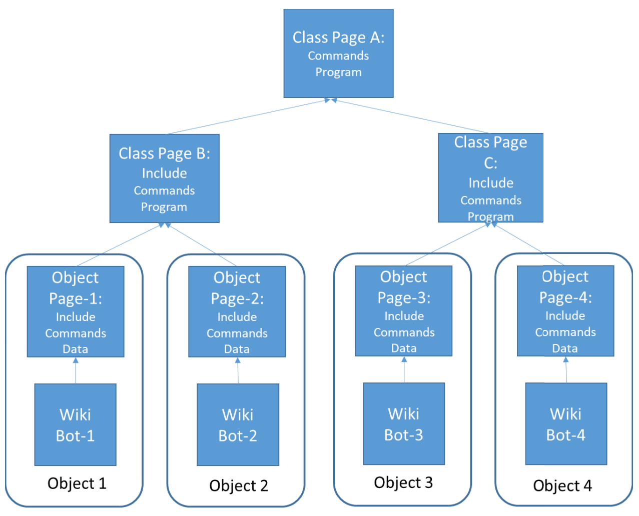
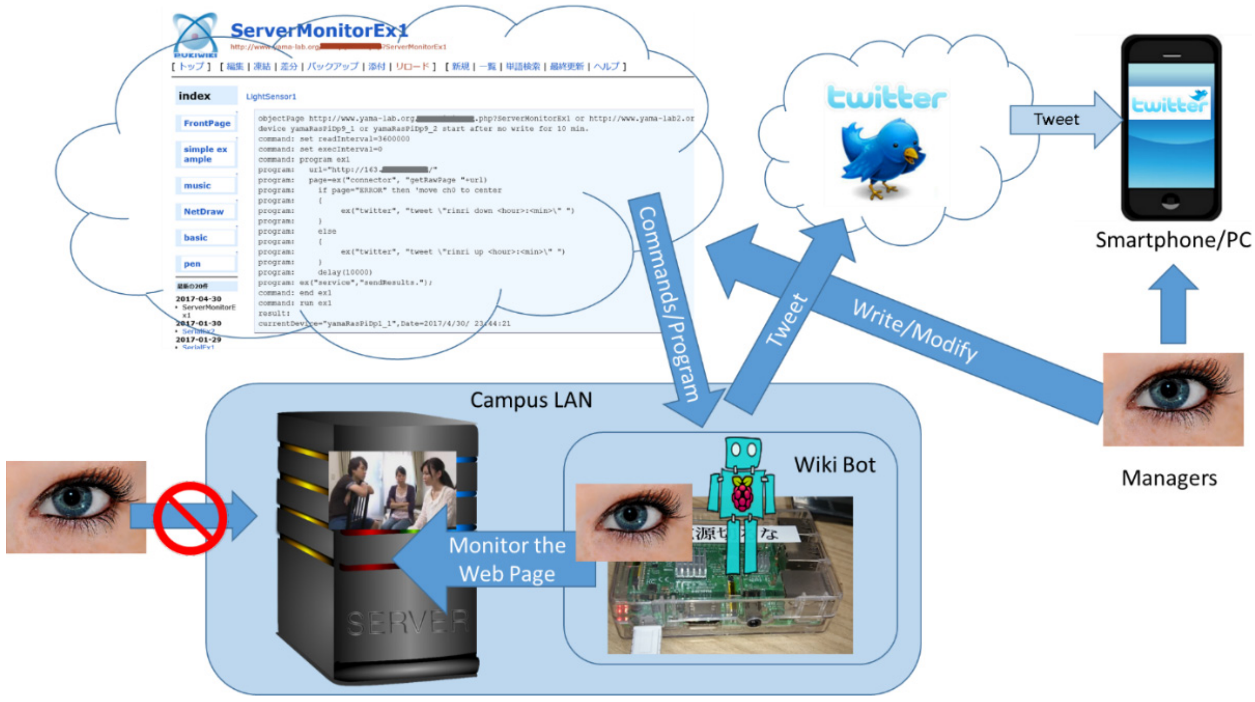


Fig. 4 Example of class hierarchy of Wiki IoT.


ServerMonitorEx1
http://www.yama-lab.org[redacted]?ServerMonitorEx1
[トップ] [編集 | 凍結 | 差分 | バックアップ | 添付 | リロード] [新規 | 一覧 | 単語検索 | 最終更新 | ヘルプ]
index
FrontPage
simple example
music
NetDraw
basic
pen
最新の20件
2017-04-30
ServerMonitorEx1
2017-01-30
SerialEx2
2017-01-29
SerialEx1
LightSensor1

```
objectPage http://www.yama-lab.org[redacted].php?ServerMonitorEx1 or http://www.yama-lab2.or
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no write for 10 min.
command: set readInterval=3600000
command: set execInterval=0
command: program ex1
program:   url="http://163.[redacted]/"
program:   page=ex("connector", "getRawPage "+url)
program:     if page="ERROR" then 'move ch0 to center
program:     {
program:          ex("twitter", "tweet \"rinri down <hour>:<min>\" ")
program:     }
program:     else
program:     {
program:          ex("twitter", "tweet \"rinri up <hour>:<min>\" ")
program:     }
program:     delay(10000)
program: ex("service","sendResults.");
command: end ex1
command: run ex1
result:
currentDevice="yamaRasPiDp1_1",Date=2017/4/30/ 23:44:21
```

Fig. 5 Monitoring of server that can-not be directly accessed from outside the LAN.

tion of the super-class program is overwritten by the function of the sub-class program.

Figure 4 shows an example of the class hierarchy of Wiki IoT. Class Page A, Class Page B, Class Page C, Object Page-1, Object Page-2, Object Page-3, and Object Page-4 are Wiki pages. The program in Class Page A is the super-class of sub-class programs in Class Page B and Class Page C. Object Page-1 and Object Page-2 use the commands and program in Class Page B. Object-3 and Object-4 use the commands and program in Class Page C.

3. Monitoring Servers from Outside the LAN

Video clips of computer ethics are shown to students to enhance cyber security on our campus [13], [14]. The video clips are stored on a Web server on our campus. The license of the video clips does not allow us to access this Web server from outside our campus. There have been instances of this Web server going down, which prevented the video clips from being shown in class. Therefore, we have started monitoring this Web server using our Wiki IoT system.

The Web server monitoring system consists of a Wiki Bot, a Wiki page, and Twitter (**Fig. 5**). The Wiki Bot is connected to the campus LAN to monitor the Web server status (this is not possible from outside the LAN). The bot repeatedly reads the Wiki page that contains the program for monitoring the Web server and executes the program. The Wiki Bot monitors the Web server status and reports it to administrators via a tweet on Twitter.

Figure 6 shows the main Wiki page for the Wiki Bot of the Web server monitoring system. This page includes the program for the Wiki Bot. The program does the followings:

1) Tries to read the Web page of the Web server.
2) If the Wiki Bot cannot read the page, it tweets the following: *"rinri down <hour:min>"*
3) If the Wiki Bot can read the page, it tweets the following: *"rinri up <hour:min>"*

We have used the Web server monitoring system for three years. **Figure 7** shows some example tweets from the Wiki Bot of the Web server monitoring system. These tweets alerted us that the Web server was down. The Web server went down between 2:55 and 3:56 on March 18, 2017. It was a weekend so we were at home at that time. After receiving the tweets, we went to the campus and fixed the problem.

We have confirmed that our use of Twitter complies with the Twitter's rules [15].

4. Monitoring of Server Room

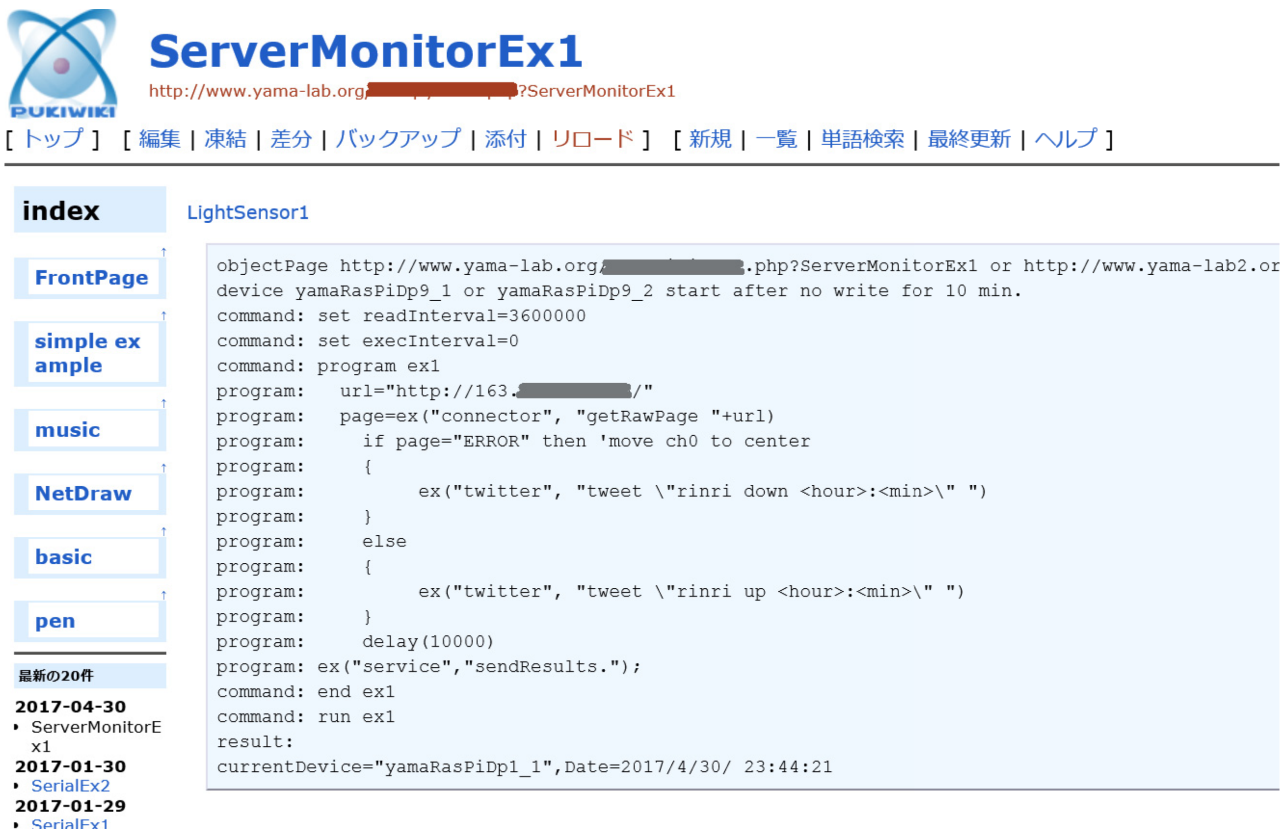
There was air conditioner malfunction in a server room on our campus in 2015. This malfunction occurred on weekend and it was not noticed until a Monday class that uses a computer laboratory. The servers in the server room also malfunctioned because of the high temperature in the room. If the problem had gone unnoticed longer, there could have been a fire in the server room.

As a result of the server malfunctions, we could not use the computer laboratories at that time because the computers use Active Directory servers and file servers in the server room.

A server room usually has no windows and the only time someone is in the server room is during server maintenance. Therefore, sensors must be used to monitor the server room conditions from the outside.

We called the air conditioner and servers maintenance companies at the time. They fixed the trouble in a few days.

We decided to start monitoring the server room. There are commercial services for remotely monitoring servers. These services usually require the installation of a monitoring program on the server. However, it is difficult to install such programs on a server that is not a property of the university. Therefore, we decided to monitor the server room using a Wiki IoT system



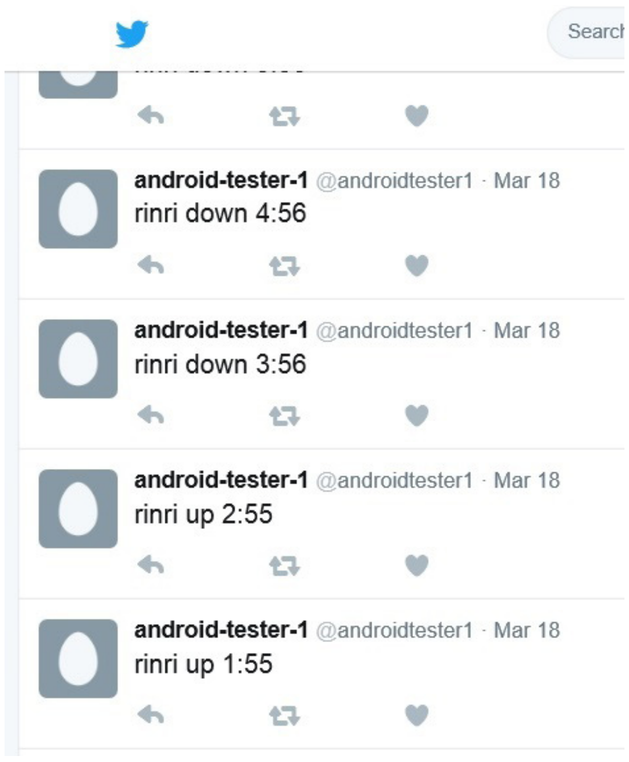
ServerMonitorEx1

http://www.yama-lab.org[redacted]?ServerMonitorEx1

LightSensor1

```
objectPage http://www.yama-lab.org[redacted].php?ServerMonitorEx1 or http://www.yama-lab2.or
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no write for 10 min.
command: set readInterval=3600000
command: set execInterval=0
command: program ex1
program:   url="http://163.[redacted]/"
program:   page=ex("connector", "getRawPage "+url)
program:     if page="ERROR" then 'move ch0 to center
program:     {
program:          ex("twitter", "tweet \"rinri down <hour>:<min>\" ")
program:     }
program:     else
program:     {
program:          ex("twitter", "tweet \"rinri up <hour>:<min>\" ")
program:     }
program:     delay(10000)
program: ex("service","sendResults.");
command: end ex1
command: run ex1
result:
currentDevice="yamaRasPiDp1_1",Date=2017/4/30/ 23:44:21
```

Fig. 6 Wiki page that controls the Wiki Bot for monitoring a Web server that can not be directly accessed from outside the LAN.



android-tester-1 @androidtester1 · Mar 18
rinri down 4:56

android-tester-1 @androidtester1 · Mar 18
rinri down 3:56

android-tester-1 @androidtester1 · Mar 18
rinri up 2:55

android-tester-1 @androidtester1 · Mar 18
rinri up 1:55

Fig. 7 Tweets of aliveness of the Web server by the Wiki IoT System.

(**Fig. 8**).

We developed *Sensor Bot*, A Sensor Bot comprises a Raspberry Pi, a temperature sensor, a light sensor, a passive infrared (PIR) motion sensor and a gas sensor. The PIR motion sensor detects human presence. The gas sensor detects the air pollution level. **Figure 9** shows a Sensor Bot. The temperature sensor and the light sensor are I2C devices. The output of the gas sensor is an analog value, which is converted to a digital value by an I2C analog-to-digital converter. The PIR motion sensor outputs a digital value, which is sent to the GPIO port of the Raspberry Pi.

The server room monitoring system consists of a Sensor Bot (Wiki Bot-1), a Wiki Bot without sensors (Wiki Bot-2), 24 Wiki pages, one for each hour in a day (H-Wikis), the class page for the bot (sensors-1-h-class), 31 Wiki pages, one for each day in a month (D-Wikis), and the class page for the bot (Daily-Class-1).

An H-Wiki contains the command for including the sensors-1-h-class page and acquires sensor data every minute for an hour. **Figure 10** shows an excerpt of the sensors-1-h-1 page, an H-Wiki page.

A D-Wiki contains the command for including the Daily-Class-1 and acquires average sensor data for each hour in a day. **Figure 11** shows the daily-1-1 page, a D-Wiki page.

Wiki Bot-1 is placed in the server room for environmental monitoring. It reads an H-Wiki that corresponds to the current time, interprets the commands on the page and the program on the sensors-1-class page, and writes back the current sensor data to the H-Wiki. The sensors-1-class page contains the program that issues commands to sensors of Wiki Bot-1 to obtain sensor values.

Wiki Bot-2 is placed in our research room. It reads the D-Wiki that corresponds to the current date, interprets the commands on the page and the program on the Daily-Class-1 page, and writes back the current sensor data to the D-Wiki. The Daily-Class-1

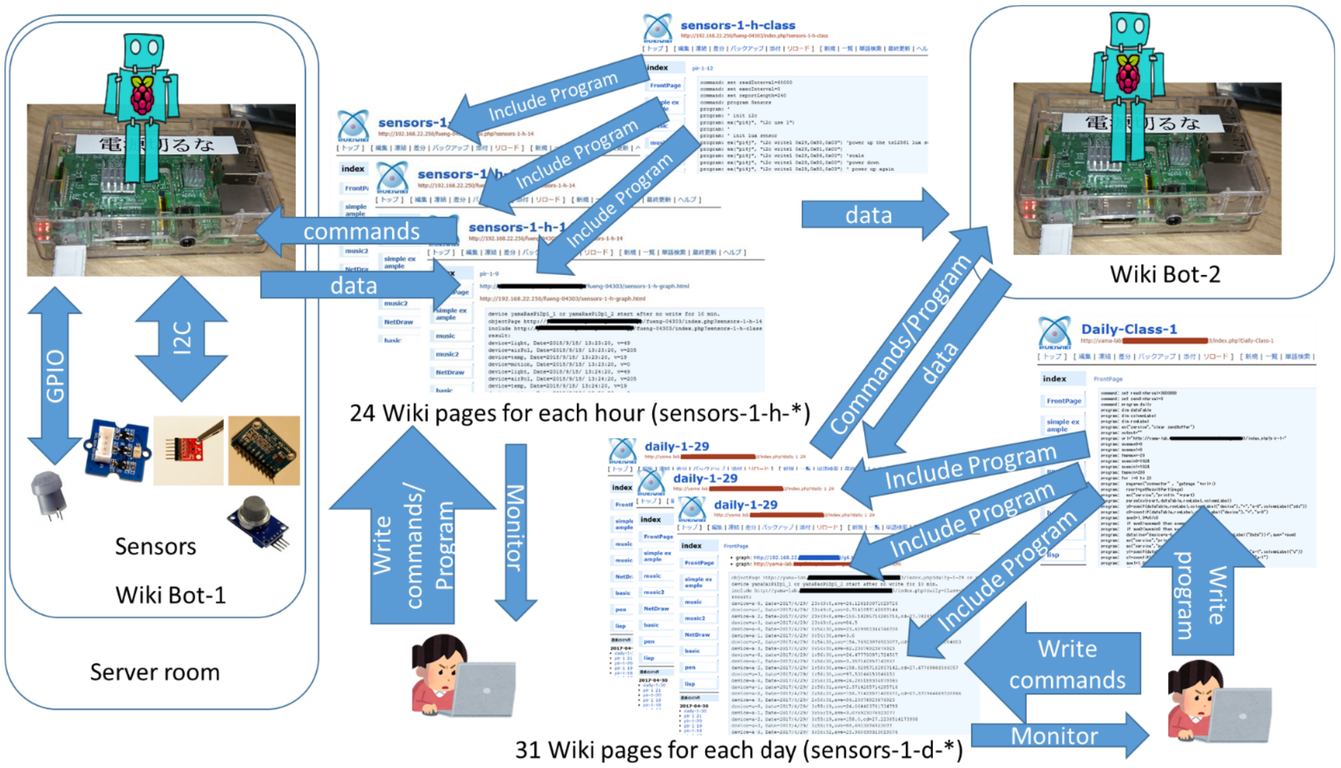


Fig. 8 Outline of the Wiki IoT system which monitors the server room.

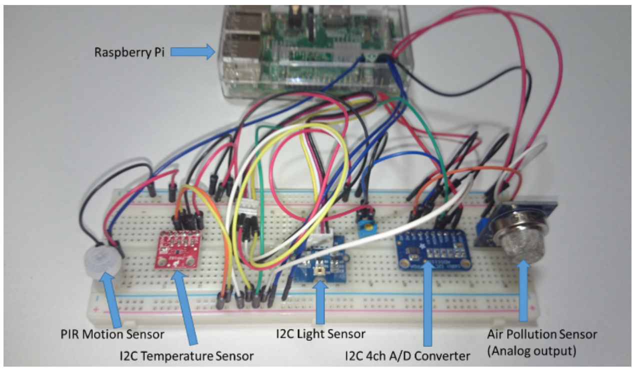
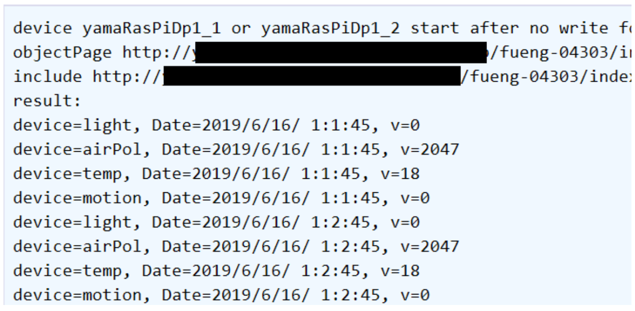


Fig. 9 Sensor Bot, a Wiki Bot with a temperature sensor, a light sensor, a PIR motion sensor and Gas sensor.

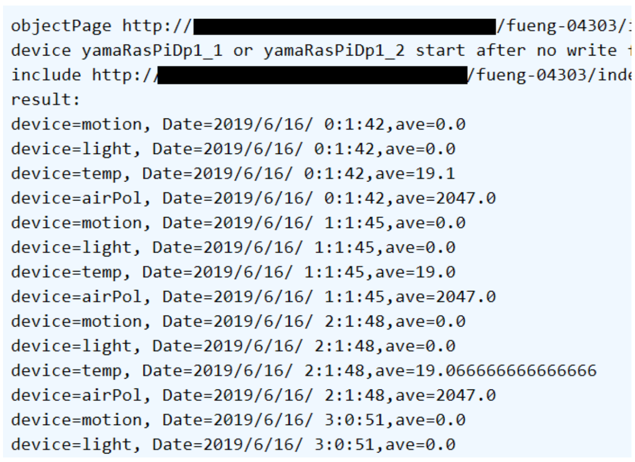


```
device yamaRasPiDp1_1 or yamaRasPiDp1_2 start after no write f
objectPage http://[redacted]/fueng-04303/i
include http://[redacted]/fueng-04303/inde
result:
device=light, Date=2019/6/16/ 1:1:45, v=0
device=airPol, Date=2019/6/16/ 1:1:45, v=2047
device=temp, Date=2019/6/16/ 1:1:45, v=18
device=motion, Date=2019/6/16/ 1:1:45, v=0
device=light, Date=2019/6/16/ 1:2:45, v=0
device=airPol, Date=2019/6/16/ 1:2:45, v=2047
device=temp, Date=2019/6/16/ 1:2:45, v=18
device=motion, Date=2019/6/16/ 1:2:45, v=0
```

Fig. 10 A part of the sensors-1-h-1 page.

page contains the program which computes the average values of data for each H-Wiki and writes back the results to the D-Wiki for the current day (Fig. 11).

H-Wikis and D-Wikis have a link to a page which displays graphs of their data. Administrators can see trends in the data, such as the temperature of the room.

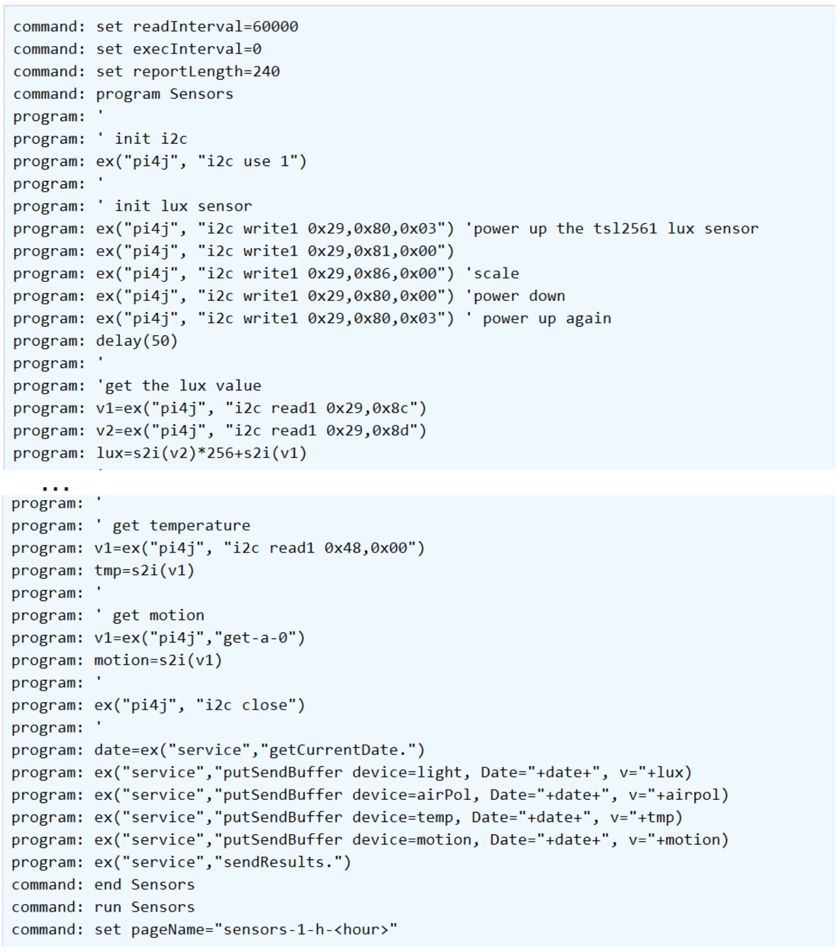


```
objectPage http://[redacted]/fueng-04303/:
device yamaRasPiDp1_1 or yamaRasPiDp1_2 start after no write 1
include http:/[redacted]/fueng-04303/inde
result:
device=motion, Date=2019/6/16/ 0:1:42,ave=0.0
device=light, Date=2019/6/16/ 0:1:42,ave=0.0
device=temp, Date=2019/6/16/ 0:1:42,ave=19.1
device=airPol, Date=2019/6/16/ 0:1:42,ave=2047.0
device=motion, Date=2019/6/16/ 1:1:45,ave=0.0
device=light, Date=2019/6/16/ 1:1:45,ave=0.0
device=temp, Date=2019/6/16/ 1:1:45,ave=19.0
device=airPol, Date=2019/6/16/ 1:1:45,ave=2047.0
device=motion, Date=2019/6/16/ 2:1:48,ave=0.0
device=light, Date=2019/6/16/ 2:1:48,ave=0.0
device=temp, Date=2019/6/16/ 2:1:48,ave=19.066666666666666
device=airPol, Date=2019/6/16/ 2:1:48,ave=2047.0
device=motion, Date=2019/6/16/ 3:0:51,ave=0.0
device=light, Date=2019/6/16/ 3:0:51,ave=0.0
```

Fig. 11 A part of the daily-1-1 page.

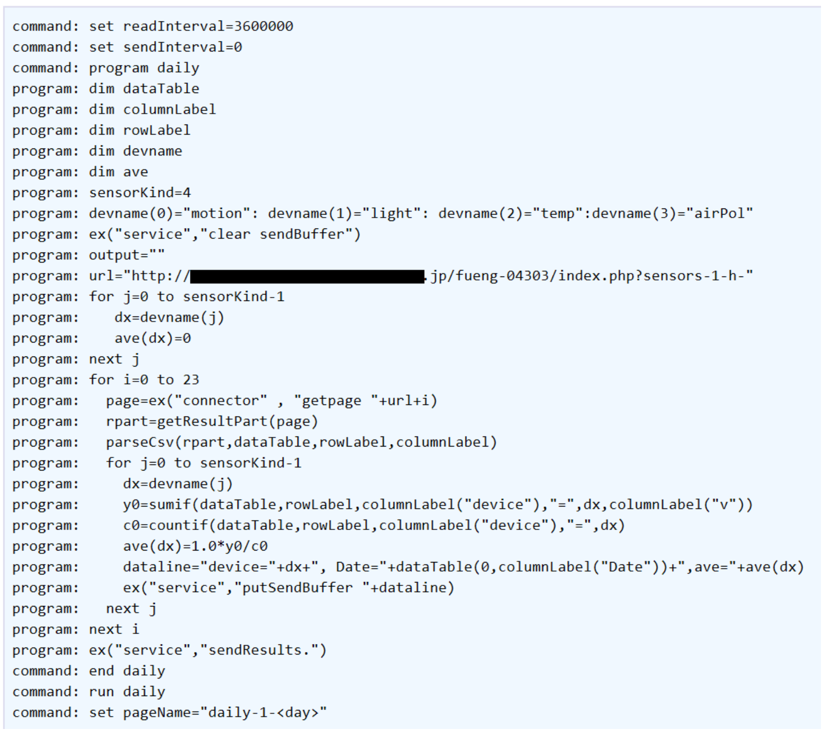
Figure 12 shows an excerpt of the sensors-1-h-class page. *command: set readInterval* = 60000 tells Wiki Bot to read this page every minute. *command: set execInterval* = 0 tells the Wiki Bot to execute the commands and program on this page immediately after this page has been read. *command: set reportLength* = 240 sets the maximum line number for the results part of the object page to 240. To display additional lines, an old line is first deleted.

The line *program: ex("pi4j", "i2c use 1")* sets the I2C channel No.1 for I2C communication. The lines *program: ex("pi4j", "i2c....")* sends an I2C command to an I2C device of the Sensor Bot. For example, *program: ex("pi4j", "i2c write1 0x29, 0x80, 0x03")* writes the one-byte value of 0x03 to the register 0x80 of the device, the I2C address of which is 0x29. *program: delay(t)* sets a delay to *t* milli seconds.



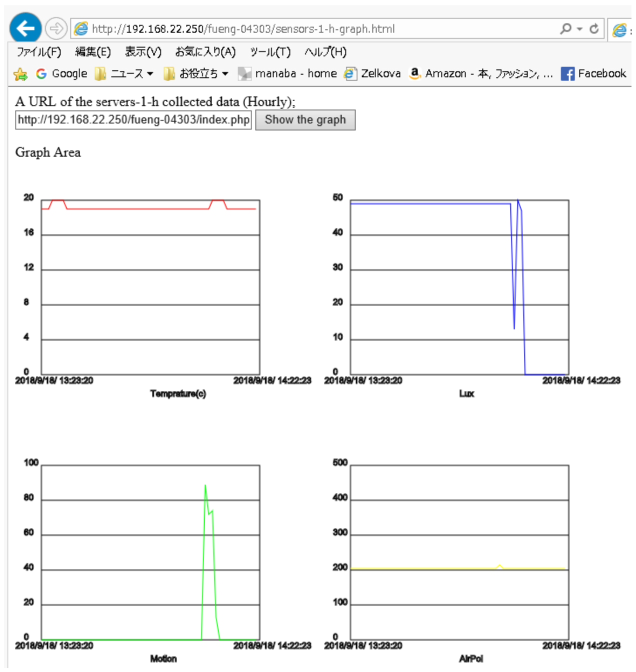
```
command: set readInterval=60000
command: set execInterval=0
command: set reportLength=240
command: program Sensors
program: '
program: ' init i2c
program: ex("pi4j", "i2c use 1")
program: '
program: ' init lux sensor
program: ex("pi4j", "i2c write1 0x29,0x80,0x03") 'power up the tsl2561 lux sensor
program: ex("pi4j", "i2c write1 0x29,0x81,0x00")
program: ex("pi4j", "i2c write1 0x29,0x86,0x00") 'scale
program: ex("pi4j", "i2c write1 0x29,0x80,0x00") 'power down
program: ex("pi4j", "i2c write1 0x29,0x80,0x03") ' power up again
program: delay(50)
program: '
program: 'get the lux value
program: v1=ex("pi4j", "i2c read1 0x29,0x8c")
program: v2=ex("pi4j", "i2c read1 0x29,0x8d")
program: lux=s2i(v2)*256+s2i(v1)
  ...
program: '
program: ' get temperature
program: v1=ex("pi4j", "i2c read1 0x48,0x00")
program: tmp=s2i(v1)
program: '
program: ' get motion
program: v1=ex("pi4j","get-a-0")
program: motion=s2i(v1)
program: '
program: ex("pi4j", "i2c close")
program: '
program: date=ex("service","getCurrentDate.")
program: ex("service","putSendBuffer device=light, Date="+date+", v="+lux)
program: ex("service","putSendBuffer device=airPol, Date="+date+", v="+airpol)
program: ex("service","putSendBuffer device=temp, Date="+date+", v="+tmp)
program: ex("service","putSendBuffer device=motion, Date="+date+", v="+motion)
program: ex("service","sendResults.")
command: end Sensors
command: run Sensors
command: set pageName="sensors-1-h-<hour>"
```

Fig. 12 A part of sensors-1-h-class page.



```
command: set readInterval=3600000
command: set sendInterval=0
command: program daily
program: dim dataTable
program: dim columnLabel
program: dim rowLabel
program: dim devname
program: dim ave
program: sensorKind=4
program: devname(0)="motion": devname(1)="light": devname(2)="temp":devname(3)="airPol"
program: ex("service","clear sendBuffer")
program: output=""
program: url="http://[redacted].jp/fueng-04303/index.php?sensors-1-h-"
program: for j=0 to sensorKind-1
program:   dx=devname(j)
program:   ave(dx)=0
program: next j
program: for i=0 to 23
program:   page=ex("connector" , "getpage "+url+i)
program:   rpart=getResultPart(page)
program:   parseCsv(rpart,dataTable,rowLabel,columnLabel)
program:   for j=0 to sensorKind-1
program:     dx=devname(j)
program:     y0=sumif(dataTable,rowLabel,columnLabel("device"),"=",dx,columnLabel("v"))
program:     c0=countif(dataTable,rowLabel,columnLabel("device"),"=",dx)
program:     ave(dx)=1.0*y0/c0
program:     dataline="device="+dx+", Date="+dataTable(0,columnLabel("Date"))+",ave="+ave(dx)
program:     ex("service","putSendBuffer "+dataline)
program:   next j
program: next i
program: ex("service","sendResults.")
command: end daily
command: run daily
command: set pageName="daily-1-<day>"
```

Fig. 13 The Daily-Class-1 page.


http://192.168.22.250/fueng-04303/sensors-1-h-graph.html
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
Google ニュース お役立ち manaba - home Zelkova Amazon - 本, ファッション, … Facebook
Show the graph
Graph Area
2018/9/18/ 13:23:20
2018/9/18/ 14:22:23
Temprature(c)
Lux
Motion
AirPol

A URL of the servers-1-h collected data (Hourly);
http://192.168.22.250/fueng-04303/index.php

Fig. 14 Changes of temperature, lux, motion, air pollution values in the monitored server room of a day, the data of which are collected by the Wiki IoT system.

The lines *program: <variable> = ex("pi4j", "i2c read1…")* sends a command to read one byte to an I2C device and the returned value is assigned to a variable. For example, *program: v1 = ex("pi4j", "i2c read1 0x29,0x8c")* sends a command to read one from register 0x8c to the I2C device, the address of which is 0x29, and the returned value is assigned to the variable v1. The line *program: v1 = ex("pi4j", "get-a-0")* gets the PIR motion value. The line *program: date = ex("service", "getCurrentData.")* assigns the current date to the variable date.

Figure 13 shows the Daily-Class-1 page. This class is read and executed every hour. For each page of H-Wiki (one per hour), the data of the corresponding page is read and the csv values of the page are stored in the *dataTable* array. The average values for each sensor are calculated using *sumif* and *countif* functions and written on the object page of the corresponding D-Wiki page.

Figure 14 shows the changes of temperature, brightness, motion, air pollution values in the monitored server room for a day, collected by the Wiki IoT system.

5. Related Work

5.1 Configuring Organizational Network Infrastructure

There are many ways to monitor servers. A designated virtual LAN (VLAN) segment is commonly used for monitoring servers in the infrastructural switching network of an organization. This method does not require any additional equipment. However, configuring the VLAN segment for an infrastructural switching network requires careful design and careful operation because an error may take down the whole infrastructural switching network. The configuration and operation of an infrastructural switching network are commonly outsourced. With outsourcing, the configuration for monitoring just one server is not easy for request and is not fast, especially for end users.

The Wiki IoT system may realize easy configuration for end users.

5.2 Virtual Private Network

For accessing sensor devices behind a NAT router from outside the LAN (i.e., tunneling the NAT router), a virtual private network (VPN), such as SoftEther VPN [16], is commonly used. A VPN may be deployed by the end-user without changing the settings of the infrastructural switching network. However, the deployment of a VPN that is suitable for accessing sensor devices requires changing the settings of the remote access VPN. A VPN can be a security hole if used incorrectly or used by a malicious third party. If the remote access VPN is taken over by a malicious third party, the whole computer hosting the VPN client software may be accessed by the malicious third party.

Wiki IoT does not need settings of network devices to be changed. Wiki IoT is safer than the VPN because it does not have the ability to access the whole computer hosting the Wiki Bot.

5.3 Webalizer

Webalizer [17] is a fast, free web server log file analysis program. It is a popular tool for visualizing the usage of a Web server. Because Webalizer is installed on the target Web server, usage cannot be viewed from outside the network if the server is not accessible from the outside. In contrast, our monitoring system allows administrators to monitor the target server from the outside.

5.4 New Relic

New Relic [18] is a popular commercial server monitoring service. New Relic can monitor almost all servers, both those accessible and inaccessible from outside the network. The user can monitor the data stored on the New Relic's Web server. However, the monitoring program for New Relic has to be installed on the target server. In contrast, our monitoring system allows administrators to monitor the target server without the installation of a program on the target server.

5.5 Kaseya and UNIFAS

KASEYA [19] is a system for remotely controlling personal computers. UNIFAS [20] is a system for remotely controlling WiFi access points. These systems are used for controlling a large number of devices of PCs or WiFi access points and can control devices behind a NAT router. Kaseya and UNIFAS require their own Web servers. In contrast, Wiki IoT uses common Wiki servers.

5.6 Obniz

The platform device obniz [21] can connect to electrical components such as motors and sensors. A device can connect to the obniz Cloud via a network (the obniz board uses Wi-Fi to connect).

After connection, the user can control the connected motors and sensors by calling APIs remotely. Obniz receives and controls remote devices behind a NAT router using a WebSocket.

It uses the obniz cloud for communication. In contrast, Wiki IoT uses common Wiki servers for communication.

5.7 NetNuclues Cloud Hub

NetNucleus Cloud Hub [22] can control IoT devices behind firewalls via an HTTPS/WebSocket connection from Internet servers. This process is very similar to the connection between a wiki page and a bot in our IoT system, even though our connection uses HTTP instead of HTTPS/WebSocket. However, although both systems allow administrators to control IoT devices behind firewalls, NetNucleus Cloud Hub does not have reconfigurable IoT devices, whereas our IoT system does.

5.8 Monitoring by Remote-Controlled Mobile Robot

Ogawa and Yoshiura proposed a method of collecting monitoring information using a remote-controlled mobile robot [23]. They conducted experiments in which monitoring devices were attached to media converters at a university, and the mobile robot collected monitoring information. The experimental results showed that the proposed method has a communication ability equivalent to that in previous research and that it can quickly collect monitoring information.

Their method can monitor a network equipment behind a physical door and does not take up a LAN port of the target network equipment.

Their method uses a physical mobile robot, whose path must be secured and maintained.

Although Wiki IoT takes up LAN ports of the target network, it does not need a path.

5.9 WxBeacon2

WxBeacon2 [24] is a commercial IoT sensor device, that collects environmental information, such as temperature, humidity, atmospheric pressure, brightness, ultraviolet rays, and noise. A smart phone application, Weather News Touch, is used to view the information collected by the WxBeacon2 and upload the data to a Weather News Inc. server. The smart phone running Weather News Touch must be physically near the WxBeacon2 device.

It is difficult to transform the data collected by WxBeacon2 into other formats, such as CSV, without reverse engineering. In contrast, our monitoring system can output collected data in any text format (e.g., CSV format) because the user can write commands that produce the data. In addition to the monitoring system users, other people can use the data by reading the URL of the Wiki page. They can then analyze the data using tools such like R and Python.

6. Concluding Remarks

Our Wiki IoT systems enhances the reliability of the target servers and server rooms. It has reduced our work load and mental stress.

There are several opportunities to improve our monitoring method, such as enhancing the security and enhancing the availability.

To enhance the security of Wiki IoT, Wiki Bots can be protected by a NAT router. However, Wiki pages are currently protected using basic authentication. TLS should be used to enhance Wiki page security.

In order to enhance the availability of the Wiki IoT, we are trying to introduce cross-over including and cross-over execution [25].

Debugging is another challenge for Wiki IoT. Errors that occur when commands are executed by a bot are not currently shown on the Wiki page. Such errors must be viewed on the Wiki page.

Acknowledgments A part of this research was supported by JSPS KAKENHI Grant Number JP16K00197, JP15H03055. We also thank students who helped us to develop monitoring systems.

References

[1] Masuya, M., Yamanoue, T. and Kubota, S.: An Experience of Monitoring University Network Security Using a Commercial Service and DIY Monitoring, *Proc. 34th Annual ACM SIGUCCS Conference on User Services*, pp.225–230, ACM (online), DOI: http://doi.acm.org/10.1145/1181216.1181267 (2006).

[2] Mattauch, T., Hatoum, R. and Pettit, H: Building a call center in 2 days: How a world class support center responds to crisis, *Proc. 40th Annual ACM SIGUCCS Conference on User Services*, pp.97–100, ACM (online), DOI: http://doi.acm.org/10.1145/2382456.2382478 (2012).

[3] Puri, R.: Bots & Botnet: An Overview, SANS InfoSec Reading Room (Dec. 2003), available from ⟨http://www.sans.org/rr/whitepapers/malicious/⟩.

[4] Yamanoue, T., Oda, K. and Shimozono. K: Capturing Malicious Bots using a Beneficial Bot and Wiki, *Proc. 40th Annual ACM SIGUCCS Conference on User services*, pp.91–96, ACM (online), DOI: https://doi.org/10.1145/2382456.2382477 (2012).

[5] Yamanoue, T., Oda, K. and Shimozono, K.: A Malicious Bot Capturing System using a Beneficial Bot and Wiki, *Journal of Information Processing* (*JIP*), Vol.21, No.2, pp.237–245 (2013).

[6] Yamanoue, T., Oda, K. and Shimozono. K.: An Inter-Wiki Page Data Processor for a M2M System, *Proc. 4th International Conference on E-Service and Knowledge Management* (*ESKM 2013*), *2013 IIAI International Conference on Advanced Applied Informatics* (*IIA-IAAI*), pp.45–50, IEEE (online), DOI: https://doi.org/10.1109/IIAI-AAI.2013.48 (2013).

[7] Yamanoue, T., Oda, K. and Shimozono. K.: Experimental Implementation of a M2M System Controlled by a Wiki Network, In Applied Computing and Information Technology, Studies in Computational Intelligence, Vol.553, pp.121–136, Springer (2014).

[8] Yamanoue, T. and Muye, L.: Experimental implementation of an IoT system which controls sensor terminals of a sensor network by a Wiki page on the Internet, IPSJ SIG Technical Reports, Vol.2017-IOT-36, No.12, pp.1–8 (2017).

[9] Yamanoue, T., Yokoyama, D., Umeda, R., Morita, S., Ozeki, T. and Nakamichi, N.: An IoT System with Remote Re-configurable Wireless Sensor Network Nodes and Its Application to Measure Activity of a Class, *7th International Conference on E-Service and Knowledge Management* (*ESKM 2018*), Yonago, Japan (2018).

[10] PukiWiki: available from ⟨https://en.wikipedia.org/wiki/PukiWiki⟩ (accessed 2019-06-17).

[11] Yamanoue, T., Oda, K. and Shimozono, K.: A Simple Application Program Interface for Saving Java Program Data on a Wiki, Advances in Software Engineering, Vol.2012, Article ID 981783, Hindawi Publishing Corporation (online), DOI: http://doi.org/10.1155/2012/981783 (2012).

[12] Yamanoue, T. et al.: A Casual Big and Wide Digital Signage System and its Automatic Operation System, IPSJ SIG Technical Reports, Vol.2019-IOT-47, pp.1–8 (2019).

[13] Yamanoue, T., Fuse, I., Okabe, S., Nakamura, A., Nakanishi, M., Fukada, S., Tagawa, T., Tatsumi, T., Murata, I, Uehara, T., Yamada, T. and Ueda, H.: Computer Ethics Video Clips for University Students in Japan from 2003 until 2013, *Proc. 38th Annual International Computer Software & Applications Conference* (*COMPSAC2013/ADMNET WS*), pp.96–101, IEEE (2014).

[14] Yamanoue, T., Nakamichi, N. and Kaneko, K.: Enhancing Campus Cyber Security through a Class with Combination of Computer Ethics Videos and Logical Thinking, *Proc. ACM on SIGUCCS Annual Conference* (*SIGUCCS'16*), pp.117–123, ACM (online), DOI: http://doi.acm.org/10.1145/2974927.2974939.

[15] About rules and best practices with account behaviors, available

from ⟨https://help.twitter.com/en/rules-and-policies/twitter-rules-and-best-practices⟩ (accessed 2019-09-21).

[16] SoftEther VPN, available from ⟨https://ja.softether.org⟩ (accessed 2019-09-12) (in Japanese).

[17] Webalizer, available from ⟨http://www.webalizer.org⟩ (accessed 2017-05-04).

[18] New Relic, available from ⟨https://newrelic.com/⟩ (accessed 2017-05-04).

[19] Kaseya, available from ⟨https://www.kaseya.com⟩ (accessed 2019-09-12).

[20] UNIFAS, available from ⟨http://www.furunosystems.co.jp/product/detail/unifas.html⟩ (accessed 2019-09-12) (in Japanese).

[21] obniz, available from ⟨https://obniz.io⟩ (accessed 2019-09-12).

[22] NetNueCleus Cloud Hub, available from ⟨https://www.tjsys.co.jp/embedded/netnucleus-cloudhub/index_j.htm⟩ (accessed 2019-09-12) (in Japanese).

[23] Ogawa, K. and Yoshiura, N.: A Monitoring Method for Network Devices Using Mobile Robots and Small Computers, *Journal of Information Processing*, Vol.60, No.2, p.668–679 (2019).

[24] WxBeacon2, available from ⟨https://weathernews.jp/smart/wxbeacon2/⟩ (accessed 2019-09-12) (in Japanese).

[25] Yamanoue, T.: Bot Computing using the Power of Wiki Collaboration, *Proc. 8th International Congress on Advanced Applied Informatics* (*IIAI-AAI*), pp.17–24 (online), DOI: https://doi.org/10.1109/IIAI-AAI.2019.00015 (2019).

Takashi Yamanoue received his B.S. M.S. and Ph.D. in computer science from Kyushu Institute of Technology, Kitakyushu, Japan, in 1982, 1984 and 1993, respectively. He was a Ph.D. candidate of the Interdisciplinary Graduate School of Engineering Sciences, Kyushu University. He is the dean and a professor of the school of engineering, Fukuyama University. His research interests include IoT, distributed computing, compiler-compilers, web mining and computer assisted teaching systems. He is a member of IEEE, ACM, Information Processing Society of Japan (IPSJ), The Institute of Electronics, Information and Communication Engineers (IEICE), the Robotics Society of Japan (JRSJ). He is also a member of the ACM SIGUCCS Hall of Fame.