The 11th Asian-Australasian Conference on Precision Agriculture (ACPA 11)
October 14-16, 2025, Chiayi, Taiwan

LOW-CODE DEVELOPMENT ENVIRONMENT AND MIDDLEWARE FOR UBIQUITOUS ENVIRONMENT CONTROL SYSTEMS

Tsuneo Nakanishi[1]

[1] Fukuoka University, Japan

*Corresponding Author: tun@fukuoka-u.ac.jp

ABSTRACT

This work presents a low-code development environment that enables non-engineers to construct a customized software for UECS devices automating horticultural facilities as well as a middleware that provides a uniform application executing environment on different platforms for the UECS software.

Keywords: facility horticulture, automation, UECS, low-code development environment

INTRODUCTION

Emergence of commoditized microcontroller boards such as Arduino, Raspberry Pi, M5 Stack etc., enables even non-engineers to construct an inexpensive, customized automation systems for facility horticulture by connecting commercially available expanding boards, sensors, and actuators to them as instructed. Using the UECS, Ubiquitous Environment Control System (https://uecs.jp/, in Japanese), an open standard protocol for interconnection among different devices from different manufacturers, it is also possible to network these homemade and/or commercially available devices to achieve advanced automated control.

Software plays a crucial role for the automated control; however, customization of software requires a considerable amount of technical knowledge and experience unlike hardware, which customization can often be achieved with simple wiring. Using ready-made software is a feasible solution but lacks flexibility. Using libraries such as UARDECS (Yasuba et al., 2018) or a software framework promise both development efficiency and flexibility, although it is a high barrier for non-engineers.

This paper describes a low-code development environment to generate customized software for UECS-based automated control systems for facility horticulture, which is designed to be used by non-engineers who are not familiar with programming. The paper also describes middleware enabling the software generated by the environment to operate on different platforms.

LOW-CODE DEVELOPMENT ENVIRONMENT

The author's low-code development environment is based on Blockly (https://developers.google.com/blockly), a visual programming environment developed by

Google, which is also used in Scratch, a well-known programming education environment. Figure 1 (left) shows its snapshot. Users can generate software that implements their desired behavior by connecting functional blocks and configuring their parameters. While they can use blocks of control structures such as loops and branches, users can build software without needing to learn symbols or syntax like in programming languages and without requiring deep technical knowledge.

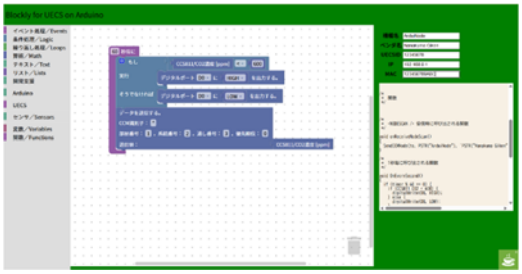
The author has added blocks to Blockly for handling operations and data exchange specified by UECS. Additionally, the author has implemented a code generator for C language, which is not provided by Blockly itself.

Users only need to describe the process they really want to realize with blocks. The low-code development environment never forces users to construct, with blocks, ancillary, technically required but ancillary codes for inserting header files, declaring necessary global variables or functions, and performing initialization. These ancillary codes are also injected at the same time when the block generates the codes for its original operation. The author's low-code development environment adds a dependency management mechanism to Blockly to inject these ancillary codes without duplication and in a valid order.

MIDDLEWARE FOR UECS

The author is also developing middleware to provide a uniform application executing environment, enabling software generated by the low-code development environment to run on different hardware platforms such as Arduino, Raspberry Pi, M5 Stack, and PCs running Windows or Linux. Figure 1 (right) shows its stacks. The majority of the middleware is codes for parsing and processing of XML-based messages defined by UECS.

For microcontroller boards such as Arduino, RAM capacity is quite limited (e.g., Arduino Uno R3 has only 2KB of RAM), while parsing and processing XML-based messages consuming memory. Memory efficiency is the primary concern in the design and implementation of this middleware. Therefore, the middleware avoids copying data in the message store in the communication buffer; instead, it directly references the data in the buffer. Furthermore, bit fields are used to minimize the number of bytes occupied by various data types. On the other hand, function callbacks are used to pass the results of message parsing to the application, namely the software generated by the low-code development environment and operated on this middleware, thereby eliminating platform dependency.



UECS Application		
UECS Platform Software Message Processor		
UECS Platform Software Message Parser		
UECS Platform SW Arduino-Dependent	UECS Platform SW Linux-Dependent	UECS Platform SW Windows-Dependent
Arduino Libraries	Linux	Windows
Arduino HW	Raspberry Pi HW	PC

Fig.1 Low-Code Development Environment (left) and Middleware (right)

REFERENCES

Yasuba, K., H. Kurosaki, T. Hoshi, T. Okayasu, Y. Tanaka, T. Goto, and Y. Yoshida. 2018. Development of program library using an open-source hardware for implementation of low-cost greenhouse environmental control system. Environ. Control Biol., 56(3): 107–112.