Information Engineering Express
International Institute of Applied Informatics
2019, Vol. 5, No. 2, 18 – 35

A Remotely Reconfigurable IoT System using Wiki Software

Takashi Yamanoue [*], Daichi Yokoyama [†], Ryoya Umeda [*],
Shota Morita [*], Takashi Ozeki [*], Noboru Nakamichi [*‡]

Abstract

Herein, an Internet of Things (IoT) system with remotely reconfigurable gateways and wireless sensor network (WSN) nodes is presented and its applications are discussed. This IoT system consists of wiki pages, bots, (defined as remotely controlled devices), and remotely reconfigurable WSN Nodes. Each bot can be used as the gateway of the WSN if the bot was connected with it. The bots/gateways and WSN nodes of the IoT system are controlled by scripts written on wiki pages that are read by the bot/gateways, which then forward the commands contained in the scripts to the WSN nodes. Each WSN node then executes the commands, obtains the resulting sensor values, and then returns those values to the bot/gateways. The bot/gateways relay the values and write them onto the wiki page. Each bot and WSN node is flexible and reconfigurable because its behavior can be changed by the script commands written onto the wiki page. Using this IoT system, we created a measurement system to evaluate activity levels of group work in a classroom. After developing the measurement system, we found it was possible to adjust the sampling rate for obtaining sensor values simply by rewriting the script on the wiki page without directly manipulating the WSN.

Keywords: Sensor Network, IoT, Bot computing, Wiki.

1 Introduction

A typical Internet of Things (IoT) system consists of Internet servers, Internet edge gateways, and devices with sensors/actuators that are connected to those gateways by wires or wireless networks. When the number of components making up an IoT system becomes very large, powerful managers must be used to administer them effectively. This issue is complicated by the fact that sensor/actuator-equipped devices may be placed in a wide variety of natural areas such as mountains, fields, rivers, the deep sea, and even outer space. Alternatively, they can be found in urban settings, such as the tops of buildings and towers, inside buildings, or even within small spaces inside machines. As a result, it is often hard, and sometimes even impossible, for a manager to directly monitor and operate such devices. Nevertheless, IoT system managers still

[*] Fukuyama University, Fukuyama, Japan
[†] Graduate School of Engineering, Fukuyama University, Fukuyama, Japan
[‡] ANKR DESIGN Inc., Tokyo, Japan

need the ability to change the configurations of such devices when it is necessary to alter sensor sensitivities, change sampling data frequencies, turn on/off devices, and so on.

A wiki [20] page is a website that allows the easy creation and editing of any number of interlinked webpages via a web browser, and can be used as a means of effective collaboration and information sharing. Wikipedia [21] is a well-known wiki site. If a wiki is friendly to people, it will also be friendly to machines. This means that if machines can automatically read and write data onto a wiki page, users can obtain much more in the way of beneficial information. Those users can also easily control machines through the wiki page, thereby facilitating machine-to-people, people-to-machine, as well as machine-to-machine communication, which would make the wiki much more useful. For example, if a well-known wiki could be used to connect sensors in a sensor network, individualized sensor networks can be created much easier (Figure 1).

To confirm the above presumptions regarding the usefulness of wikis, we developed an IoT system using wiki software [5][6][7][8][10]. This system consists of wiki pages using wiki software and bots (defined herein as remotely controlled devices). This IoT system may have wireless sensor network (WSN) nodes [9]. The bots used in our IoT system are flexible and reconfigurable because they are controlled by the script which is consists of commands and a program, written on wiki pages. The bot reads the script from the wiki page and then executes them periodically. WSN nodes are also flexible and reconfigurable because their behavior can be changed remotely simply by revising the script on the wiki page.

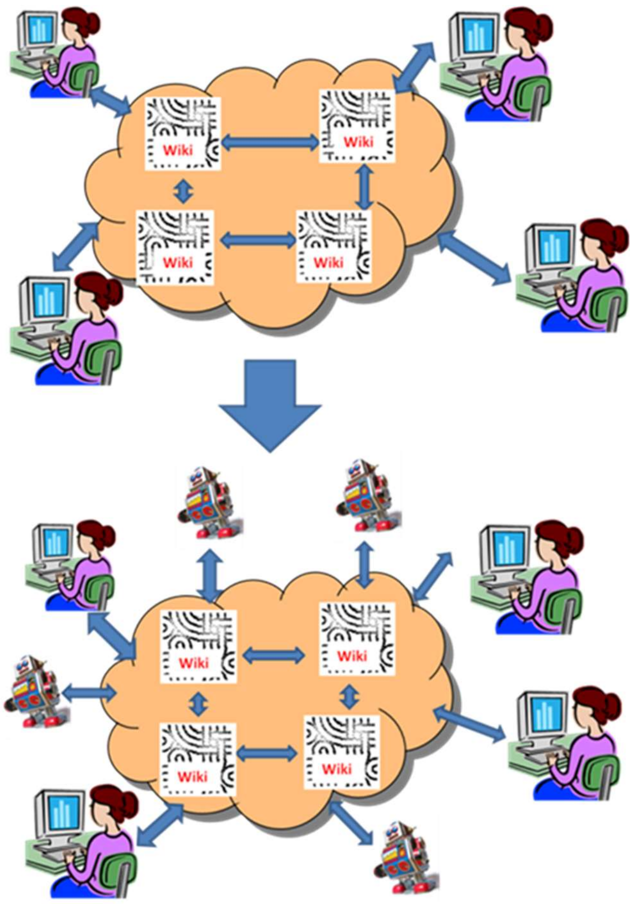


Figure 1: Wiki for people to Wiki for people and machines

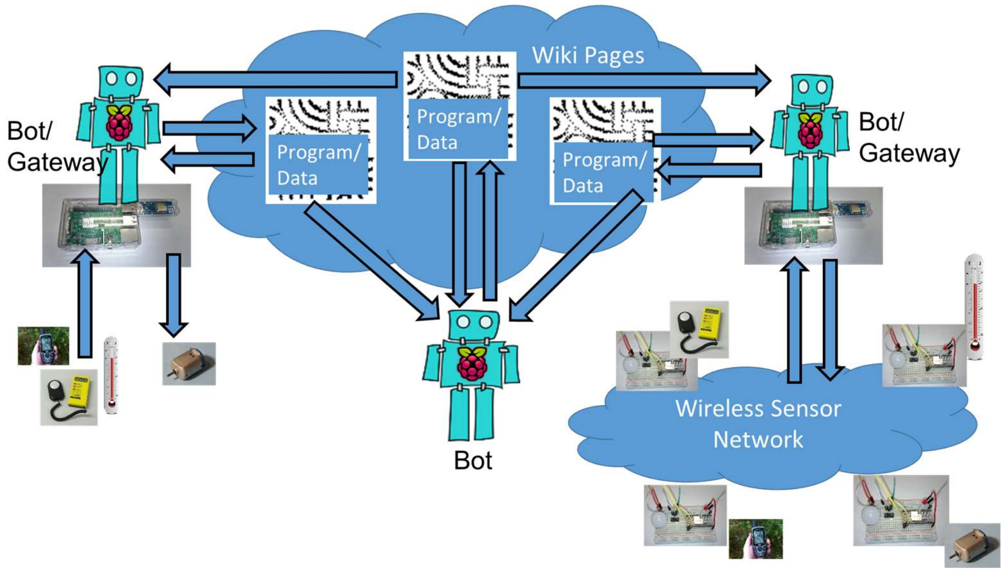


Figure 2: Our IoT system overview.

By combining flexible bots and flexible WSN nodes, IoT system managers can configure, maintain, and reconfigure many aspects of a system, thereby identifying issues and eliminating problems simply by rewriting the system script on wiki pages, without the need to physically access the bots and WSN nodes. In the next section, we will explain how we created an activity measurement system to evaluate group work activity in a classroom using our proposed IoT system. When developing our measurement system, we focused on ensuring it was possible to alter sensor sensitivities, change sampling data frequencies, turn on/off devices, and so on, simply by rewriting the command sequence and the program on the wiki page without directly manipulating the WSNs.

2 Reconfigurable Iot System

A. Structure of the reconfigurable IoT system

Figure 2 shows an overview of our reconfigurable IoT system, which consists of wiki pages, wiki software, bots/gateways, and WSN nodes. PukiWiki [16] was used for the wiki software, and Raspberry Pi computers [17] and ZigBee [24] transceiver modules were used to connect the bot/gateways and WSN nodes. Here, it should be noted that the ZigBee devices are programmable and equipped with sensors and actuators.

In operation, if the bot/gateway has a WSN, the bot/gateway reads/writes data from/to the ZigBee devices, which are connected by the ZigBee WSN. These collect and send sensor data to the wiki page based on the script written on that page. The wiki page can also include a bot program that reads data from Internet webpages and WSN nodes can be placed anywhere the node and WSN bot/gateway can establish and maintain communication. Furthermore, since all the WSN nodes are ZigBee network components and can route data to other WSN nodes, it is possible to place nodes in locations where it would be impossible for them to communicate directly with the bot/gateway.

If a WSN node is equipped with actuators, they can be operated by the bot/gateways based on the PukiWiki webpage script.

There are some bots without the WSN gateway function. Some of these bots are connected to sensors and actuators directly. Some other bots can be used to analyze data on other wiki pages (or other websites) and report back the analysis results to the originating wiki page. These bots can run on any personal computer (PC) that can execute the required software.

B. Bot/Gateways

Since the bot/gateways of our IoT system are script interpreters, bot behavior is defined by the script. A wiki page script consists of a command sequence and a program. The bot hardware consists of a Raspberry Pi combined with a MONOSTICK [14] ZigBee transceiver module (Figure 3). The bot/gateway communicates with wiki software via the Internet using a local area network (LAN) interface or the Wi-Fi transceiver of the Raspberry Pi. In operation, the bot controls and reads sensor data from WSN nodes using the ZigBee transceiver, but our IoT system can also support bots without the transceiver. Some of them have sensors and/or actuators which are connected with the bots directly and they can input data from sensors and/or can control actuators. Other bots are used to analyze data written on other wiki pages in our IoT system (or other web pages, and then write the results on a wiki page where they can be read by users and other IoT system bots/gateways). Some of bots are equipped with R language processor in order to perform this analysis. The R program for this analysis is also included in the script written on the wiki page [2].

Figure 4 shows the behavior of the bot/gateway. After starting, the bot/gateway repeats the following sequence:

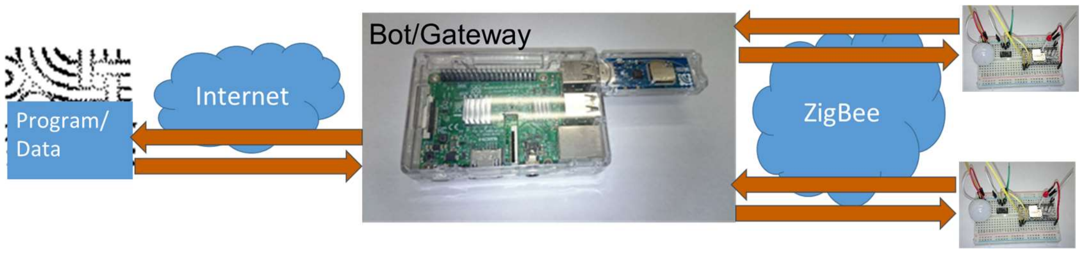


Figure 3: Bot/Gateway overview.

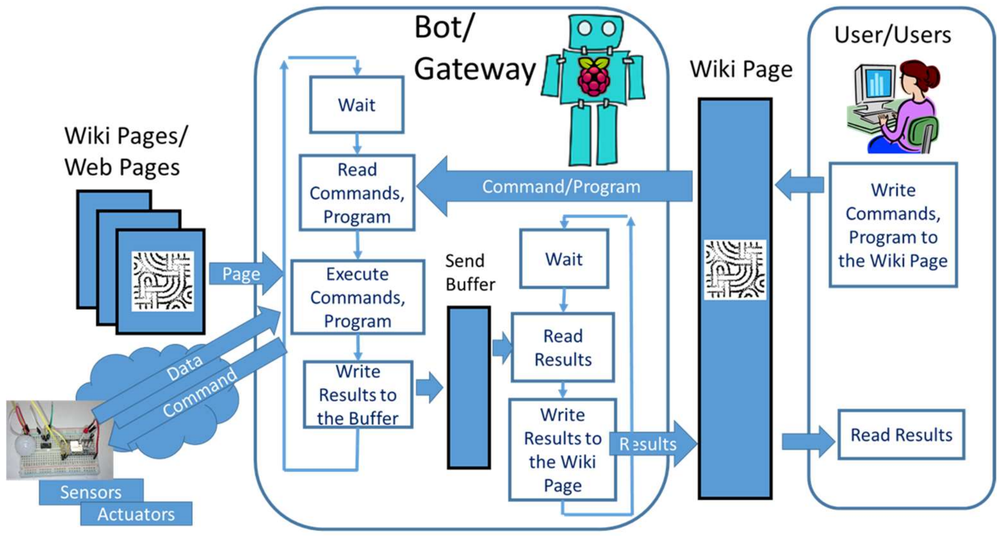
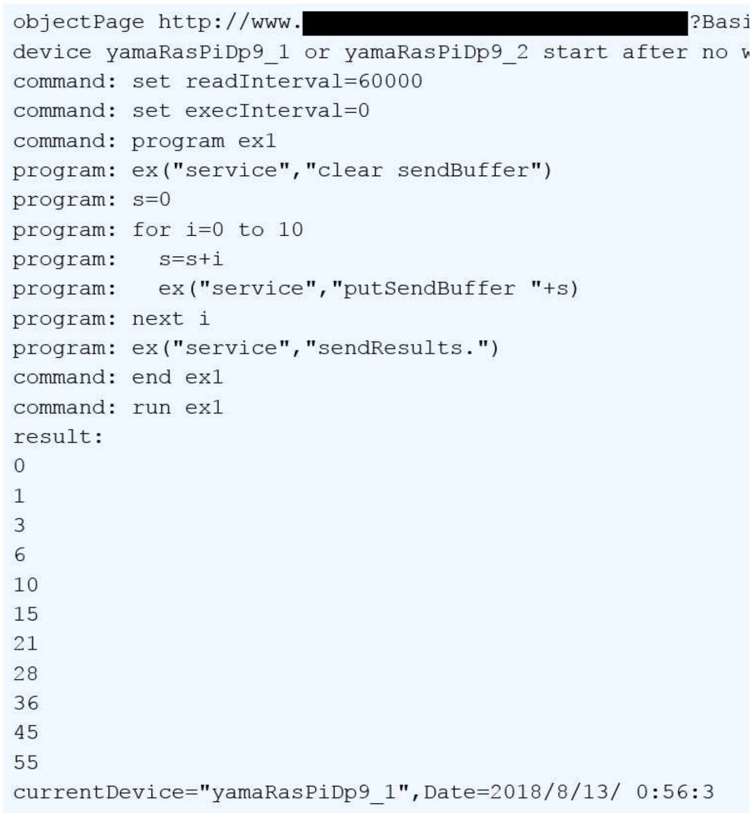


Figure 4: Bot/Gateway behavior.

1. Read the command sequence and program on the wiki page. The URL of the wiki page is written in the bot setting file or input via the bot graphical user interface (GUI).
2. Interpret and execute the command sequence and program.
3. After executing the command sequence lines and program, write back the execution results to the wiki page.

   Figure 5 shows an example of the program and command sequence for the bot on a wiki page. This script directs the bot to do the following:

- Repeat reading of this page at intervals of one hour by the command line of *"command: set readInterval=60000"*
- Define the program by the following lines.
- *command: program ex1*
- *program: ...*
- …
- *program: ...*
- *command: end ex1*
- These lines show that the program, between the lines of "*command: program ex1*" and "*command: end ex1*", is stored in the bot under the name of "*ex1*". The lines of the program are denoted by lines that start with the word "*program:*". The program computes the sum from one to ten.
- Execute the program ex with the line "*command: run ex1*"



```
objectPage http://www.[redacted]?Basi
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no w
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
```

Figure 5: Example of a program embedded within a series of commands.

When the data of one wiki page is very large, it is hard for users to read as well as hard for the bot to read and write. Partitioning provides an effective way to cope with this problem. In order to partition the data to several wiki pages, the command "*set pageName="<next page name>"*" can be used. When the bot reads this command, the wiki software then reads the <*next page name*> wiki page during the next read, instead of current page. The <*next page name*> may also have <*hour*> or <*day*> commands. In such cases, <hour> is substituted for the current hour of execution and <*day*> is substituted for the current day of the execution. For example, if the "*command: set pageName="pir-h-<hour>"*" exists in the wiki page, and there are corresponding pages (named "pir-h-0" to "pir-h-23") for each hour in the wiki software, the bot changes the wiki page to read every hour.

C. *Sharing the Common Script and Facilitating High Availability*

Some bots in the IoT system use the same script. However, since it is troublesome to write the same commands and program to each wiki page, as in object-oriented programming, the IoT system might also have a class page for sharing common script among the wiki pages of such objects. A wiki page consisting of objects is called an object wiki page. An object page uses the "include" command for a class wiki page when sharing the common class among object pages.

Our IoT system also facilitates high availability by introducing "crossover including" to Wiki pages and "crossover execution" to bots. Figure 6 shows the sharing common script by include commands and facilitating high availability by cross over including.

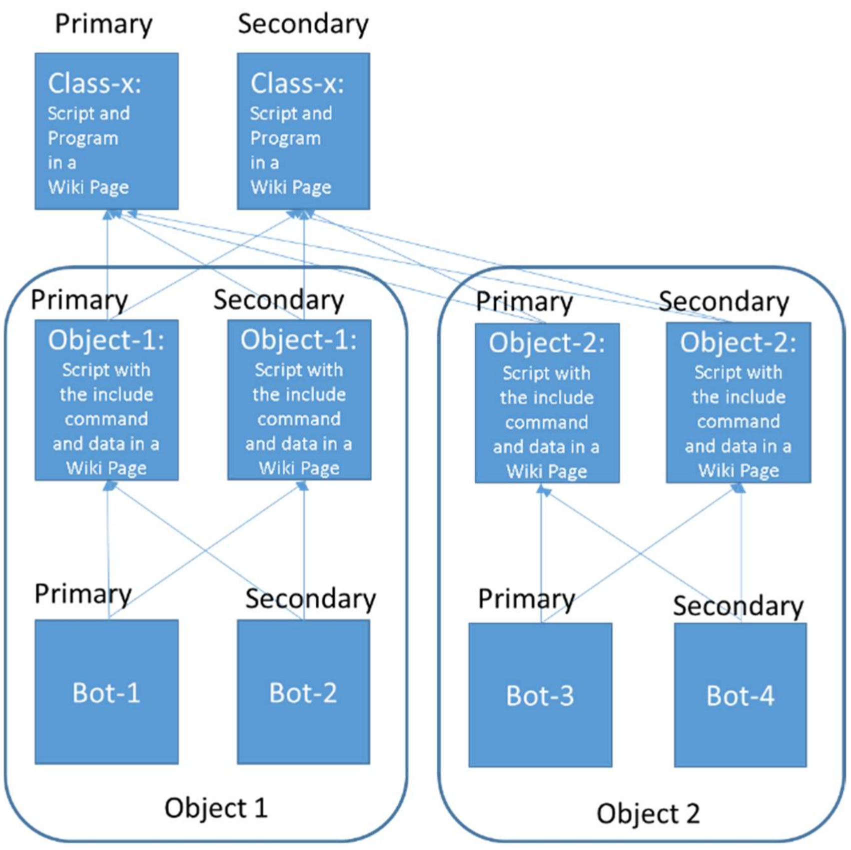

Primary
Secondary
Class-x: Script and Program in a Wiki Page
Class-x: Script and Program in a Wiki Page
Primary
Secondary
Object-1: Script with the include command and data in a Wiki Page
Object-1: Script with the include command and data in a Wiki Page
Primary
Secondary
Object-2: Script with the include command and data in a Wiki Page
Object-2: Script with the include command and data in a Wiki Page
Primary
Secondary
Bot-1
Bot-2
Primary
Secondary
Bot-3
Bot-4
Object 1
Object 2

Figure 6: Sharing the Common Script and Facilitating High Availability.

D. Message Passing between Internal Objects

A bot has internal objects such as ”service” object, “connector” object, “pi4j” object, “twitter” object, and the object of the program of the script. The “service” object is the main internal object of the bot. The script for the bot is interpreted by this object. This internal object also provides functions for manipulating the send buffer, the content of which will be written on the result part of the object wiki page, setting interval times of reading, writing, and so on. The “connector” object is in charge of communication between the bot and the wiki page. The “pi4j” object is the wrapper object of the GPIO and the serial interface of Raspberry Pi. The “twitter” object is in charge of communication between the bot and Twitter.

An internal object can be used by the program of the script by using the follow program function:

ex(<internal object name>, "<message to the object>")

The <*internal object name*> can be “service”, “connector”, “pi4j”, “twitter”, and the name of the program. The <*message to the object*> is the message which can be interpreted by the internal object.

For example, the following line of the Figure 5 shows that “clear” the send buffer of the “service” object.

ex("service","clear sendBuffer")

The following line shows that the value of the variable “*s*” is added to the last line of the send buffer of the “service” object. The command *“putSendBuffer <string>”* of the “service” object add the <*string*> to the send buffer.

ex(“service”, “putSendBuffer “+s)

The following line shows that the lines in the send buffer are sent to the Wiki software and they are written after the “result:” line of the object wiki page.

ex("service","sendResults.")

E. Commands for I2C interface

Raspberry Pi has an I2C interface as the GPIO facility of Pi. I2C is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus invented in 1982 by Philips Semiconductor (now NXP Semiconductors). It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication [13]. Many kinds of many I2C devices, such as sensors and actuators, can be attached to Raspberry Pi using the single I2C interface. It is possible to change settings of such devices by issuing commands through the I2C interface.

Commands for manipulating I2C devices through I2C interface can be used in the script written on the wiki page and the bot interprets these commands and manipulates devices which are connected to the Raspberry Pi using its I2C interface.

The bot starts to use the I2C interface by using the following program function:

ex("pi4j", "i2c use 1")

The following program function writes the value <*v*> to the register <*r*> of the I2C slave device, the address of which is <*addr*>:

ex(“pi4j”, “i2c write1 <addr>, <r>,<v>”)

<*addr*>is a 7bit value in hexadecimal notation. <*r*>,<*v*> are 8bit value in hexadecimal no-tation.

The following assign statement reads the value of the register <*r*> of the I2C slave device, the address of which is <*addr*>, by the function *ex(“pi4j”, “i2c read1 <addr>,<r>”*) and assign the value to the variable “v”.

v=ex(“pi4j”, “i2c read1 <addr>,<r>”)

F. Commands for WSN nodes

The bot/gateway sends commands to the WSN nodes using the following program function:

ex(“pi4j”, “serial send \” <command for the WSN node>\”.”)

When the WSN node receives a command, it executes the command and sends the results back to the bot/gateway based on the conditions outlined in the command. The results appear in the following format:

return a32=<receiver-id>, from=<sender-id>,
port=<analog/digital-input-port>, v=<value>,
event=<cause of the return>

If the *<receiver-id>* of the bot/gateway receiving the results matches the bot/gateway’s ZigBee transceiver module, the bot/gateway relays the results and writes the following back to the wiki page:

device=sensorNetwork, Date=<date and time of received>,
a32=<receiver-id>, from=<sender-id>,
port=<analog/digital-input-port>, v=<value>,
event=<cause of the return>

The Universal Serial Bus (USB) serial interface and Pi4J are used for communication between the transceiver module and the Raspberry Pi.

G. Wireless Sensor Network Node

In our network, TWELITE DIP [19] programmable ZigBee devices, which are equipped with sensors and actuators, are used for WSN nodes. We programmed the command interpreter to use the WSN nodes, a sample of which is shown in Figure 7. The dual inline package (DIP) shown on the right side of the figure is the ZigBee device, whereas the white spherical part is a passive infra-red (PIR) sensor that can detect the motion of a human body or animal. Since our IoT system WSN is a ZigBee network, WSN nodes can be placed in locations where they would normally be unable to communicate directly with the bot/gateway by taking advantage of ZigBee router nodes.

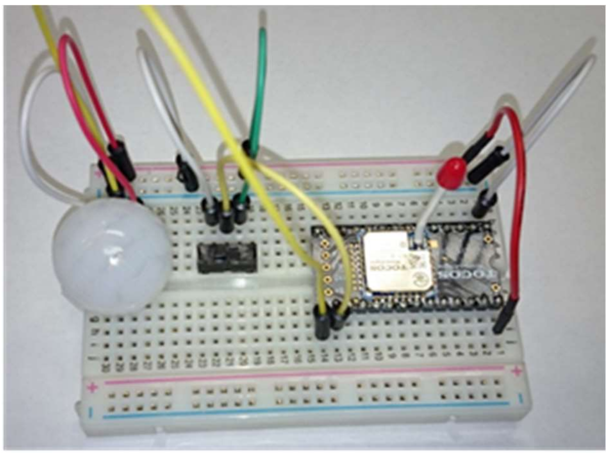


Figure 7: Sample WSN node.

Communications between all WSN nodes is realized by broadcasting. Furthermore, since WSN nodes can interpret and execute received commands, WSN node behavior can be changed by command and parameter revisions. The following are examples of WSN node commands:

- *get <id> port <analog/digital input port>.*

When a node with the *<id>* node identifier receives this command, the value at the *<analog/digital input port>* is immediately returned to the bot/gateway. The *<id>* is given in the fol-lowing format.

<id>=| a32=<hex> | id=<number> | handle=<strconst>*

The * means all nodes in the WSN except for the transmitter node of the command. The *<hex>*is the 8-byte identifier of the terminal in the WSN given in the hexadecimal format. The *<number>* is a 2-byte logical terminal identifier in the WSN. The *<strconst>* is the name of the node. For example, the command,

get id=1 port DI1

means if the logical identifier of the node matches the one in the command, the node should immediately return the digital value (zero or one) at digital input port one with “get” as the *<cause of the return>*.

- *set <id> sendif <analog/digital input port> interval <time-msec>.*

When a node with the *<id>* node identifier receives this command, the node repeatedly returns the value at the *<analog/digital input port>* at intervals of *<time-msec>*, with “interval” as the *<cause of the return>*.

- *set <id> sendif <analog/digital input port> changed <value>.*

When a node with the *<id>* node identifier receives this command and the value at the *<analog/digital input port>* has been changed to exceed the *<value>* compared with the previous *<value>* at 0.5 sec intervals, the node returns the current port value with “changed” as the *<cause of the return>*. The WSN node has an internal status for mode. When the *<analog/digital input port>* is a digital input port and the value of the mode is one, the value at the port after the change is returned to the bot/gateway.

When the *<analog/digital input port>* is a digital input port and the value of the mode is two, the value at the port accumulates at 0.5 second intervals and is returned to the bot/gateway when the accumulated value exceeds the changed *<value>*, after which the registered accumulation is cleared.

When the *<analog/digital input port>* is an analog input port and the value of the mode is one, the value at the port after the change is returned to the bot/gateway.

When the *<analog/digital input port>* is an analog input port and the value of the mode is two, the value at the port accumulates at 0.5 sec intervals and is returned to the bot/gateway when the accumulated value exceeds or drops below the changed *<value>*, after which the registered accumulation is cleared.

- *set <id> sendif <analog/digital input port> sendAccumulation <time-msec>.*

When a node with the *<id>* node identifier receives this command, the value at the *<analog/digital input port>* accumulates during the intervals of *<time-msec>* and the accumulated value is returned to the bot/gateway with "sendAccumulation" as the *<cause of the return>*, after which the registered accumulation is returned. The sampling interval at the port is 0.5 sec.

- *set <id> sendif <analog/digital input port> setmode <mode>.*

When a node with the *<id>* node identifier receives this command, set the status of mode to the *<mode>*.

- *set <id> i2c <address in hex> <register in hex> <data-size> <value in hex>.*

When a node with the *<id>* node identifier receives this command, send the *<value in hex>* to the register *<register in hex>* of the I2C device along with the I2C address of the *<address in hex>*. The size of the value is *<data-size>*. The I2C device is connected to the I2C port of the ZigBee device through the I2C bus.

- *get <id> i2c <address in hex> <register in hex> <data-size>.*

When a node with the *<id>* node identifier receives this command, return the value in the register *<register in hex>* of the I2C device to the bot/gateway with"i2c" as the *<cause of the return>*

3 Demonstration of IoT System Reconfiguration

A. Reconfiguration of remote illuminance measuring system

We have made a remote illuminance measuring system using our IoT system. This system measures illuminance of the place where the wiki bot with a light sensor, and shows the current value of illuminance on the wiki page of the system. We use the TSL2561 for the light sensor.

The TSL2561 I2C light sensor is an precise, allowing for exact Lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 0.1 - 40,000+ Lux on the fly. This sensor contains both infrared and full spectrum diodes. The user can separately measure infrared, full-spectrum or human-visible light. The user can have either a gain of 0 (no extra gain, good in low light situations) or a gain of 16 which will boost the light considerably in dim situations. The user can also change the integration time, which is how long it will collect light data for. The longer the integration time, the more precision the sensor has when collecting light samples [11][18].

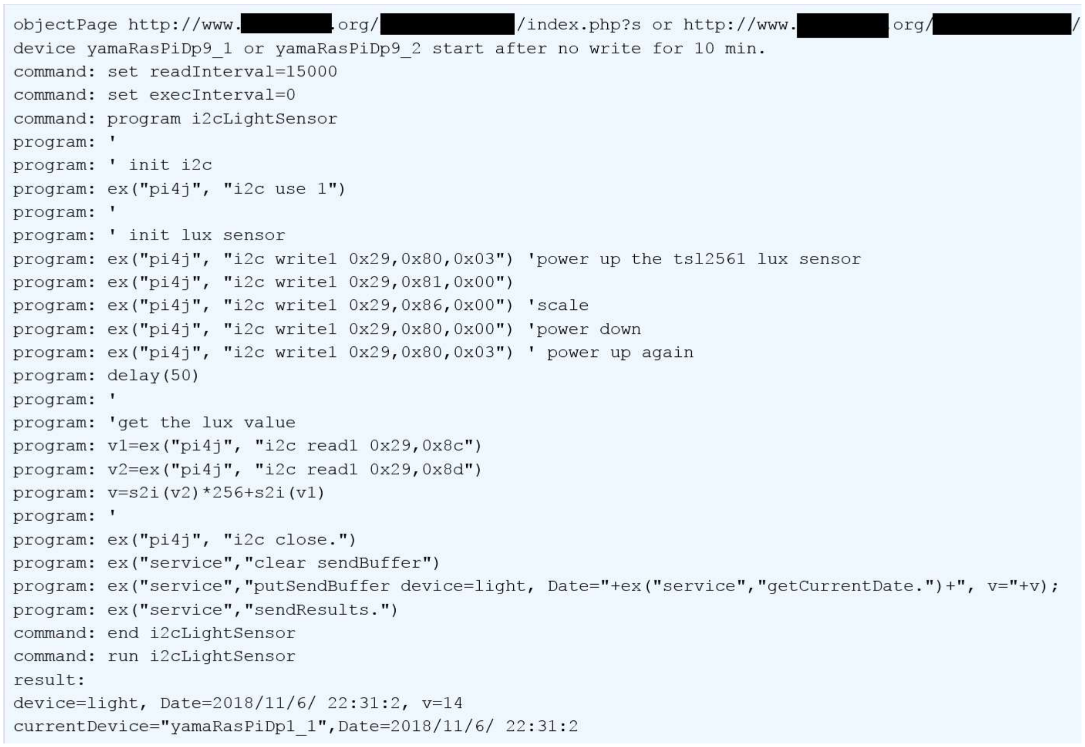

```
objectPage http://www.[redacted].org/[redacted]/index.php?s or http://www.[redacted].org/[redacted]/
device yamaRasPiDp9_1 or yamaRasPiDp9_2 start after no write for 10 min.
command: set readInterval=15000
command: set execInterval=0
command: program i2cLightSensor
program: '
program: ' init i2c
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: v=s2i(v2)*256+s2i(v1)
program: '
program: ex("pi4j", "i2c close.")
program: ex("service","clear sendBuffer")
program: ex("service","putSendBuffer device=light, Date="+ex("service","getCurrentDate.")+", v="+v);
program: ex("service","sendResults.")
command: end i2cLightSensor
command: run i2cLightSensor
result:
device=light, Date=2018/11/6/ 22:31:2, v=14
currentDevice="yamaRasPiDp1_1",Date=2018/11/6/ 22:31:2
```
Figure 8: Script of the remote illuminance measuring system which measures the spectrum from infrared to visible light.

Figure 8 shows the script of the remote illuminance measuring system which measures the spectrum from infrared to visible light. The value of illuminance is shown at the end of the second line from the bottom. It is 14 in this case.

In this figure, the line *ex("pi4j", "i2c write1 0x29,0x80,0x03")* shows that write 0x03 to the control register (*0*h) of the TSL2561 device(*0x29*). The TSL2561 has 16 registers and they are specified by the 8bit command register. In this case, *0x80* means “select the control register *0*h”. The *8* of *0x80* means set CMD bit of the command register. When the value *0x03* were written to the control register, the power of the TSL2561 is up. The line *ex("pi4j", "i2c write1 0x29,0x81,0x00")* shows that writing *0x00* to the timing register(*1*h) of the TSL2561 device(*0x29*). In this case, *0x00* means “set low gain, stop an integration cycle, set *0* to integrate time. The line *ex("pi4j", "i2c write1 0x29,0x86,0x00")* shows that writing *0x00* to the interrupt register(*6*h) of the TSL2561 device(*0x29*). This operation disables interrupt operation. The line *ex("pi4j", "i2c write1 0x29,0x80,0x00")* shows that writing *0x00* to the control register(*0*h) of the TSL2561 device(*0x29*). This operation power down the TSL2561. The line *ex("pi4j", "i2c write1 0x29,0x80,0x03")* shows that operation of power up of the TSL2561 device(*0x29*). The line *delay(50)* shows that waiting 50 milliseconds. The line *v1=ex("pi4j", "i2c read1 0x29,0x8c")* shows that reading the value of the low byte of ADC channel 0 (register *C*h) and assigning the value to the variable v1. The return value of the function *ex* is in string type. The line *v2=ex("pi4j", "i2c read1 0x29,0x8d")* shows that reading the value of the high byte of ADC channel 0 (register *D*h) and assign the value to the variable v2. Registers for channel 0 hold the value of full spectrum illuminance. The line *v=s2i(v2)*256+s2i(v1)* shows that computing the value of ADC channel 0. The function *s2i* transform the value in string type to the value in integer type.

When the user want to obtain infrared illuminance instead of full spectrum illuminance, this is possible by rewriting the following part of Figure 6

```
program: 'get the lux value
program: v1=ex("pi4j", "i2c read1 0x29,0x8c")
program: v2=ex("pi4j", "i2c read1 0x29,0x8d")
program: v=s2i(v2)*256+s2i(v1)
```

by the following.

```
program: 'get the ir lux value
program: v1=ex("pi4j", "i2c read1 0x29,0x8e")
program: v2=ex("pi4j", "i2c read1 0x29,0x8f")
program: v=s2i(v2)*256+s2i(v1)
```

This reconfiguration can be realized by the user without going to the place where the wiki bot and the sensor is, from any place in the world.

B. Reconfiguration of Activity Measurement System.

As shown in Section two, bots and WSN nodes are both controlled by the wiki page script. This means the manager can change IoT system behavior by rewriting the script without going to where the bots and WSN nodes are located. The manager does not even need to stop the bots and WSN nodes of the IoT system while script rewriting is in progress because the wiki software's mutual exclusion function prevents inconsistent updating of command sequences and programs on wiki pages.

As an example of the usefulness of this flexibility, we show an activity measurement system that measures activity of group work in a classroom using our IoT system.

On Nov. 20, 2014, The Central Council of the Educational Committee of Japan reported the adoption of a new direction for the twenty-first century that focused on ability as a key competency, as well as the International Baccalaureate curriculum, and active learning [1].

We are now in the process of confirming the effectiveness of active learning. However, in order to evaluate it effectively, a benchmark of the "activeness" of active learning is needed. Accordingly, in an attempt to measure the physical motions of participants in a group work as a candidate benchmark such "activeness", we adopted PIR motion sensors, which are commonly used in home security systems, automatic lighting control systems, and so on.

More specifically, we developed an experimental activity measurement system by combining our reconfigurable IoT system and PIR sensors. Figure. 9 shows an outline of our combined system being used to measure the group work activity of individual groups in a classroom. The system consists of Internet-based wiki pages, a bot/gateway, and eight WSN nodes. We created 24 wiki object pages for storing hourly activity data and one wiki class page for storing the script used for controlling the bot and WSN nodes. Figure 10 shows an example of the WSN nodes that were placed at the center of the table for each group in the classroom.

We then measured the activity of group work in a classroom using our measurement system and compared activity transitions with the video taken during the class in order to determine the relationships between them. If motion within a group increased, the value extracted from the measurement system became larger. During the measurement system development process, we were able to improve the system by reconfiguring WSN nodes without manipulating them directly, simply by rewriting the script on the class wiki page.

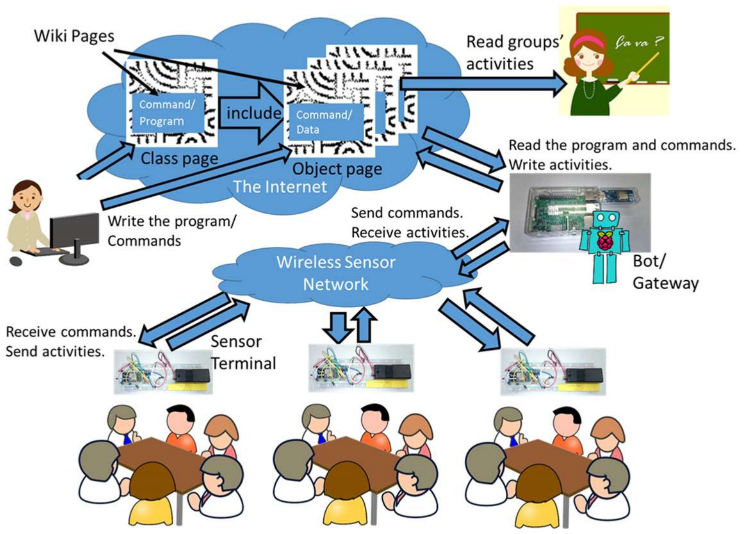


Figure 9: Measurement system outline

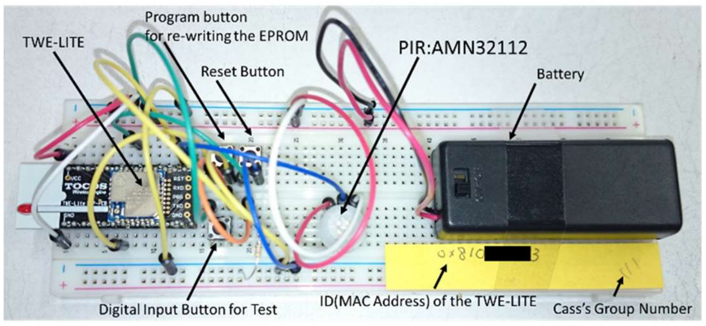
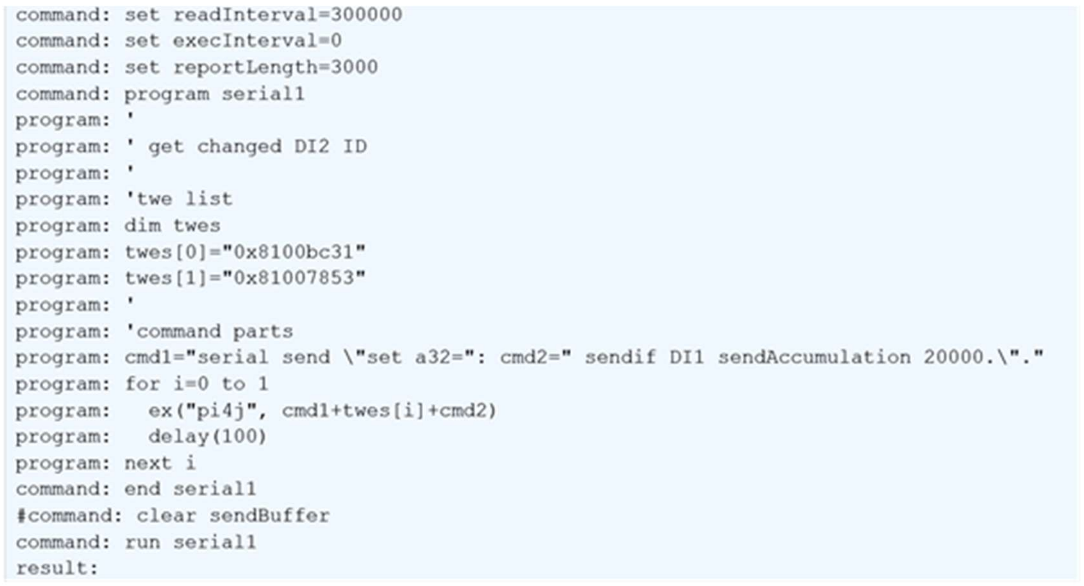

TWE-LITE
Program button for re-writing the EPROM
Reset Button
PIR:AMN32112
Battery
[redacted]
Digital Input Button for Test
ID(MAC Address) of the TWE-LITE
Cass's Group Number

Figure 10: Sample WSN node.

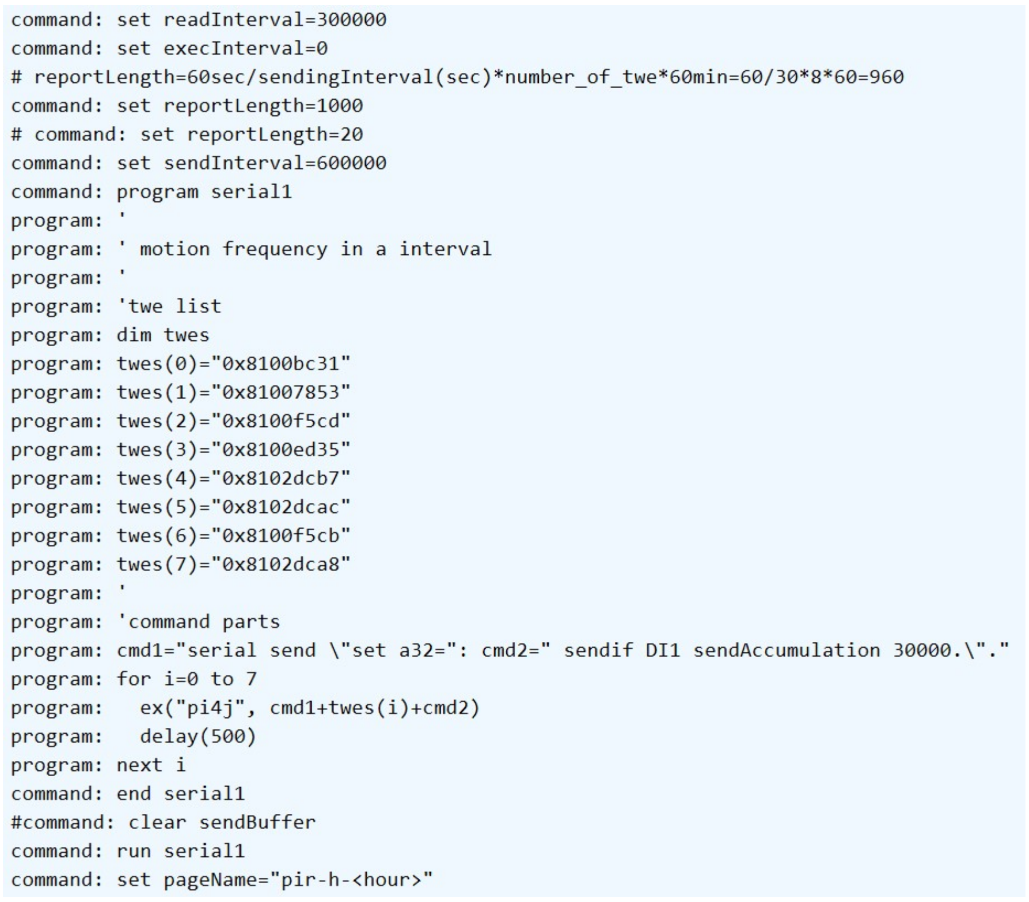
Figure 11 shows the class wiki page of the measurement system before the improvement. This figure shows that when the bot sent the "sendif D11 sendAccumulation 2000" command to two of the WSN nodes of the measurement system at five-minute intervals (300000 msec), two of the WSN nodes return accumulated values to the DI1 digital port at 20 sec (20000 msec) intervals. When eight WSN nodes were use, 1440 result lines were collected each hour. However, this made the measurement system unstable because adding new data lines to such a large text block consumes significant amounts of time. The line, "command: set reportLength=3000" in Figure 9 indicates the setting for the max number of lines after the "results:" line in the wiki page. However, 3000 was also too large for the stable execution of the command sequence and the program. More specifically, if there is no send interval setting command for the page, the bot will attempt to write the buffered results to the wiki page immediately after executing the command sequence and program. This was also found to cause unstable execution.

Figure 12 shows the improved class wiki page for the measurement system. As can be seen in the figure, we rewrote the "sendif D11 sendAccumulation 2000" command to "sendif D11 sendAccumulation 3000" in order to the reduce the hourly number of results lines from 1440 to 960, set the reportLength to 1000, and set the write back intervals to ten minutes. We also added the "set pageName="pir-1-<hour>"" command at the end of the command sequence in order to partition the results lines into hourly pages. It can be seen that these improvements made the measurement system more stable.



```
command: set readInterval=300000
command: set execInterval=0
command: set reportLength=3000
command: program serial1
program: '
program: ' get changed DI2 ID
program: '
program: 'twe list
program: dim twes
program: twes[0]="0x8100bc31"
program: twes[1]="0x81007853"
program: '
program: 'command parts
program: cmd1="serial send \"set a32=": cmd2=" sendif DI1 sendAccumulation 20000.\"."
program: for i=0 to 1
program:   ex("pi4j", cmd1+twes[i]+cmd2)
program:   delay(100)
program: next i
command: end serial1
#command: clear sendBuffer
command: run serial1
result:
```

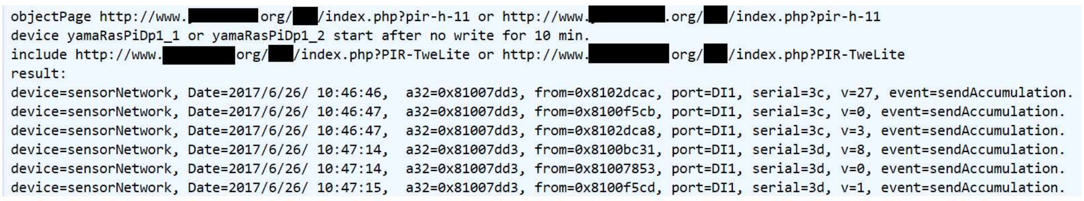
Figure 11: Sequence of commands and the program on the wiki object page of the measurement system. (Before improvement.)



```
command: set readInterval=300000
command: set execInterval=0
# reportLength=60sec/sendingInterval(sec)*number_of_twe*60min=60/30*8*60=960
command: set reportLength=1000
# command: set reportLength=20
command: set sendInterval=600000
command: program serial1
program: '
program: ' motion frequency in a interval
program: '
program: 'twe list
program: dim twes
program: twes(0)="0x8100bc31"
program: twes(1)="0x81007853"
program: twes(2)="0x8100f5cd"
program: twes(3)="0x8100ed35"
program: twes(4)="0x8102dcb7"
program: twes(5)="0x8102dcac"
program: twes(6)="0x8100f5cb"
program: twes(7)="0x8102dca8"
program: '
program: 'command parts
program: cmd1="serial send \"set a32=": cmd2=" sendif DI1 sendAccumulation 30000.\"."
program: for i=0 to 7
program:   ex("pi4j", cmd1+twes(i)+cmd2)
program:   delay(500)
program: next i
command: end serial1
#command: clear sendBuffer
command: run serial1
command: set pageName="pir-h-<hour>"
```

Figure 12: Command sequence and program on the wiki object page of the measurement system. (After improvement.)

Figure 13 shows the execution results on the object wiki page after the improvements. Here, each line of the figure shows the returned information from a WSN node, the identifier of which is the right-hand side of the equation of “a32=”. This value, which shows the activity at each work group, is the right-hand side of the “v=” equation.



```
objectPage http://www.[redacted]org/[redacted]/index.php?pir-h-11 or http://www.[redacted].org/[redacted]/index.php?pir-h-11
device yamaRasPiDp1_1 or yamaRasPiDp1_2 start after no write for 10 min.
include http://www.[redacted]org/[redacted]/index.php?PIR-TweLite or http://www.[redacted]org/[redacted]/index.php?PIR-TweLite
result:
device=sensorNetwork, Date=2017/6/26/ 10:46:46,  a32=0x81007dd3, from=0x8102dcac, port=DI1, serial=3c, v=27, event=sendAccumulation.
device=sensorNetwork, Date=2017/6/26/ 10:46:47,  a32=0x81007dd3, from=0x8100f5cb, port=DI1, serial=3c, v=0, event=sendAccumulation.
device=sensorNetwork, Date=2017/6/26/ 10:46:47,  a32=0x81007dd3, from=0x8102dca8, port=DI1, serial=3c, v=3, event=sendAccumulation.
device=sensorNetwork, Date=2017/6/26/ 10:47:14,  a32=0x81007dd3, from=0x8100bc31, port=DI1, serial=3d, v=8, event=sendAccumulation.
device=sensorNetwork, Date=2017/6/26/ 10:47:14,  a32=0x81007dd3, from=0x81007853, port=DI1, serial=3d, v=0, event=sendAccumulation.
device=sensorNetwork, Date=2017/6/26/ 10:47:15,  a32=0x81007dd3, from=0x8100f5cd, port=DI1, serial=3d, v=1, event=sendAccumulation.
```

Figure 13: Measurement results on the wiki object page of the measurement system. (After improvement.)

Once again, it should be noted that, except for starting them up and turning them off, we did not manipulate the bot and WSN nodes directly. It should also be noted that when there is an indication of something wrong, or when something interesting is occurring at a particular location, the IoT system manager will often need to intensively review sensor data for a particular place at a particular time. Our IoT system also can satisfy such needs.

4 Related Work

A. Xively

Xively [23] is a cloud service that provides application program interfaces (APIs) for IoT devices. When a developer constructs an IoT system using Xively, the developer writes the IoT device program using Xively’s APIs. However, since Xively does not have a function for sending commands from its API’s to the IoT devices, program updates must be manually performed by the developer at the device location in order to reconfigure the IoT devices. In contrast, as stated above, our IoT can be reconfigured by updating the wiki page script, without requiring the developer to physically access the device.

B. Windows and Apple Software Updates

Windows PC and Macintosh OS users must occasionally download updates from Microsoft [22] and Apple [12], respectively. These updates normally require computer restarts when the process is complete. In contrast, our IoT system does not need restarting after the command sequence and program has been updated on the wiki page. Additionally, the Windows and Apple update processes cannot change the behavior of the computers while rewriting is taking place. In contrast, the command sequence and program of our IoT system can be used to modify the behavior of bots and WSN nodes while the system is operating. Furthermore, Windows and Apple software updates are seldom sent more than once a day, while the wiki page reading frequency of the bot of our IoT system can occur as often as every five minutes.

C. Space Satellites

Serviceable space satellites are usually equipped with in-orbit software replacement functions for maintenance purposes [3] because it is usually impossible to physically repair or update a satellite in space. This situation is similar to the WSN nodes of our IoT. However, software updates for space satellites seldom occur more than once a day, while the wiki page reading frequency of the bot of our IoT system can occur as often as every five minutes.

D. Szczodrak and others

Szczodrak and others have proposed a framework to dynamically reconfigure the WSN and adapt its power consumption, transmission reliability, and data throughput to the different requirements of the applications [4]. Their framework configures the WSN automatically and adjusts the data exchange frequencies. It is possible modify our IoT system to provide similar functions by introducing similar algorithms to the program in the wiki page of our IoT. However, their framework is not suitable for changing IoT functions, while our IoT system permits function changes.

E. NetNucleus Cloud Hub

NetNucleus Cloud Hub [15] achieves the function of controlling IoT devices behind firewalls via a HTTPS/WebSocket connection from IoT devices to Internet servers. This process is very similar to the connection between a wiki page and bot in our IoT system, even though our connection uses HTTP instead of HTTPS/WebSocket. However, while both the NetNucleus Cloud Hub and our IoT system provides a method for managers to control edge devices behind firewalls, NetNucleus Cloud Hub does not have reconfigurable WSN nodes while our IoT system does.

5 Conclusions

We proposed, implemented and demonstrated the remotely reconfigurable IoT system. The IoT system consists of wiki pages, wiki software, bots/gateways, and wireless sensor network nodes. We have realized “wiki for people and machines” by the IoT system. In order to facilitate high availability of the IoT system, we showed cross over including and cross over execution. As examples of reconfiguration, we showed reconfiguration of remote illuminance measuring system and reconfiguration of an activity measurement system.

Since WSNs are not always reliable, IoT developers must plan for unforeseen events. For example, if actuators are connected to WSN nodes, IoT system malfunctions may cause fires or other serious problems.

Currently, our wiki IoT system does not have functions that allow remote reconfigurations, such like exchanging physical sensors and actuators after device deployment, although we expect to address such issues in the future.

Furthermore, we have not yet addressed security for our IoT system, even though basic authentication for wiki sites can be used, and bots and WSN nodes can be placed behind firewalls. Accordingly, it will be necessary to ensure the security of our IoT system before it can be de-ployed for real world usage.

6 Acknowledgement

A part of this research was supported by JSPS KAKENHI Grant Number JP16K00197. We thank students who helped us to conduct the experiment in this paper.

7 References

[1] S. Asanuma, “Japanese Teachers’ Struggle for Active Learning”, 2015 International Consortium for Universities of Education in East Asia, Nov. 1., 2015.

[2] A. Hirata, K. Fujita, K. Isemoto, “Experimental Implementation of an IoT system, Which Can Analysis Data by a Program in R language on a Wiki Page”, Proceedings of the IPSJ IOT Symposium 2016, vol.2016, 2016, pp.91. (in Japanese)

[3] J. Lianxiang, X. Peipei, F. Xuyang, “Software Reconfiguration Technology for Serviceable Satellite OBDH System”, 2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE), DOI: 10.1109/ ICMCCE.2017.30

[4] M. Szczodrak, O. Gnawai, L. P. Carloni, “Dynamic Reconfiguration of Wireless Sensor Networks to Support Heterogeneous Applications”, 2013 IEEE International conference on Distributed Computing in Sensor Systems (DCOSS), DOI: 10.1109/ DCOSS.2013.21A.J. Albrecht, ”Measuring Application-Development Productivity,” Programmer Productivity Issues for the Eighties, 2nd ed., C. Jones, ed., IEEE CS, 1981, pp. 3443.

[5] T. Yamanoue, K. Oda, K. Shimozono “Capturing Malicious Bots using a Beneficial Bot and Wiki”, 2012, In Proceedings of the 40th annual ACM SIGUCCS conference on User services (Memphis, Tennessee, USA. 15-19 Oct. 2012). ACM, New York, NY, 91-96. DOI=https://doi.org/10.1145/2382456.2382477

[6] Takashi Yamanoue, Kentaro Oda, Koichi Shimozono. “A Malicious Bot Capturing System using a Beneficial Bot and Wiki”, 2013, Journal of Information Processing(JIP), vol.21, No.2, pp.237-245.

[7] T. Yamanoue, K. Oda., K. Shimozono. “An Inter-Wiki page Data Processor for a M2M System”, 2013, In Proceedings of the 4th International Conference on E-Service and Knowledge Management (ESKM 2013), Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on.(Matsue, Shimane, Japan, 31 Aug-4 Sep. 2013) IEEE, Los Alamitos, CA. 45-50. DOI= https://doi.org/10.1109/IIAI-AAI.2013.48

[8] T. Yamanoue, K. Oda, K. Shimozono, “Experimentall Implementation of a M2M System Controlled by a Wiki Network”, 2014, In Applied Computing and Information Technology,Studies in Computational Intelligence, Springer, Vol.553, 121-136.

[9] T. Yamanoue, M. Luo, “Experimental implementation of an IoT system which controls sensor terminals of a sensor network by a Wiki page on the Internet”, 2017. IPSJ SIG Technical Report, Vol. 2017-IOT-36, No.12, pp.1-8.(In Japanese)

[10] T. Yamanoue, “Monitoring Servers, With a Little Help from my Bots”, 2017, In Proceedings of the 45th annual ACM SIGUCCS conference on User services (Seattle, Washington, USA. 01-04Oct. 2017). ACM, NewYork, NY, 173-180. DOI=https://doi.org/10.1145/3123458.3123461

[11] Adafruit TSL2561 Digital Luminosity/Lux/Light Sensor Breakout, https://www.adafruit.com/product/439

[12] Apple Software Update, https://en.wikipedia.org/wiki/List_of_macOS_components#Software _Update

[13] I2C-Bus, https://www.i2c-bus.org

[14] MonoStick, https://mono-wireless.com/jp/products/MoNoStick/

[15] NetNucleus Cloud Hub,
https://www.tjsys.co.jp/embedded/netnucleus-cloudhub/index_j.htm

[16] PukiWiki, http://pukiwiki.sourceforge.jp/

[17] Raspberry Pi, https://www.raspberrypi.org/

[18] TAOS, TSL2560, TSL2561 Light to Digital Converter;
https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf

[19] TWELITE DIP, https://mono-wireless.com/jp/products/TWE-Lite-DIP/index.html

[20] Ward Cunningham, Wiki Wiki Web, http://c2.com/cgi/wiki?WikiWikiWeb

[21] Wikipedia, http://www.wikipedia.org/

[22] Windows update, https://en.wikipedia.org/wiki/Windows_Update.

[23] Xively, https://xively.com

[24] ZigBee Aliance, http://www.zigbee.org