An Application Development Environment for Rule-Based I/O Control Devices

Ryohei Sagara[1], Yasue Kishino[1], Tsutomu Terada[1], Tomoki Yoshihisa[2], Masahiko Tsukamoto[3], and Shojiro Nishio[1]

[1] Graduate School of Information Science and Technology, Osaka University, Japan
[2] Academic Center for Computing and Media Studies, Kyoto University, Japan
[3] Faculty of Engineering, Kobe University, Japan

Abstract. In this paper, we propose an application development environment for the ubiquitous chip, which is a rule-based event-driven input/output (I/O) control device for constructing ubiquitous computing environments. The proposed development environment simulates the behaviors of multiple ubiquitous chips and helps users to create rules. Moreover, it has a function for developing applications by cooperation between virtual ubiquitous chips and real ubiquitous chips. The application environment enables both programmers and general users to develop and customize applications for ubiquitous computing environments.

1 Introduction

Recent evolutions in the miniaturization of computers and component devices such as microchips, sensors, and wireless modules, contribute to the achievement of ubiquitous computing environments [4, 8, 10]. In our ubiquitous computing environments, small devices are embedded in many places to support daily human life. To construct ubiquitous computing environments, we propose a rule-based I/O control device called *ubiquitous chip* [9].

The behaviors of a ubiquitous chip are described by a set of event-driven rules, and a ubiquitous chip can dynamically change its behavior by modifying stored rules. In our assumed environments, ubiquitous chips are embedded into almost any artifacts to enrich our daily-life, and we can customize functions and services in ubiquitous chips according to our preference.

To achieve such environments, we need an application development environment that enables both programmers and general users to intuitively develop/customize applications. In response to these requirements, we propose a development environment for ubiquitous chips that simulates the behaviors of multiple ubiquitous chips and helps users to create rules. Moreover, this proposed environment includes a function for developing applications through cooperation between virtual and real ubiquitous chips.

The remainder of this paper is organized as follows. Section 2 outlines the ubiquitous chip. Section 3 describes the design of the proposed application development environment, and Section 4 describes a prototype system. Section 5 discusses the development environment and Section 6 sets forth our conclusions and planned future work.

2 Ubiquitous Chip

As shown in Figure 1, a ubiquitous chip consists of a core part, which is the main unit, and a cloth part that has connectors and a rechargeable battery. It has five digital input ports, one analog input port, twelve digital output ports, two serial communication ports, and a multi-purpose LED. Figure 2 shows the various input/output devices for the ubiquitous chip such as sensors, input devices, and actuators. Using these attachments, we can flexibly change configurations of ubiquitous chips. The behaviors of ubiquitous chip are described by a set of ECA rules, which are used for describing behaviors in event-driven databases. An ECA rule consists of Event, Condition, and Action. Event is an occurring event, Condition is a condition for executing actions, and Action is the operations to be carried out. Tables 1, 2, and 3 show the lists of events, conditions, and actions that can be used on the ubiquitous chip.

A ubiquitous chip communicates with other ubiquitous chips via its serial communication ports. We can use the SEND_MESSAGE action, the SEND_DATA action, and SEND_COMMAND action as communication functions. The SEND_MESSAGE action sends a message that has a specific ID (0-7). The SEND_DATA action sends one byte data that is specified in the rule or input voltage of the analog port. The SEND_COMMAND action sends a command to remotely manage ECA rules stored in ubiquitous chips. Table 4 shows the lists of commands that can be sent by the SEND_COMMAND action. The DEMAND_DATA command demands the one byte data specified address of the memory in a ubiquitous chip. When a ubiquitous chip receives a DEMAND_DATA command, it returns the required data as a REPLY_DATA command.

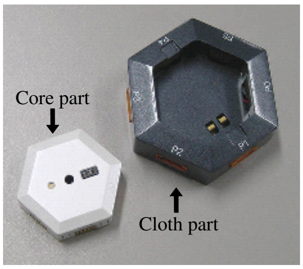

Fig. 1. Ubiquitous chip

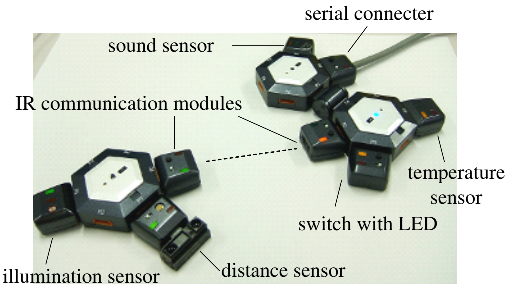

Fig. 2. Attachments for ubiquitous chip

Table 1. Events

Name	Contents
TIMER_EXPIRE	Firing a timer
RECEIVE_MESSAGE	8 types of message reception via a serial port
RECEIVE_DATA	1 byte data reception via a serial port
NONE	Evaluating conditions at all times

Table 2. Conditions

Name	Contents
INPUT	On/Off state of input ports
ANALOG_INPUT	Range of input from the analog port
INPUT_STATE	Value of internal variables
TIMER_ID	ID of fired timer
MESSAGE_ID	ID of received message
DATA_RANGE	Range of received data

Table 3. Actions

Name	Contents
OUTPUT	On/Off control of output ports
OUTPUT_STATE	On/Off control of state variables
TIMER	Setting a new timer
SEND_MESSAGE	Sending a message
SEND_DATA	Sending a 1 byte data
SEND_COMMAND	Sending a command
HW_CONTROL	Hardware control

Table 4. Commands

Name	Contents
ADD_ECA	Adding a new ECA rule
DELETE_ECA	Deleting a specific ECA rule(s)
ENABLE_ECA	Enabling a specific ECA rule(s)
DISABLE_ECA	Disabling a specific ECA rule(s)
DEMAND_DATA	Requesting a data of EEPROM
REPLY_DATA	Sending a data of EEPROM (reply to DEMAND_DATA)

3 Design of Application Development Environment

3.1 Requirements

In this research, we assume that ubiquitous chips are embedded into almost any artifacts such as furniture, appliances, walls, and floors, and that they cooperate with each other and provide various services. These services are required to be adaptable to user preferences, as users may want to customize services according to their own requirements. For example, we envisage the following situations:

– When a user buys a new piece of furniture that features an embedded ubiquitous chip and sensors, he/she customizes a room automation application, which is already available in his/her room to integrate the new furniture into the application.
– When a user redecorates his/her room, he/she modifies the application according to the new allocation.
– When a user changes his/her routine, he/she adjusts the applications.
– A user uses actual I/O devices to check the behavior of an application.

We construct an application development environment for ubiquitous chips that visualizes the behavior of applications and achieves easy development for users.

Moreover, the development environment also provides a function for verifying applications with actual I/O devices and ubiquitous chips to enable users to develop/customize applications intuitively.

3.2 Approach

In order to satisfy the above requirements, our application development environment has the following functions.

Simulation with Virtual Ubiquitous Chips

Services in ubiquitous computing environments are realized through cooperation among multiple ubiquitous chips. In such situations, it is difficult for users to grasp the existing configurations and construct applications taking into consideration of the relationships among multiple ubiquitous chips. Therefore, our application development environment needs a function that simulates multiple virtual ubiquitous chips, which process their ECA rules in the same way as the real ubiquitous chip. A virtual ubiquitous chip has the following characteristics:

– A virtual ubiquitous chip has a hexagonal shape, I/O ports, serial ports, and a multi-purpose LED, the same as a real ubiquitous chip.
– An arbitrary number of virtual ubiquitous chips can be added/deleted to/from the simulation environment. A user can add/delete connections between I/O ports and serial ports over multiple ubiquitous chips.
– The state of I/O ports and the multi-purpose LED are displayed at all times as a series of colored circles.
– A user can check a virtual ubiquitous chip's internal variables and stored ECA rules even when an application is running.
– A user can add new ECA rules easily without professional knowledge. The application development environment has an ECA rule editor, which enables general users to write ECA rules easily. Moreover, a user can check stored ECA rules in a style similar to natural language.

Cooperation Among Real/Virtual Ubiquitous Chips

The application development environment has a function for constructing applications through cooperation between a virtual ubiquitous chip and a real ubiquitous chip. This function achieves the following implementation styles:

Case 1. A user customizes the application that is in-service on a real ubiquitous chip.
Case 2. A user checks the behaviors of real I/O devices at the final step of application development.

The application development environment manages the state of a real ubiquitous chip in the same way as a virtual ubiquitous chip by linking their states. For example, as Figure 3 shows, when a user pushes the button connected to the real ubiquitous chip, the input port of the associated virtual ubiquitous chip is turned on. Likewise, when the output port of the virtual ubiquitous chip is turned on, the output port of the real ubiquitous chip is also turned on.

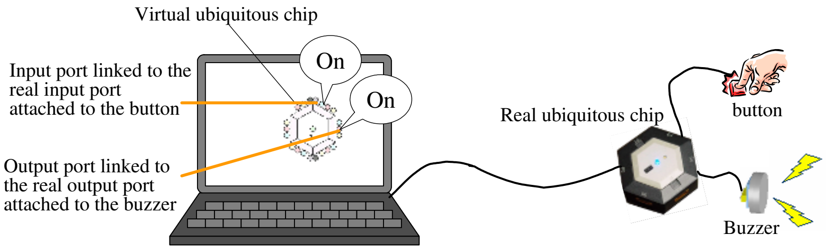
Virtual ubiqiutous chip
Input port linked to the real input port attached to the button
Output port linked to the real output port attached to the buzzer
On
On
Real ubiquitous chip
button
Buzzer

Fig. 3. Cooperation among real and virtual ubiquitous chips

4 Implementation

We have implemented a prototype of the application development environment. In this section, we explain the details of its implementation and show an example of its use. Figure 4 shows a snapshot of the application development using a PC and a real ubiquitous chip.

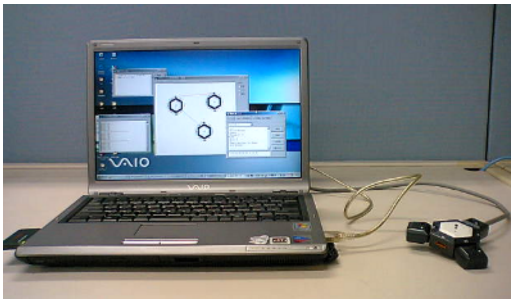

Fig. 4. Example of a application development using a PC and a ubiquitous chip

4.1 Simulation with Virtual Ubiquitous Chips

Figure 5 shows a screenshot of the development environment. In the proposed development environment, the behaviors of ubiquitous chips are simulated by virtual ubiquitous chips. A virtual ubiquitous chip is illustrated as a hexagon and the circles indicate I/O ports, serial ports, and a multi-purpose LED. One input port and two output ports are placed along each edge of the hexagon, likewise in the real ubiquitous chip. The state of the I/O ports and the multi-purpose LED are expressed by differences in their color. A user operates the virtual ubiquitous chip in the following ways:

- places multiple virtual ubiquitous chips in the simulation area.
- toggles input ports.
- checks the state of the output ports and the multi-purpose LED.

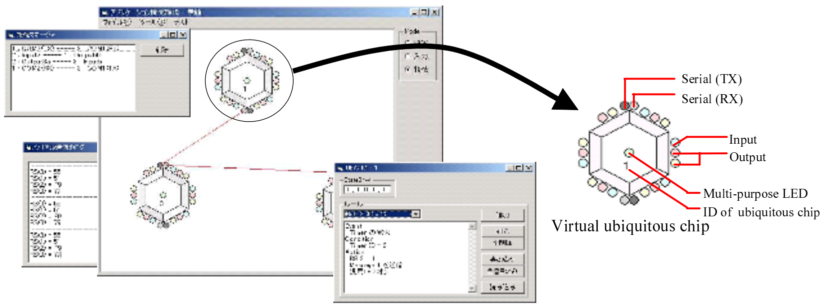

Fig. 5. Screenshot of the application development environment

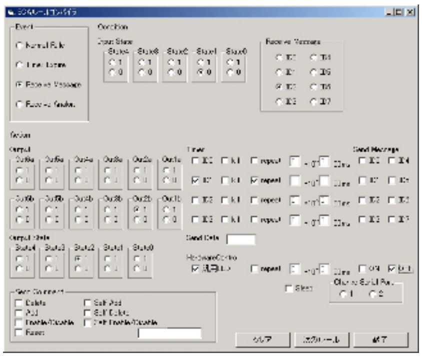

Fig. 6. ECA rule editor

– connects I/O ports and serial ports to the other ubiquitous chips.
– checks the value of the internal variables and the stored ECA rules.
– adds new ECA rules using ECA rule editor (Figure 6).

Users have only to use a mouse to achieve the above operations.

4.2 Cooperation Among Real/Virtual Ubiquitous Chips

As described in Section 3.2, cooperation is classified into two cases: a user customizes an application using a real ubiquitous chip and a user checks the behaviors of real I/O devices.

In the former case, cooperation is achieved as follows:

Step 1. A user connects a real ubiquitous chip to the PC.
Step 2. A user places a new virtual ubiquitous chip in the simulation area.
Step 3. The application development environment reads the ECA rules stored in the real ubiquitous chip and adds them to the virtual ubiquitous chip.
Step 4. The development environment sends the DELETE_ECA command to the real ubiquitous chip to delete all stored ECA rules, this prevents conflict among the ECA rules.

Table 5. Formula for creating control rules

Original rule	Control rule
E: (Any event) C: I(i)=0 (i=1-5) A: (Any action)	E: NONE C: I(i)=0, S(i-1)=1 A: S(i-1)=0, SEND_DATA(2i-1)
E: (Any event) C: I(i)=1 (i=1-5) A: (Any action)	E: NONE C: I(i)=1, S(i-1)=0 A: S(i-1)=1, SEND_DATA(2i)
E: (Any event) C: (Any condition) A: O(i)=0 (i=1-12)	E: RECEIVE_DATA C: RECEIVED_DATA=2i-1 A: O(i)=0
E: (Any event) C: (Any condition) A: O(i)=1 (i=1-12)	E: RECEIVE_DATA C: RECEIVED_DATA=2i A: O(i)=1
E: (Any event) C: (Any condition) A: HW_CONTROL	E: RECEIVE_DATA C: RECEIVED_DATA=25 A: HW_CONTROL
E: (Any event) C: (Any condition) A: HW_CONTROL(M_LED OFF)	E: RECEIVE_DATA C: RECEIVED_DATA=26 A: HW_CONTROL(M_LED OFF)

Step 5. The development environment writes rules to the real ubiquitous chip, which lets the real ubiquitous chip behave in the same manner as the virtual ubiquitous chip.

In the latter case, cooperation is realized by performing only Steps 4 and 5 of the above procedure.

Table 5 shows the formula for creating control rules. When the state of a real input port changes, the real ubiquitous chip sends one byte data to the development environment. When the development environment receives the data, it changes the state of the associated virtual input port.

4.3 Example

In this section, we give an example of the use of the proposed application development environment. The sample application behaves as though "a user is sitting on a chair, and the desk lamp lights automatically when there is not enough bright." In this application, we use three ubiquitous chips called UC1, UC2, and UC3. UC1 is attached to the chair and has a pressure sensor that detects when the user is sitting. UC2 is attached to the desk and is connected to the desk lamp in order to control it. UC3 is attached to the wall and has an illumination sensor. Figure 7 shows the connection relationship of the ubiquitous chips.

The user programs the application in the following way:

1. The user positions the three virtual ubiquitous chips, UC1, UC2, and UC3.
2. The user connects the I/O ports and serial ports as shown in Figure 7.
3. The user adds the ECA rules shown in Table 6 using the ECA rule editor.

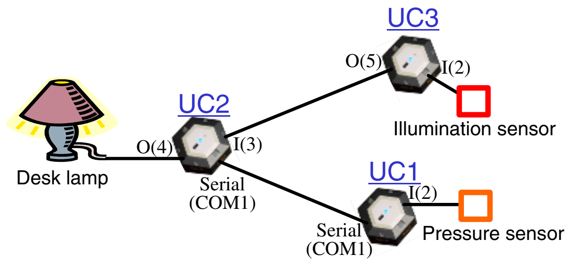

Fig. 7. System structure of sample application

Table 6. Rule set for the sample application

Rules for UC1 (2 rules)	
E: NONE C: I(2)=1, S(0)=0 A: S(0)=1, SEND_MESSAGE(M0)	E: NONE C: I(2)=0, S(0)=1 A: S(0)=0, SEND_MESSAGE(M1)

Rules for UC2 (5 rules)		
E: RECEIVE_MESSAGE C: MESSAGE_ID=0 A: S(0)=1	E: RECEIVE_MESSAGE C: MESSAGE_ID=1 A: S(0)=0, O(4)=0	E: NONE C: I(3)=1 A: S(1)=0, O(4)=0
E: NONE C: I(3)=0 A: S(1)=1	E: NONE C: S(0)=1, S(1)=1 A: O(4)=1	

Rules for UC3 (2 rules)	
E: NONE C: I(2)=1 A: O(5)=1	E: NONE C: I(2)=0 A: O(5)=0

Control rules for UC3 (2 rules)	
E: NONE C: I(2)=0, S(2)=1 A: S(2)=0, SEND_DATA(3)	E: NONE C: I(2)=1, S(2)=0 A: S(2)=1, SEND_DATA(4)

4. The user checks the behavior of the ubiquitous chips by toggling their input ports. If bugs are found, the user modifies the rules.
5. When the user wants to confirm the behavior of the application with a real illumination sensor, he connects a real ubiquitous chip to a PC and links UC3 and the real ubiquitous chip. In this case, the control rules shown in Table 6 are automatically added to the real ubiquitous chip. The user changes the brightness of the room and checks the behavior.
6. When he completes the application, he writes ECA rules to real ubiquitous chips and attaches them to furniture.

5 Consideration

5.1 Planned Functions

When more than seven virtual ubiquitous chips are placed, the simulation area becomes full. Thus, it is difficult to develop applications consisting of ten or more ubiquitous chips. To solve this problem, we are planning to develop functions that can group several ubiquitous chips into a meaningful unit and that can manage groups collectively.

Although we can grasp the state of I/O ports through the circles of a virtual ubiquitous chip, we cannot know the behavior of connected devices. Therefore, we should provide virtual I/O devices and functions that can simulate their behaviors.

In this paper, we focus on serial ports connected by means of wired cables. Practically, we have provided various wireless communication units for ubiquitous chips such as Infrared (IR) units, Radio Frequency (RF) units, and Bluetooth units. The development environment should be able to support to simulate wireless communication.

5.2 Related Work

Smart-It [2], MOTE [3], and U-Cube [5] are small devices for constructing ubiquitous computing environments and sensor networks. These devices have sensors/actuators and wireless modules and they are similar to ubiquitous chip in the point that we can customize system configurations by changing the attached devices. However, we cannot change their behaviors or the attached devices while applications are running. Therefore, it is difficult to dynamically customize the behaviors of embedded devices according to user demands. Moreover, since these devices are developed with a C-like programming language, it is difficult for general users to develop and customize applications.

MINDSTORMS [6] and ROBOT WORKS [1] have application development environments for specific hardware. Users can easily program applications by aligning blocks in which conditions and operations are described. However, these development environments do not have simulation functions. Moreover, they cannot develop applications through cooperation with actual hardwares.

MPLAB [7] is a development environment for PIC, which is a microprocessor used in ubiquitous chip. MPLAB can simulate the behaviors of PIC by displaying the values of variables. However, it cannot simulate the behaviors of multiple PICs and it cannot visualize the states of I/O ports.

6 Conclusion

In this paper, we described the design and implementation of an application development environment for ubiquitous chips. The proposed development environment simulates the behaviors of multiple ubiquitous chips. Moreover, it has a function for verifying applications through cooperating with real ubiquitous chips.

In future, we have plans to construct functions for developing large-scale applications, for simulating I/O devices, and for cooperating with multiple real ubiquitous chips. We also plan operational tests and further evaluation of the application development environment.

Acknowledgement

This research was partially supported by The 21st Century Center of Excellence Program "New Information Technologies for Building a Networked Symbiotic Environment" and Grant-in-Aid for Scientific Research (A)(17200006) from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

1. BANDAI: "ROBOT WORKS," http://www.roboken.channel.or.jp/borg/.
2. M. Beigl and H. Gellersen: "Smart-Its: An Embedded Platform for Smart Objects," *Smart Objects Conference (sOc)* (May. 2003).
3. Crossbow Technology Inc.: "MICA," http://www.xbow.com/products/Wireless_Sensor_Networks.htm.
4. J. Kahn, R. Katz, and K. Pister: "Mobile Networking for Smart Dust," in *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99)*, pp. 271–278 (Aug. 1999).
5. Y. Kawahara, M. Minami, H. Morikawa, and T. Aoyama: "Design and Implementation of a Sensor Network Node for Ubiquitous Computing Environment," in *Proc. VTC2003-Fall* (Oct. 2003).
6. LEGO: "MINDSTORMS," http://mindstorms.lego.com/japan/products/.
7. Microchip Technology Inc.: "MPLAB," http://www.microchip.com/1010/index.htm.
8. K. Sakamura: "TRON: Total Architecture," in *Proc. Architecture Workshop in Japan'84*, pp.41–50 (Aug. 1984).
9. T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, S. Nishio, and A. Kashitani: "Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing," in *Proc. Int'l Conf. on Pervasive Computing (Pervasive 2004)*, pp.238–253 (Apr. 2004).
10. M. Weiser: "The Computer for the Twenty-first Century," *Scientific American*, Vol. 265, No. 3, pp. 94–104 (Sept. 1991).