IoT Educational System on Mixed Reality Environments with Context-Oriented Programming

Ikuta Tanigawa[1,a) Harumi Watanabe[2] Nobuhiro Ohe[2] Mikiko Sato[2] Nobuhiko Ogura[3]
Takeshi Ohkawa[2] Kenji Hisazumi[1] Akira Fukuda[1]

Abstract: This article contributes to a step by step education for learning Internet of Things (IoT) developments. These steps are as follows: (1) Learning real-world problems, (2) Learning environmental adaptation for IoT, (3) creating service. To realize those steps, we introduce a mixed reality IoT educational system. Firstly, the system hides the complexity of the software and allows programming beginners to concentrate on the implementation of robot behavior. Secondly, the system supports Context-Oriented Programming (COP) for environmental adaptation of IoT. Finally, the system provides a projection mapping environment. To evaluate the system, we apply it to a class of programming.

Keywords: Context-Oriented Programming, Internet of Things, Mixed Reality

1. Introduction

In recent years, the importance of IoT education is increasing. The IoT isn't easy to learn for students because its knowledge is wide: from the mechanism of the things and the internet to service creation. Thus, we consider the step by step education is important. These steps are as follows: (1) Learning real-world problems, (2) Learning environmental adaptation for IoT, (3) creating service. This article introduces the IoT education system on mixed reality environments for the step by step education.

For the first step, the system hides the complexity of the software and allows programming beginners to concentrate on the implementation of robot behavior. For the second step, the system supports the extend Context-Oriented Programming (COP) [1], [2], [3], [4], [5] to C#. COP treats context explicitly and provides mechanisms to dynamically adapt behavior in reaction to changes in context at runtime. COP provides layers for modularizing related context-dependent behaviors. COP applications activate and deactivate these layers for adapting to various context. These functions facilitate the implementation of environmental adaptation for IoT. For the third step, the system provides a projection mapping environment. This environment is useful for creating IoT services and experimenting with them.

To evaluate the system, we applied it in a class of programming. Thirty-seven students participated in this class, and they divided with six groups. Students developed games with the robot and projection mapping, e.g., an action game, quiz game.

The remainder of this article describes step (2)(3). The article skips step (1) because it doesn't include technical issues. Section 3 describes the implementation of the educational system.

1 Kyushu Uniersity, 744, Motooka, Nishi-ku, Fukuoka, Japan
2 Tokai University, 2-3-23, Takanawa, Minato-ku, Tokyo, Japan
3 Tokyo City University, 3-3-1 Ushikubo-nishi, Tsuzuki-ku, Yokohama, Japan
a) tanigawa@f.ait.kyushu-u.ac.jp

2. IoT Educational System

In this section, we introduce our IoT educational system. Section 2.1 shows the COP program. Section 2.2 describes a mixed-reality system.

2.1 Step (2): COP Program

Figure 1 shows the COP program of controlling the robot on the projection mapping environment. The program of Figure 1 is based on ContextCS [6] that is extended C#. The behavior of the robot depends on floor colors. On the blue floor, the robot goes straight. On the green floor, the robot moves back and forth.

This program consists of two layers: *BaseLayer* and *LayerA*. The program changes the behavior of *BaseLayer* classes at the activation of *LayerA*. The first half in *MainLoop* activates and deactivates *LayerA* for changing the behavior of method *M1* of class *C1* according to floor colors. The rest of *MainLoop* calls the method *M1*. This method is changed behavior by activated layers.

2.2 Step (3): Outline of Mixed-Reality

Figure 2 shows the outline of the system. The system consists of two screens and projectors. Each screen is controlled by individual PCs. In this figure, the wall screen shows an image of the user interface of the tool. Students describe and run the program on the development tool displayed on the wall screen. We provide the library for controlling the actual robot and support COP.

Floor screen displays two-colored floor for the practice of IoT service. The vacuum cleaner robot reads floor colors by the color sensor on top. To solve the issue (3), students develop an application on projection mapping environment with the robot library and COP.

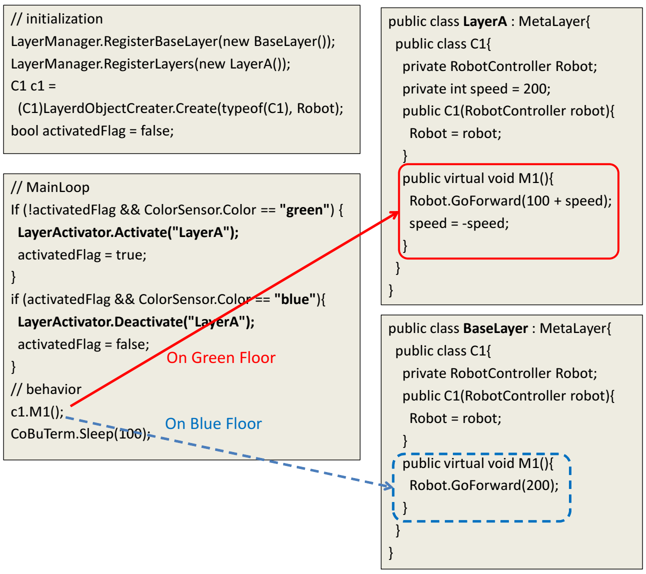

Fig. 1 COP Program for IoT Practice

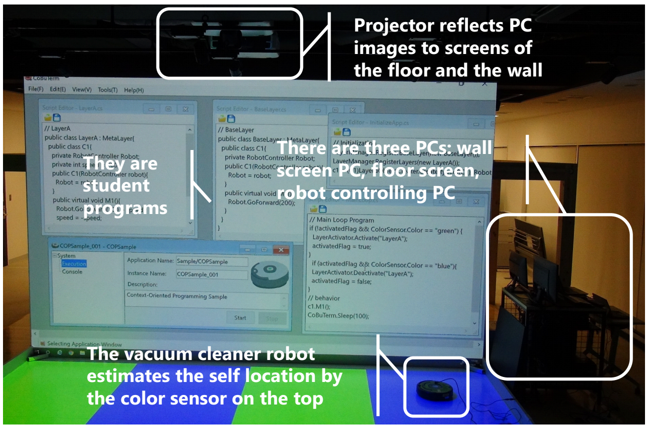

Fig. 2 Outline of Mixed-Reality

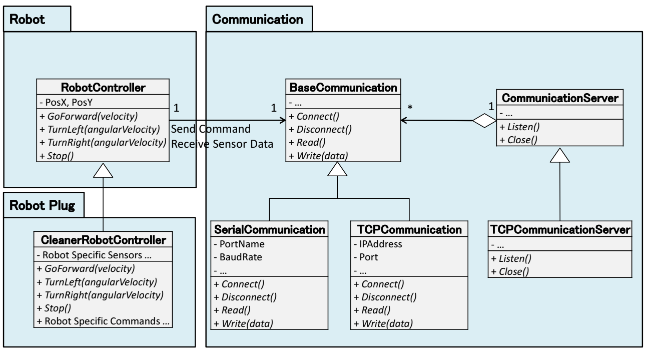

Fig. 3 Class Diagram of Robot API

3. Implementation

In this section, we describe the implementation of our IoT educational system. Figure 3 shows the class diagram of the robot API for its communication. In this design, users can choose several communication systems such as TCP/IP and serial communication. Figure 4 shows the relation between the user application and the robot framework. In this design, users can easily construct a robot system according to their purposes.

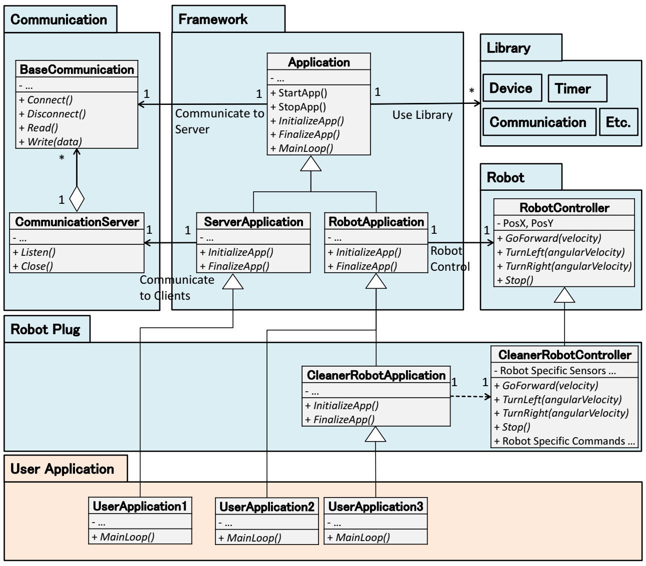

Fig. 4 Relation between user application and robot framework

4. Conclusion

This article introduced the mixed reality IoT educational system. The system aims to realize three steps for the education of Internet of Things (IoT) developments. These steps are as follows: (1) Learning real-world problems, (2) Learning environmental adaptation for IoT, (3) creating service. Firstly, the system hides the complexity of the software and allows programming beginners to concentrate on the implementation of robot behavior. Secondly, the system supports Context-Oriented Programming (COP) for environmental adaptation of IoT. Finally, the system provides a projection mapping environment. To evaluate the system, we applied it to a class of programming. Students developed games that were bigger than we imagined.

For future work, we plan to support the collaboration to cloud services for learning more practical IoT system development.

References

[1] R. Hirschfeld, P. Costanza and O. Nierstrasz. 2008. Context-oriented Programming. Journal of Object Technology, Vol. 7, No. 3, 125-151.

[2] G. Salvaneschi, C. Ghezzib and M. Pradella. 2012. Context-oriented Programming: A Software Engineering Perspective. Journal of Systems and Software, Volume 85, Issue 8, 1801-1817.

[3] M. Appeltauer, R. Hirschfeld and J. Lincke. 2013. Declarative Layer Composition with the JCop Programming Language. Journal of Object Technology, Vol. 12, No. 4, 4:1-37.

[4] M. Appeltauer, R. Hirschfeld, M. Haupt and H. Masuhara: ContextJ: Context-oriented Programming with Java, In proceedings of the JSSST Annual Conference 2009, pp. 1-15, (2009).

[5] P. Costanza and R. Hirschfeld. 2005. Language Constructs for Context-oriented Programming: An Overview of ContextL. In DLS '05: Proceedings of the 2005 symposium on Dynamic languages, 1-10.

[6] I. Tanigawa, N. Ogura, M. Sugaya, H. Watanabe, K. Hisazumi: A Structure of a C# Framework ContextCS based on Context-Oriented Programming, Proceedings of the 14th International Conference on Modularity, pp.21-22, 2015.