Electronic Preprint for Journal of Information Processing Vol.30

Regular Paper

A Campus Equipment Controller Using an IoT System that Can Configure and Control its Edge Devices Behind a NAT Using Wiki Pages on the Internet

Takashi Yamanoue[1,a)]

Received: May 30, 2021, Accepted: December 3, 2021

Abstract: This paper describes the development of an electric appliance controller using an IoT system. The IoT system can configure and control its edge devices which are placed behind a Network Address Translator (NAT) using Wiki pages. These operations were realized by Bot Computing, a framework for Internet of Things (IoT). Any electric appliance can be controlled using the combination of Bot Computing and edge devices with an Infra-Red (IR) transmitter, if the appliance has the IR remote controlled function. We can program the power on/off time of any electric appliances, by writing a script on a Wiki page on the Internet, using Bot Computing. We can change the program anytime, anywhere. We also can control turning on or off the appliance anytime, anywhere.

Keywords: remote control, network management, Wiki, Bot, IoT

1. Introduction

Campus technical managers have to manage a lot of equipment, such as air conditioners, lights in classrooms, video projectors and so on, on our campuses. And usually, the number of them is increasing. Their number and kind are much larger than such equipment at home. However, managers should ensure their campuses are green in order to keep the earth green and in order to maintain the sound finance of our campuses. In order to operate such equipment effectively while keeping the campus green, frequent and flexible power on and power off operations are required. Managers are not in a position to do such operation themselves because they are too busy. So, managers use appliance timers [1] or sensors for the automatic operation of campus equipment. However, it is hard to change the settings of a lot of campus equipment manually. It is almost impossible to do irregular operation of a piece of equipment in a locked building at night manually.

There has been an attempt to carry out an automatic and flexible operation of such equipment using *Bot Computing* [2], [3], [4]. Bot computing enables the remote control of edge devices which are protected by NAT routers, from the Internet. Any electric appliances can be controlled using the combination of Bot Computing and edge devices with an Infra-Red (IR) transmitter, if the appliance has the IR remote controlled function.

We can program the power on time and the power off time of any electric appliances, by writing a script on a Wiki page on the Internet using Bot Computing. We can change the program anytime, anywhere. We can also command the turning on or off the appliances anytime, anywhere. We have used this method to realize a big digital signage system of our university using windows of a building in our university. The signage system displays university information at night. We could have scheduled operation of video projectors in a building. We also could have commanded the turning on of video projectors and have changed the schedule of operation from the outside of the building at night, when it is is locked.

2. Bot Computing and Wiki IoT System

A remote-controlled computer or program is a *bot* in this paper. Bots are often malicious programs that form a botnet [5], but they can also be used for beneficial tasks [2], [3], [4], [6]. We define *Bot computing* as computing by bots. Bot computing is parallel computing. The bot, called *Wiki Bot*, is Raspberry Pi [7] that runs the bot software which is written in a Wiki page. Some of them are equipped with actuators and sensors. 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 which are equipped with a WSN transmitter, are gateways to the WSN. The IoT system in this paper 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], [10]. We adopt PukiWiki software [11] for the Wiki IoT because PukiWiki is simpler to de-

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

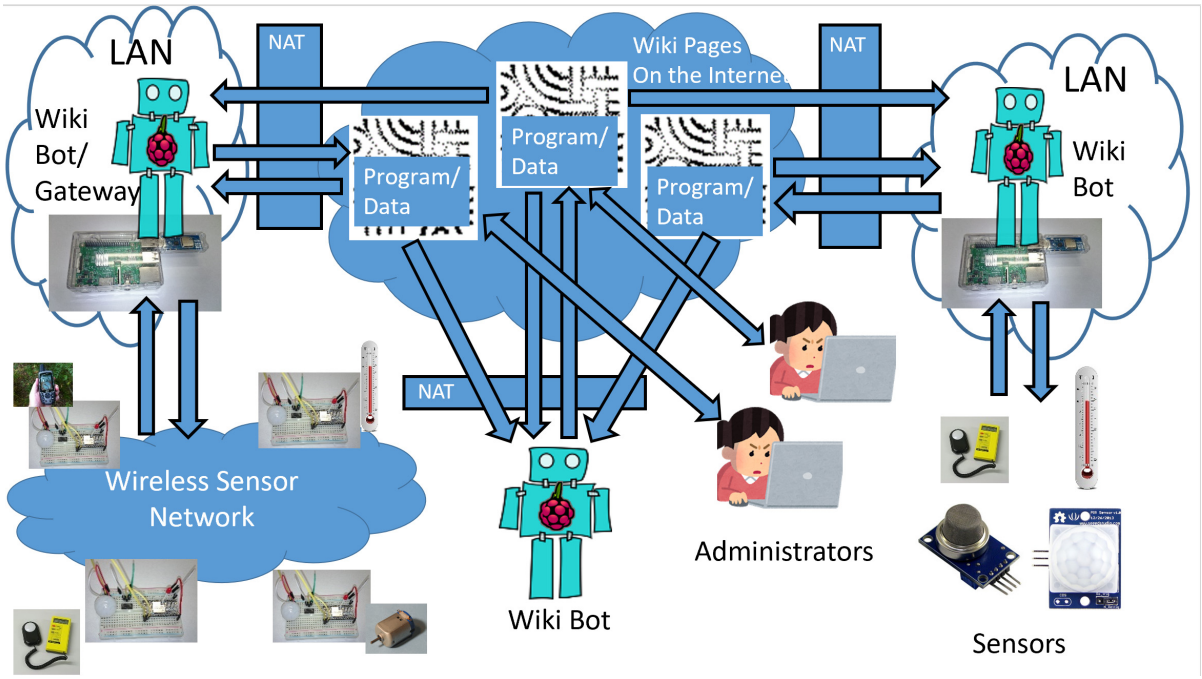

Fig. 1 Outline of a Wiki IoT system.

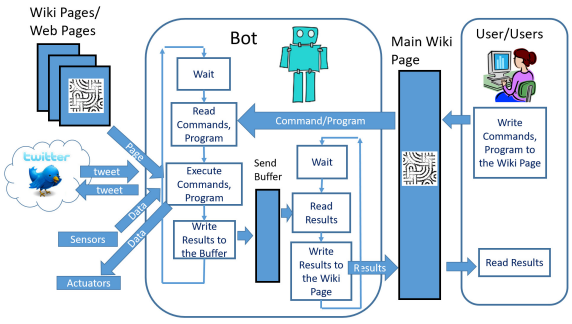

Fig. 2 Outline of behavior of a bot.

ploy than many other wikis. PukiWiki has minimal requirements, not even needing data base engines such as MySQL. By extracting PukiWiki tar-ball to a web server root directory, it works well [12]. When an appliance is placed in a campus building that cannot be accessed from the outside directly, it is impossible to directly control the appliance from outside of the building. However, such controlling would be convenient when administrators are not in the building. A Wiki Bot [3], [4] can help administrators control the appliance in such cases. Buildings are usually physically isolated from the outside at night. There are no people in the building at night.

2.1 Behavior of a Wiki Bot

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

(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*. Sensors are not considered in this paper. Wiki IoT systems with sensors are described in other papers [4].

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.

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 it.

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

```
objectPage http://[redacted]/index.php?Basic
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no wr
command: set readInterval=60000
command: set execInterval=0
command: program ex1
program: ex("service","clear sendBuffer")
program: s=0
program: for i=0 to 10
program:   s=s+i
program:   ex("service","putSendBuffer "+s)
program: next i
program: ex("service","sendResults.")
command: end ex1
command: run ex1
result:
0
1
3
6
10
15
21
28
36
45
55
currentDevice="yamaRasPiDp9_1",Date=2018/8/13/ 0:56:3
```

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

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/")

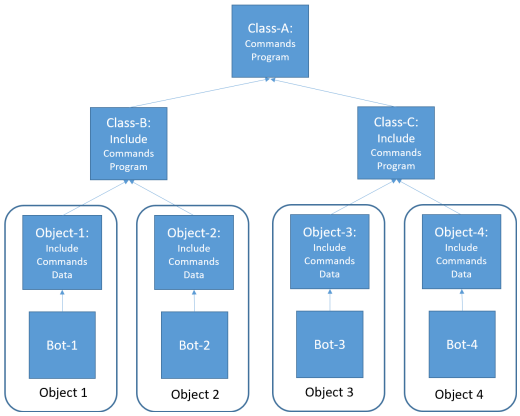

Fig. 4 An example of the program of a Wiki Bot and its output after the execution of the program.

2.4 Class Pages and Object Pages

Wiki IoT is an object-oriented computing system [3], [13]. 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 function 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. Arduino IR Remote Control Transceiver

Many modern appliances are equipped with IR remote control function. We have designed and implemented the Wiki Bot with the Arduino IR remote control transceiver (IR transceiver), in order to control appliances with the remote-control function (**Fig. 5**). Arduino is a popular micro controller for hobbyists [14].

The hardware of a Wiki Bot with the IR transceiver consists of a Raspberry Pi and an IR transceiver. They are connected by a USB cable. The IR transceiver consists of an Arduino Nano micro controller, an IR remote control signal receiver, and an IR

LED. They are connected by a bread board and some jumper cables. **Figure 6** shows the circuit of the IR transceiver. In this Figure, two buttons and two 470 k ohm registers are also used in the circuit. These buttons and registers are used to test the function of the transceiver. The sketch (program), which is written based on the program by Mr. Shirriff [15], is written in the Arduino of the IR transceiver. The sketch has the following functions.

a. Receives a command from the USB serial interface, interprets the command, transforms the command into the code for the IR signal, and sending the IR signal, which is corresponding to the command, to an appliance using the IR LED. When the signal is transmitted from the IR LED, when the appliance receives the signal, and when the received signal was appropriate for the appliance, the appliance plays the corresponding action to the command. The format of a command is "*irCmd <maker><code><code-length>;*". In this format, *<maker>* shows the format of the *<code>*. *<code>* shows the IR code in hexadecimal value. *<code-length>* is the length of the IR code.

b. Receives the signal of IR remote controller of an appliance using the IR remote control signal receiver, analyzing the signal and displaying the code of the signal to the serial monitor of the Arduino IDE.

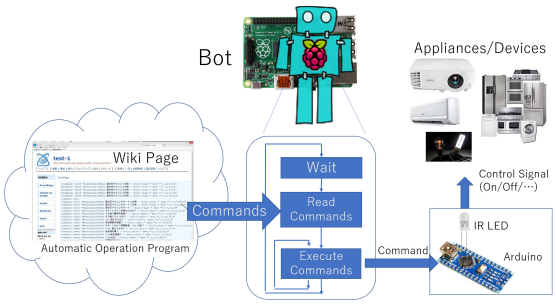

Fig. 5 Wiki Bot with the Arduino IR remote controller.

Figure 7 shows a part of the sketch. In the figure, the line, No.310, "*String str = Serial.readStringUntil(';');*" shows that the program receives the command line which is ended

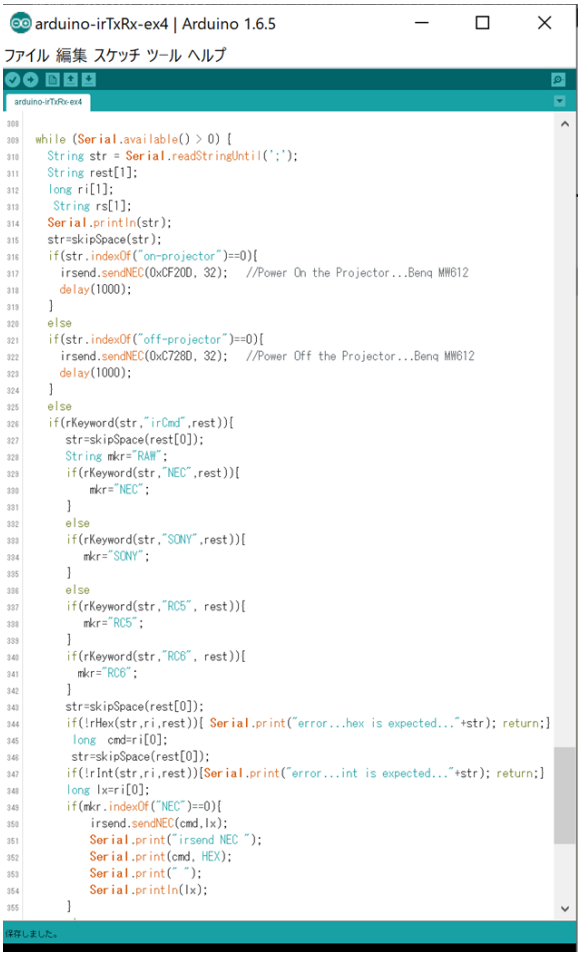

Fig. 7 A part of the sketch of the Arduino IR remote control transceiver (IR transceiver).

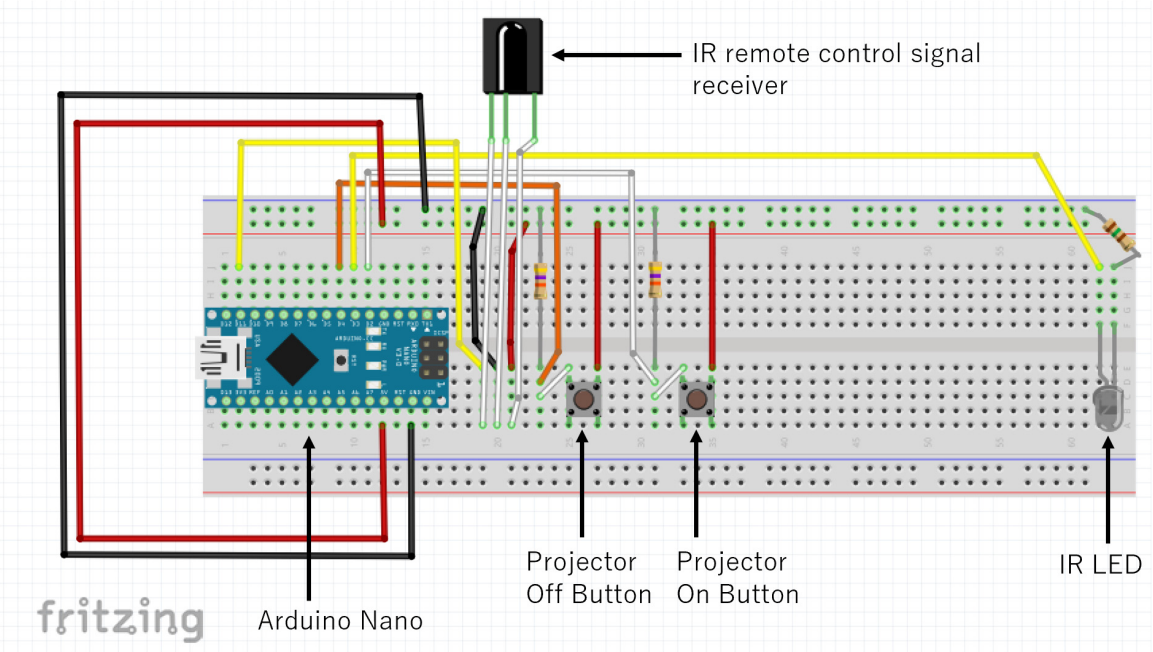

Fig. 6 Circuit of the Arduino IR remote control transceiver (IR transceiver).

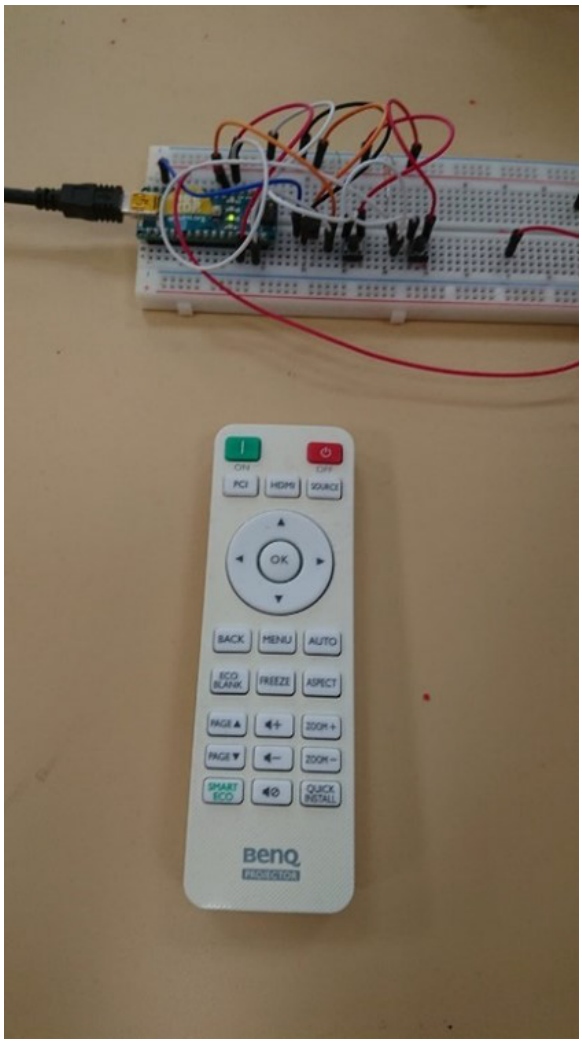

Fig. 8 Receiving the IR command using the IR transceiver.

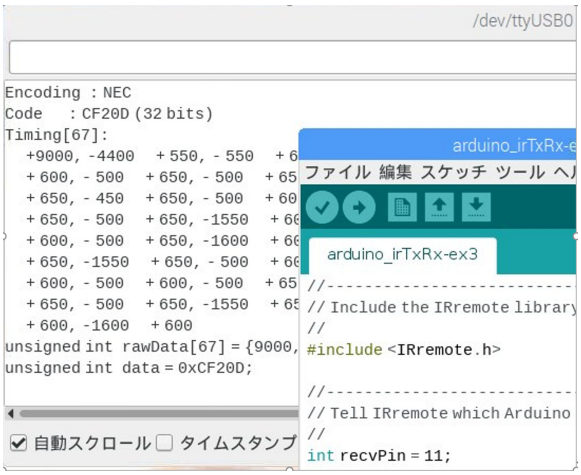

Fig. 9 Received IR command by the IR transceiver.

with the letter "*;*" from the USB serial interface and assigns the command line into the variable "*str*". The line, No.326, "*if(rKeyword(str,"irCmd",rest))*{" shows that if the command line was started by the keyword "*irCmd*", the program deletes the keyword from the command line, passes the rest of the command line to the variable "*rest*", and performs the statements within the corresponding braces "{}". The line, No.344, "*if(!rHex(str,ri,rest))*{" shows that if there was a hexadecimal value at the head of the command line in the variable "*str*", the program reads the hexadecimal value, deletes the hexadecimal value from the command line, assigns the value into the array "*ri*", passes the rest of the command line into the variable "*rest*", and executes the following statements. The value in the element "*ri[0]*" shows the code in the command line. The line, No.350, "*irsend.sendNEC(cmd,lx)*" shows that the NEC format code in the variable "*cmd*" is sent to the IR LED. The variable "*lx*" shows the bit length of the code.

The IR remote control signal receiver is used for getting the IR command. When the IR remote controller of an appliance was pointed toward the IR remote control signal receiver of the IR transceiver, and a button of the IR remote controller was pressed, the corresponding code to the button is acquired by the Arduino Nano and the code is shown in the serial port monitor of the Arduino IDE. The code, which corresponds to a function of an appliance, can be acquired by this procedure. **Figure 8** shows the IR transceiver which is receiving the IR command from the projector controller. **Figure 9** shows the received IR command in the serial port monitor, which is received by the IR transceiver when the power on button of the controller was pushed. The format of the code, NEC, was also shown in the serial port monitor.

4. Automatic and Flexible Operation of Campus Equipment

Fukuyama university has a building at Fukuyama city downtown. We have made a big digital signage system using windows of the building and projectors. We were showing university information on the signage system at night. **Figure 10** shows a scene where the information of our university is displayed by the big digital signage.

Figure 11 shows the outline of the inside of the big digital signage system. This system uses multiple projectors in order to display a large size image without decreasing its brightness. Each projector displays the image on the display of the laptop PC for screen receiving (receiver PC). The original image is sent from the laptop PC for screen transmission (transmitter PC). Each receiver PC receives the original image and displays a part of the image. Distributed web screen share (DWSS) function of the portable cloud computing system [16] is used for this broadcast.

Projectors of the signage system should be turned off at daytime in order to save power and life of projectors' lights. In order to turn off the projector at daytime and to turn on at night, we were using hardware timers and timers in projectors.

When we wanted to change the time of turning on/off the signage system, we had to enter the building and change the settings of hardware timers and timers in the projectors.

It is not easy to change the time of turning on/off the signage system because it takes about 30 minutes by car from the main campus of our university to the building. And it was impossible to change the time of turning on/off at night, because we can't enter the building at night. In order to improve this situation, we made the automatic and flexible operation system ("our system" in the following) using bot computing. Our system was used for one month, for displaying the public relations video of the 400-th anniversary of the Fukuyama castle completion during September

Fig. 10 Display of the Big digital signage.

2019 [17].

Our system archieves the scheduled operation and the ordered turning on/off of equipment using the script on a Wiki page.

Figure 12 shows an outline of the power control part of the big digital signage system. An "Arduino IR remote control transceiver (IR transceiver)" is placed near each projector. The an IR transceiver sends IR signal, which turns each projector on/off, to the projector. An IR LED is connected to the IR receiver window of the projector. Each IR transceiver is connected to a Raspberry Pi using a USB cable. This Raspberry Pi is the Wiki bot of the Wiki IoT.

A Wiki Bot reads the Object page on a Wiki server periodically. There is an include command, which includes the Class page, in the object page. The script which turns all projectors on/off simultaneously is written in the Class page. So, this script is read

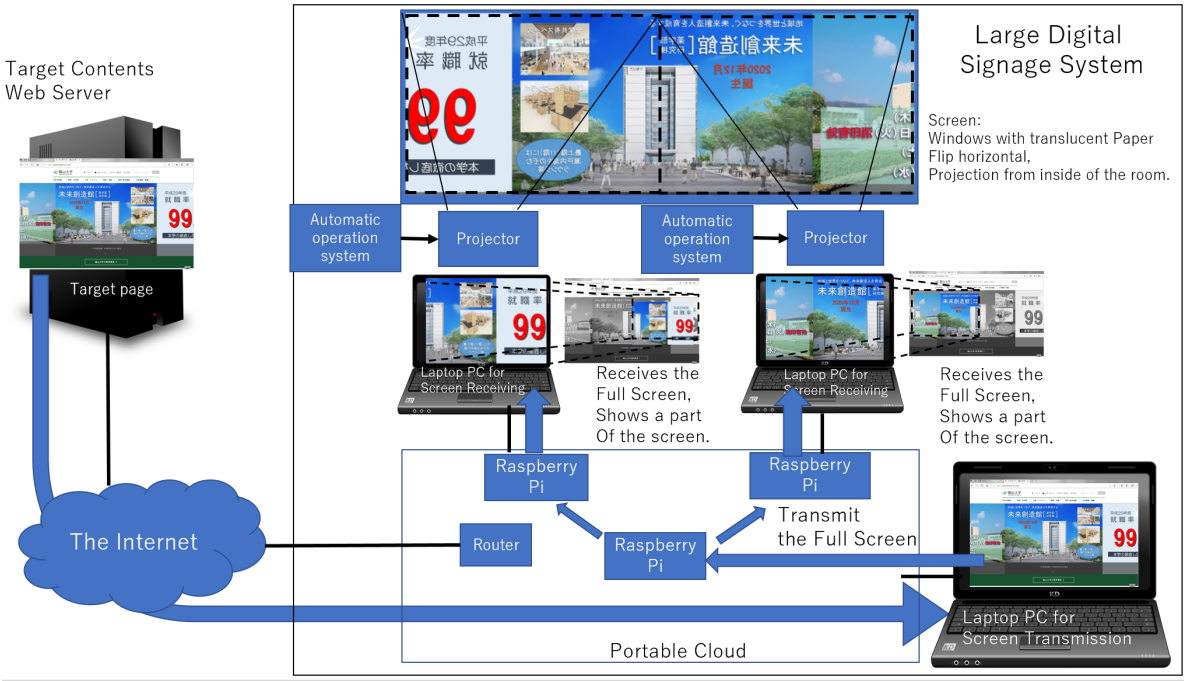

Fig. 11 Outline of the Inside of the Big digital signage system.

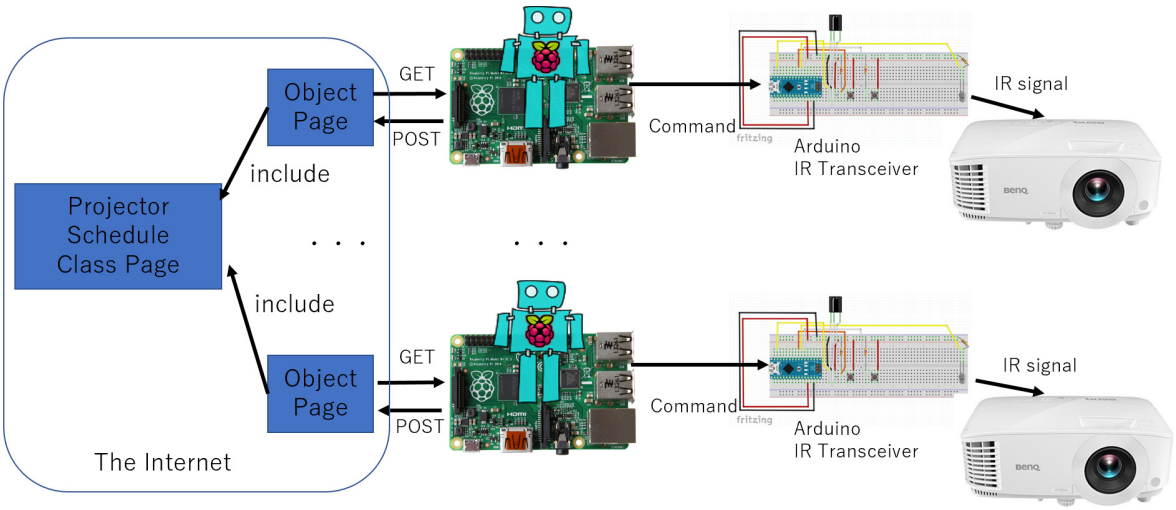

Fig. 12 Outline of the power control part of the big digital signage system.

```
command: set readInterval=120000
command: set execInterval=500
command: set sendInterval=600000
command: program ex1
program: def abs(x)= if x>0 then x else -x
program: def onPrj()=
program: {
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program:   delay(1000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program:   delay(1000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program: }
program: def offPrj()=
program: {
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program: }
program: dim comTab
program: ex("service","clear sendBuffer")
program: output=""
program: dx="2019/10/"
program: comTab(0,0)=dx+"1/ 18:30:00":comTab(0,1)="onPrj"
program: comTab(1,0)=dx+"1/ 22:15:00":comTab(1,1)="offPrj"
```

● ● ●

```
command: set readInterval=120000
command: set execInterval=500
command: set sendInterval=600000
command: program ex1
program: def abs(x)= if x>0 then x else -x
program: def onPrj()=
program: {
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program:   delay(1000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program:   delay(1000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")
program: }
program: def offPrj()=
program: {
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program:   delay(2000)
program:   ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")
program: }
program: dim comTab
program: ex("service","clear sendBuffer")
program: output=""
program: dx="2019/10/"
program: comTab(0,0)=dx+"1/ 18:30:00":comTab(0,1)="onPrj"
program: comTab(1,0)=dx+"1/ 22:15:00":comTab(1,1)="offPrj"
```

Fig. 13 A part of the class page which turns all projectors on/off simultaneously.

and executed by all wiki bots.

Figure 13 shows a part of the class page which turns all projectors on/off simultaneously. *def onPrj()=. . .* shows the function which sends the command, which transmits the signal of turn on code from the IR LED, from a Raspberry Pi to an IR transceiver. *ex("pi4j", "serial send ¥"irCmd NEC 0xCF20D 32;¥".")* in this function shows the command. "*pi4j*" shows the object which performs input and output of the Raspberry Pi. "*serial send <command>*" shows that the command "*irCmd NEC 0xCF20D 32;*" is sent to the IR transceiver using the USB serial interface of the Raspberry Pi. The IR transceiver receives the command and recognizes that this code is the command for the IR transceiver by looking at the head part of this command, irCmd. Then the transceiver transforms the command "*NEC 0xCF20D 32*" into the signal and the signal is sent to the projector. *NEC* shows that the format of this code is the format of NEC IR remote controller. *0xCF20D* is the code that turn on the projector and 32 shows that the length of this code is 32 bits. The code *NEC 0xCF20D* was acquired by the IR transceiver, by receiving the turn on signal from the original remote controller of the projector. *def offPrj()=. . .* shows the function which sends the command, which transmits the signal of turn off code from the IR LED, from a Raspberry Pi to a IR transceiver. *ex("pi4j", "serial send ¥"irCmd NEC 0xC728D 32;¥".")* in this function shows the command. "*irCmd NEC 0xC728D 32;*" is sent to the IR transceiver using the USB serial interface of the Raspberry Pi, and the turn off signal is transmitted to the projector.

5. Related Work

5.1 Virtual Private Network

For accessing edge devices behind a NAT router from outside the LAN (i.e., tunneling the NAT router), a virtual private network (VPN), such as SoftEther VPN [18], 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.

5.2 Monitoring of Servers using Bot Computing

Papers of Refs. [2] and [4] show the way using bot computing for doing beneficial things such as monitoring a server instead of doing malicious things. In these papers, information at local places is collected to servers on the Internet.

On the other hand, bot computing in this paper is used to control things at local places from the server on the Internet.

5.3 Hardware Timer

Many hardware timers [1] are commonly used for automating power on/off control of an appliance. Hardware timers can control power on/off of any kind of appliance, even if it does not have the IR remote control function. However, there is some equipment which is not recommended to have the power turned off by unplugging the power cord from power supply such as some projectors.

We had combined the timer in the projector and the hardware timer to enable the scheduled turning on/off of a projector before. It is not flexible as we wrote in section one.

As shown in section four, when we wanted to change the time of turning on/off the signage system, we could turn on the projectors and correct the turn on/off schedule immediately after we noticed the mistake, using the Wiki IoT. It took less than five minutes to open the web page and re-write the script, using a mobile phone. It would have been more than 14 hours to correct

the time if we were using the hardware timer, because we could not enter the building at night, 17:30 to 9:00, and it was 19:00 when we noticed the projectors did not light. Even if we could enter the building, we would have to borrow the key to open the room, go to the room, open the door of the room, enter the room, change both of the turn on/off :time of the hardware timer and the turn off time of the projector, confirm the settings of all, leave the room, close the door of the room, and return the key. It would take about 30 minutes to accomplish all of these steps. If we were far from the building, time for going to the building woud have to be added.

5.4 Smart Speaker

The automatic turning on/off of appliances can also be realized using the combination of Google Calendar, IFTTT [19] and a Smart Speaker [20] such as Google Echo. The user of these does not need to write the program. Our system needs to write the program. However, the program realizes more flexible control of appliances.

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

5.6 Smart Campus Using IoT

Jain and others have shown automation techniques and a module that works for a room automation and ease of access to appliances with digital control [22]. Their techniques did not show the scheduled operation of appliances and they did not implement their module's cooperation with cloud services at the time this paper was published.

5.7 Campus Automation System Using BOLT-IOT

Bindupriya and others showed the development of an economical and efficient campus-automation system through Internet of Things (IOT) [23]. The proposed campus-automation system uses Bolt-IOT as cloud and Bolt ESP8266 based IoT platform as the Microcontroller and Android mobile application. Their system is similar to the way Smart Speakers are used. Their paper did not show the way of scheduled controlling of appliances.

6. Concluding Remarks

We can control the turning on/off of projectors, on the schedule written in the program of the class page. We can also turn projectors on/off any time we like. There was an instance when there was a mistake in the schedule. Projectors did not turn on at the time we expected. At that time, we could turn on the projectors from outside the building and correct the schedule immediately. We could use IR control signals for many kinds of appliances using the receiver function of our IR transceiver. So, we can control any appliances if they have IR remote control functions.

Unfortunately, there are no other cases of the use of the Wiki Bot, which controls appliances other than projectors, by the IR transceivers, until now. However, we have been using Wiki Bot for the wearable signage system, Teleport Dresser, for more than one year [24], [25]. The Teleport Dresser shows a picture or an animation on the wearable LED matrix display. Super imposing letters on the picture or the animation is also possible. The picture, the animation, or the letters can be changed by the script on the web page of the Wiki Bot. There was very little problem other than a few hard-wear troubles such as bad connections. There is a serious problem on Wiki IoT systems. A Wiki Bot freezes from time to time. During one month's display of the PR video of 400-th anniversary of the Fukuyama castle completion [17], there were a few freezes on our system. This might cause a fire in the building by heating up the lights of projectors when the system trouble has occurred while the projectors were on. We are dealing with this problem by a fail safe setting of projectors. The projectors are configured to turn off automatically when they were turning on over a specific time. In addition to this setting, we cope with this problem by setting that the program of the Wiki Bot starts automatically when the Raspberry Pi boots using systemd and reboots the Raspberry Pi at an interval shorter than the freeze interval using crontab. This setting made the Wiki Bot much more reliable. Our Wiki Bot with this setting could run more than one month after that.

In order to cope with troubles immediately after the troubles we experienced on our system, we would wish to know whether the projectors turned on or turned off after they received the command. This is possible by recording illuminance or temperature of the projector using Wiki Bot with illuminance sensors or temperature sensors [4].

In order to avoid the malicious operation of equipment by a third party, such as an unauthorized change of the main pages and class pages by others, we are using basic authentication of web pages [3], [4]. We know this is not enough and we should use https. We are changing the protocol of web servers of object pages and class pages from http to https. We also should adopt other security measures such as terminal authentication.

Acknowledgments A part of this research was supported by JSPS KAKENHI Grant Number JP16K00197, JP15H03055, JP21K11858. We also thank Professor Tanaka and students of Fukuyama university who helped us to develop the system.

References

[1] Timer, available from ⟨https://en.wikipedia.org/wiki/Timer⟩ (accessed 2021-05-29).
[2] Yamanoue, T.: Monitoring Servers, With a Little Help from my Bots, *Proc. 2017 ACM SIGUCCS Annual Conference On User Services* (Seattle, Washington), pp.173–180, Association for Computing Machinery, DOI: 10.1145/3123458.3123461 (2018).
[3] Yamanoue, T.: Bot Computing using the Power of Wiki Collaboration, *Proc. 2019 8th International Congress on Advanced Applied Informatics* (*IIAI-AAI*), pp.17–24, DOI: 10.1109/IIAI-AAI.2019.00015 (2019).
[4] Yamanoue, T.: 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, *Journal of Information Processing*, Vol.28, pp.204–213, DOI: 10.2197/ipsjjip.28.204 (2020).

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

[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*), *Advanced Applied Informatics* (*IIAIAAI*), pp.45–50, DOI: 10.1109/IIAI-AAI.2013.48 (2013).

[7] Gay, W.: Raspberry Pi Hardware Reference (1st ed.), *Apress* (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*) (2018).

[10] Yamanoue, T., Yokoyama, D., Umeda, R., Morita, S., Ozeki, T. and Nakamichi, N.: A Remotely Reconfigurable IoT System using Wiki Software, *Information Engineering Express*, Vol.5, No.2, pp.18–35 (2019).

[11] PukiWiki, available from ⟨https://en.wikipedia.org/wiki/PukiWiki⟩ (accessed 2021-05-29).

[12] 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, DOI: 10.1155/2012/981783 (2012).

[13] Bruegge, B. and Dutoit, A.A.: Object-Oriented Software Engineering; *Conquering Complex and Changing Systems*, Prentice Hall PTR (1999).

[14] Banzi, M.: Getting Started with Arduino, *Make Community*, *LLC* (2008).

[15] Ken Shirriff: Arduino-IRremote (2016), available from ⟨https://github.com/z3t0/Arduino-IRremote⟩.

[16] Yamanoue, T., Tetaka, S., Oda, K. and Shimozono, K.: Portable Cloud Computing System: A System which Makes Everywhere an ICT Enhanced Classroom, *Proc. 42nd Annual ACM SIGUCCS Conference on User Service*, pp.85–88 (2014).

[17] Turning windows into media (in Japanese), available from ⟨https://www.fukuyama-u.ac.jp/blog/17522/⟩ (accessed 2021-09-12).

[18] SoftEther VPN (in Japanese), available from ⟨https://ja.softether.org⟩ (accessed 2021-05-29).

[19] IFTTT, available from ⟨https://ifttt.com⟩ (accessed 2021-05-29).

[20] Smart Speaker, available from ⟨https://en.wikipedia.org/wiki/Smart_speaker⟩ (accessed 2021-05-29).

[21] obniz, available from ⟨https://obniz.com⟩ (accessed 2021-05-29).

[22] Jain, M., Kaushik, N. and Jayavel, K.: Building automation and energy control using IoT – Smart campus, *2017 2nd International Conference on Computing and Communications Technologies* (*ICCCT*), pp.353–359 (2017).

[23] Kumar, V., Vishnuvardhan, Subash, Bindupriya, Theerthagiri, P.: A Cost Effective Campus Automation System Using BOLT-IOT, *International Journal of Control and Automation*, Vol.13, No.4, pp.935–941 (2020), available from ⟨http://sersc.org/journals/index.php/IJCA/article/view/18893⟩.

[24] Teleport Dresser, available from ⟨https://protopedia.net/prototype/1812⟩ (accessed 2021-09-05).

[25] Yamanoue, T.: Yet Another Wearable LED Matrix Sign System for Campus Guiding, *Proc. 2022 ACM SIGUCCS Annual Conference On User Serviices* (2022).

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 chair of the graduate school and a professor at 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, IPSJ, The Institute of Electronics, IEICE, JRSJ. He is also a member of the ACM SIGUCCS Hall of Fame.