INTRODUCTION

Computers are very effective for controlling the environment in greenhouses (Takakura et al., 1979), and computer control systems are widely used in large-scale greenhouse horticultural production. Various methods for controlling greenhouse environments have been developed (Chalabi et al., 1996; Aaslyng et al., 2005; El Ghoumari et al., 2005), and computer systems have become indispensable for complicated control. But high-performance computer system is expensive, and it is difficult to install the system in small-scale greenhouses. Although commercial computer systems have highly user-friendly interfaces and the ability to precisely control various kinds of device (e.g., heaters, ventilators, shading curtains, irrigation systems, fogging systems), the control logic in these commercial systems is sometimes inflexible, and it is sometimes impossible to modify the original complicated control logic. Modifying the original control logic frequently requires users to choose dedicated hardware and develop new software to operate it (Handarto et al., 2006; 2007). This inflexibility restricts the spread of this type of computer system, and because development of specialized programs is necessary in the case of controlling simple logic, it is not very user-friendly. At the present time, there is no flexible, low-cost environmental control system suitable for use in small-scale greenhouses.

Against this backdrop, the Ubiquitous Environment Control System (UECS) was developed as a decentralized autonomous system for controlling greenhouse environments (Hoshi et al., 2004). UECS is a system for controlling greenhouse environments that communicates information via a local area network, and devices utilizing the library that we developed can perform the UECS defined communication tasks automatically. With the help of this library, device developers no longer need to program the communications aspects of the device and can concentrate on programming setting and control logic of the device. The library occupies about 29 kilobytes of the read only memory area of the target board. The library and associated open-source microcontroller boards are powerful tools for developing low-cost environmental control systems.

Keywords: Arduino, environment control system, open source, program library, ubiquitous environment control system

Development of Program Library Using an Open-source Hardware for Implementation of Low-cost Greenhouse Environmental Control System

Ken-ichiro YASUBA1*, Hidehito KUROSAKI2*, Takehiko HOSHI3, Takashi OKAYASU4, Yoshiyuki TANAKA1, Tanjuro GOTO1 and Yuichi YOSHIDA1

1 Graduate School of Environmental and Life Science, Okayama University, 1–1–1 Tsushimanaka Kita-ku, Okayama 700–8530, Japan
2 Western Region Agricultural Research Center, National Agriculture and Food Research Organization (NARO), 2575 Ikano, Zentsuji, Kagawa 765–0053, Japan
3 Faculty of Biology-Oriented Science and Technology, Kinki University, 930 Nishimitani, Kinokawa, Wakayama 649–6493, Japan
4 Faculty of Agriculture, Kyushu University, 6–10–1 Hakozaki, Higashi-ku, Fukuoka 812–8581, Japan

(Received August 30, 2016; Accepted May 2, 2018)

A program library for use on open-source hardware was developed in order to construct a low-cost environmental control system for greenhouses. The library facilitated the development of environmental sensing and control devices that conform to the protocols of the Ubiquitous Environment Control System (UECS). The open-source hardware used was the “Arduino Ethernet” and the “Arduino Mega 2560 with Ethernet Shield” microcontroller boards. UECS is a system for controlling greenhouse environments that communicates information via a local area network, and devices utilizing the library that we developed can perform the UECS defined communication tasks automatically. With the help of this library, device developers no longer need to program the communications aspects of the device and can concentrate on programming setting and control logic of the device. The library occupies about 29 kilobytes of the read only memory area of the target board. The library and associated open-source microcontroller boards are powerful tools for developing low-cost environmental control systems.

Keywords :  Arduino, environment control system, open source, program library, ubiquitous environment control system
ate for use in small-scale greenhouse control systems. In additional, highly precise environmental control might be attained by introducing this system (Hoshi et al., 2011; Yasuba et al., 2011).

The specifications of the communications protocol for UECS are available on the UECS website (http://www.uecs.jp/en/protocol-e.html). Messages for communication with UECS are formatted in Extensible Markup Language (XML), and each message is communicated by means of a user datagram packet (UDP). Each node in the system needs to gather the information communicated by using UDP and analyze it. Although the work of the analyzing algorithms is almost routine, manually coding the steps involved in analyzing the information is difficult. It might, therefore, be very useful to have a library of code to draw upon to conduct routine communication and analysis work.

Recently, the Arduino open-source electronic prototyping platform has been attracting attention. The electrical circuit diagrams of the Arduino microcomputer boards are open to all, and the boards can be self-produced. Moreover, the development environment for programing the Arduino boards is offered freely. These boards have been successfully used in the development of several types of devices. For example, Teikari et al. (2012) developed a stimulator system for vision research using an Arduino board for educational and research usage, and they reported that it was inexpensive. Kornuta et al. (2013) developed a microcontroller for time-varying waveforms on a peristaltic pump using an Arduino board, and they reported that this board was an inexpensive and flexible platform for creating controllers. Arduino boards have also been utilized in devices developed for factory environmental monitoring (Lian et al., 2013) and outdoor meteorological measurements (Hirafuji et al., 2013). Arduino boards are primarily used to reduce costs of the developed devices, and using these boards could possibly decrease the cost of producing UECS nodes. A meteorological measuring node for UECS was developed using the Arduino Ethernet board (Yasuba et al., 2013).

A program library for use with Arduino boards that can automatically carry out transmission and reception of the correspondence based on the UECS protocol would be very useful for developing UECS nodes. If such a library could be supplied, an environmental control system could be easily constructed at low cost, and all by using freely available open-source software and hardware components. We developed such a program library for developing UECS nodes.

MATERIALS AND METHODS

Microcomputer boards for the program library

Our developed library requires Version 1.0.4 of the Arduino application development environment. The library is supported by two microcomputer boards: Arduino Ethernet and Arduino Mega 2560 with Ethernet Shield. The electrical circuit diagrams of these boards are openly available on the Arduino website (for example, Arduino Ethernet, http://arduino.cc/en/uploads/Main/arduino-ethernet-R3-schematic.pdf).

The library is contained in a folder named ARDUUECS02. This folder is not initially in the development environment but must be imported from the “libraries” folder from within the Arduino 1.0.4 development environment.

**UECS communication formats**

In UECS, information is communicated from one node to another via LAN. The format for the information is defined in the UECS protocol book (http://uecs.jp/uecs/kiyaku/UECSStandard100_E10.pdf). A message with the defined format is called a common correspondence message (CCM) in the protocol book. Information concerning sensor readings, actuator movement status, and remote control is all defined in this system, and each node analyzes information from other nodes via LAN and controls certain actuators based on that information. Simple logic for controlling environments can be mounted directly into the node program, but more complicated logic is done by sending CCMs for remote control from a personal computer (PC).

The main work of our developed library is to manage the sending and receiving of CCMs.

**Items necessary for use of the library**

Code required to use a library must be described in what is called a “sketch” in the Arduino development environment. In the first part of the sketch are the preprocessor “#include” directives (lines 1–6 in the sample code, Fig. 1). Next, some constant values and variables must be set by the developer (Fig. 1, lines 14–46).

The user program is mainly divided into four functions defined by the library (UserInit, setSendP1Page, UserEvery1Sec, and UserEveryLoop). The UserInit function describes the processes that are to be dealt with just after starting the node program. The UserEvery1Sec function is called periodically at intervals of 1 s. This is the shortest interval defined for CCM communication, so it is the most appropriate timing. The UserEveryLoop function is called in a loop function, which must be defined as a standard of the Arduino native development environment. This function is run at a very high frequency compared with the UserEvery1Sec function. The setSendP1Page function is called when someone accesses the HTTP server, which is supplied by the library, and sends an HTTP protocol “GET” request.

**Operating algorithm of the function**

Each node, which incorporates the developed library, is managed according to the algorithm shown in Fig. 2. The UserInit function is run first, and messages received and sent must be registered into the function. The MAC address must be defined in this function. In the sample code, two sent CCMs are defined by the UECSSetCCM function, and the MAC address is set in the U_orgAttribute.mac array in the UserInit function (Fig. 1, lines 48–57). After that, the algorithm goes into an infinite loop. Immediately after entering the loop, the library checks whether an HTTP request has been received. If a request is received, it transmits the response for displaying on a screen on a web browser (as in Fig. 3) and runs the setSendP1Page func-
The library checks whether a UDP concerning a CCM was received on port 16520. In order to use the information from other nodes, it is necessary to register received CCMs in the UserInit function. The UECSSetCCM function is used for registration of sent and received CCMs. The function must be used in the UserInit function. So, for registration of sent and received CCMs in the UserInit function, the library sends and receives the CCMs automatically.

If the format of the received packet conforms to the specifications of the U ECS protocol book and the content has the information of the type registered in UserInit function, the latest value is updated according to the content of the packet. The library checks whether the transmitting request of correspondence was received on UDP port 16529.

Fig. 1 Sample code for using the library.

Fig. 2 Flowchart of node program using the library we developed.
If it is received, the library answers the correspondence with CCMSCAN or NODESCAN, which is defined in the UECS application protocol. The library runs the UserEveryLoop function after that.

The UserEvery1sec function is run if 1 s or more has passed after last performing this function. If the developer sends a CCM outside the usual transmitting timing, a flag for transmitting might be set in this function. After this function, the library sends a CCM with UDP protocol. The periodically transmitted information is defined in the UserInit function, so the library transmits the message formatted according to the UECS protocol at the defined interval.

The following is an example of the message defined in Fig. 1, line 50 of the sample code, showing the case when the IP address is set to 192.168.1.7:

```xml
<?xml version="1.0"?>
<UECS ver="1.00-E10">
  <DATA type="cnd.xxx" room="0" region="0">
    order="0" priority="29">0</DATA>
  <IP>192.168.1.7</IP>
</UECS>
```

The UDP, in which the above message is described, is broadcasted through the LAN every second. The type attribute of the message defined in Fig. 1, line 49 is “DryTemp.xxx”, which is different from that of above message (“cnd.xxx”). This message is broadcasted every 10 s. The value of the <DATA> tag changes depending on the input voltage of the analog pin, which is named A0 (see Fig. 1, line 60).

**Http server**

The library also supplies the service of an HTTP server. If the end-user accesses TCP port 80 from a browser by hypertext transport protocol (HTTP), the library responds with the HTTP document (Fig. 3A). The document includes the values from the received CCM defined in the UserInit function and the send values of the sender CCM. The information concerning the CCM is displayed in a table on the upper side of the web page (Fig. 3B).

In addition, if the developer wants to provide information other than the CCM to the end-user, they need to modify the values of the U_html array in the UserInit function (Fig. 1, lines 35–38). The U_html array is of UECSUserHtml structure, and there are various types of information in the UECSUserHtml structure that can be displayed on the web page. The HTTP server provides three web pages for end-users (Fig. 3).

End-users can set the values for developed devices by using the HTTP server provided by the library. The HTTP server provides a service for inputting text (Fig. 3B) and selecting some items.

**Size of the library**

The library uses about 29 kilobytes of ROM area. The sample code shown in Fig. 1 occupies 29,270 bytes of ROM area. If it is used with the Arduino Ethernet board, the available ROM remaining for developer coding is about 3 kilobytes. If used with the Arduino Mega 2560 with Ethernet Shield, the available ROM area remaining is about 228 kilobytes.

**Node using the developed library**

A node that was created using the developed library is show in Fig. 4. This node measures dry- and wet-bulb temperatures and calculates relative humidity from the two measured values. Values of dry-bulb temperature and relative humidity are sent automatically in the UECS CCM format. An electrical relay to control a ventilation fan is in
installed at the node. The end-user sets the ventilation temperature after accessing the HTTP server on the node, and the contact of the relay is closed when the dry-bulb temperature exceeds the set value.

RESULTS AND DISCUSSION

The library was developed in order to construct a low-cost environmental control system for use in small-scale greenhouse production. We found that construction of the system using Arduino boards has several merits. D’Ausilio (2012) indicated that the Arduino platform was more attractive than other platforms for controlling devices for laboratory usage partly because the open-source philosophy of this electronics platform means that information is readily available via the internet. We also thought that the development of nodes utilizing the Arduino platform and our developed library would be effective for constructing a low-cost environmental controlling system in protected horticulture.

UECS would be useful for controlling environments, irrespective of the scale of greenhouse, by using both the decentralized autonomous mode, in which the program within the node controls the environment, and the remote controlling mode by PC. The library offers simple services only, because we could not install a program of larger size on the Arduino Ethernet board. A node fabricated with an Arduino board using this library is shown in Fig. 4. Those simple features could be installed on an Arduino Ethernet board, but installation of more complicated logic on an Arduino Ethernet board is difficult. The use of an Arduino Mega 2560 with Ethernet Shield could be selected to overcome this limitation, or the node could be controlled remotely, if the PC that sent the message for the remote control operation complies with the UECS protocol.

To test newly developed environmental control logic, controlling systems have sometimes been constructed on original vendor-developed systems (Chalabi et al., 1996; Aaslyng et al., 2005). However, the hardware of the computers used for controlling the environment was not open source, which means that if the vendor stopped the supply of the devices, it would be difficult to recreate the system with the same logic. Data logging systems have been installed and programmed for controlling devices (Handarto et al., 2006; Handarto et al., 2007), and high-performance data logging systems (e.g., the CR1000 or CR10X provided by Campbell Scientific, North Logan, UT, USA) are adequate to test complicated algorithms. However, the programs must be specifically designed for the controlling devices using non-open-source and machine-dependent programming languages. These methods pose the risk of the system becoming inoperable if the vendor discontinues support services. A completely open-source system, on the other hand, could be operated stably by either a third party or with support from the developed library. In addition, end-users using the UECS monitoring software (Yasuba et al., 2012) are able to publish tables and time-series graphs concerning environmental control on the internet by using FTP (file transfer protocol) and receive daily reports and alert information by email.

We expect that the developed library will improve the budget for programming. The node created for measuring the environment was developed according to the UECS application protocol (Yasuba et al., 2013). In this case, if the code were made from scratch, the developer would need to write more than 350 lines of program code. However, if the library is used for creating the node, the developer would need to write no more than 70 lines of code to install nearly the same functions as the communication manager for UECS and an HTTP server.

It was expected that environmental controlling systems made by utilizing Arduino boards could be constructed at low cost. Traditional environmental controlling systems are too expensive to install in small greenhouses, and it has been difficult for horticulturists with small greenhouses to receive the benefit of integrated environmental control. The library that we developed is expected to be of use in the construction of UECS for small-scale protected horticulture.

CONCLUSION

We developed the program library for UECS to use Arduino boards as UECS device. Environmental controlling device was expensive in general, so it has been not spread in Japan. Spread of low cost devices is expected by combination with open source hardware and program library. Research project for installing environmental controlling devices in small scale greenhouse at low cost was started newly, and the library was used for making low cost environmental controlling devices. It becomes easy to make self-made device by this library, so we will expect to develop original controller with creative controlling logic by this library users.

ACKNOWLEDGEMENTS

This work was supported by KAKENHI (Grant-in-Aid for Scientific Research (B) No. 13379368).

REFERENCES

Aaslyng, J. M., Ehler, N., Jakobsen, L. 2005. Climate control software integration with a greenhouse environmental control computer. Environ. Modell. Softw. 20: 521–527.

Chalabi, Z. S., Bailey, B. J., Wilkinson, D. J. 1996. A real-time optimal control algorithm for greenhouse heating. Comput. Electron. Agric. 15: 1–13.

D’Ausilio, A. 2012. Arduino: A low-cost multipurpose lab equipment. Behav. Res. Methods 44: 305–313.

El Ghomari, M. Y., Tantau, H. -J., Serrano, J. 2005. Non-linear constrained MPC: Real-time implementation of greenhouse air temperature control. Comput. Electron. Agric. 49: 345–356.

Handarto, Hayashi, M., Ohyama, K., Toida, H., Goto, E., Kozai, T. 2006. Developing control logic for a high-pressure fog cooling system operation for a naturally ventilated greenhouse. Environ. Control Biol. 44: 1–9.

Handarto, Hayashi, M., Goto, E., Kozai, T. 2007. Experimental verification of control logic for operation of a fog cooling system for a naturally ventilated greenhouse. Environ. Control Biol. 45: 47–58.
Higashide, T., Yasuba, K., Kuroyanagi, T., Nakano, A. 2015. Decreasing or non-decreasing allocation of dry matter to fruit in Japanese tomato cultivars in spite of the increase in total dry matter of plants by CO2 elevation and fogging. Hortic. J. 84: 111–121.

Hirafuji, M., Yoichi, H., Miki, Y., Kiura, T., Fukatsu, T., Tanaka, K., Matsumoto, K., Hoshi, N., Nesumi, H., Shibuya, Y., Itoh, A., Ninomiya, S., Adinarayana, J., Sudharsan, D., Saito, Y., Kobayashi, K., Suzuki, T. 2013 Development of an open field server and sensor cloud system. (in Japanese with English summary) Agric. Info. Res. 22: 60–70.

Hoshi, T., Hayashi, Y., Uchino, H. 2004. Development of a decentralized, autonomous greenhouse environmental controlling system in a ubiquitous computing and internet environment. Proc. 2004 AFITA/WCCA Joint Congress on IT in Agriculture. 490–495.

Hoshi, T., Higa, H., Goto, K., Niibori, K. 2011. Effects of supplemental lighting on the quality of tomato seedlings raised in greenhouses. Acta Hortic. 907: 117–123.

Kornuta, J. A., Nipper, M. E., Dixon, J. B. 2013. Low-cost microcontroller platform for studying lymphatic biomechanics in vitro. J. Biomech. 46: 183–186.

Lian, K. Y., Hsiao, S. J., Sung, W. T. 2013. Mobile monitoring and embedded control system for factory environment. Sensors 13: 17379–17413.

Takakura, T., Okada, M., Shimaji, H., Nara, M. 1979. Development of microcomputer-based multi-variable control system for greenhouse. (in Japanese with English summary) J. Agr. Met. 35: 97–102.

Teikari, P., Najjar, R. P., Malkki, H., Knoblauch, K., Dumortier, D., Gronfier, C., Cooper, H. M. 2012. An inexpensive Arduino-based LED stimulator system for vision research. J. Neurosci. Meth. 211: 227–236.

Yasuba, K., Suzuki, K., Takachi, M. 2011. Influence of humidity control for decreasing air temperature without shading on light environment and leaf temperature in a greenhouse. Acta Hortic. 907: 151–155.

Yasuba, K., Hoshi, T., Kaneko, S., Highashide, T., Omori, H., Nakano, A. 2013. Establishment of an environmental measurement node using open source hardware. (in Japanese with English summary) Agric. Info. Res. 22: 247–255.

Yasuba, K., Kurosaki, H., Takaichi, M., Suzuki, K. 2012. Development of a management software for protected horticulture based on the practical protocol of ubiquitous environment control system. (in Japanese with English summary) Bull. Natl. Inst. Veg. Tea Sci. 11: 63–72.