International Journal of Internet, Broadcasting and Communication Vol.16 No.4 226-234 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.4.226

IJIBC 24-4-26

Development of Cloud-based Smart Farm Management System while Considering Its Maintenance Aspects

Minseok Choi

Associate Professor, Div. of A.I Convergence, Sahmyook University, Korea
mschoi@syu.ac.kr

Abstract

Measures to enhance a cloud-based smart farm management system were proposed to improve its efficiency and maintenance. The existing system had achieved efficiency and stability by utilizing a web-based operating program with a general-purpose microcomputer and Linux. However, this system faced issues with synchronization and maintenance while concurrent tasks were being performed. Synchronization issues were solved by implementing an embedded DB, and the system was upgraded to allow over-the-air (OTA) software updates. Additionally, a method was also proposed to enable remote maintenance using tunneling. It was determined that applying the proposed method can contribute to the widespread adoption of smart farms, in addition to reducing maintenance costs. Furthermore, this system can also be expanded into a universal system applicable to different service models in the future.

Keywords: *Smart Farm, Cloud Farm, Intelligent Cultivation, Greenhouse, Remote Maintenance*

1. Introduction

The merging of information and communication technology (ICT) with other sectors has led to the emergence of various synergies. In particular, there is a growing need for advanced agricultural environments that integrate ICT to address the challenges posed by declining and aging rural populations and to enhance competitiveness [1,2]. Smart agriculture, which combines ICT and agriculture, has been garnering attention as a key driver of innovation and sustainable growth in agriculture [3]. Efforts are being made to pivot from traditional agriculture based on accumulated knowledge to intelligent agriculture that utilizes data. In addition, intelligent environments are being disseminated without spatiotemporal constraints through networks as a model for future agriculture [4]. Furthermore, the Internet of Things (IoT) has made it possible to collect and remotely monitor agricultural environment data, while the application of artificial intelligence (AI) can enhance the efficient management of resources and the overall quality of the agricultural system [5-8]. In addition, real-time event monitoring through various sensors and the analysis of collected data have provided efficient management tools for better decision-making and improved operations and management [9-11].

Manuscript Received: September. 30, 2024 / Revised: October. 5, 2024 / Accepted: October. 10, 2024
Corresponding Author: mschoi@syu.ac.kr
Tel: +82-2-3399-1560, Fax: +82-2-3399-1567
Associate Professor, Div. of A.I Convergence, Sahmyook University, Korea

Recently, there has been a growing interest in urban agriculture, as well as a desire for short weekend getaways to experience rural lifestyles. This has created a demand for small-scale, low-cost vertical smart farms that can use minimal space for agricultural activities [11]. For the mass production and distribution of these small-scale smart farms, it is crucial to design standardized hardware and implement efficient control systems. It is also necessary to consider the scalability and maintenance of the management system for different usage environments. Therefore, this study introduces a previously proposed cloud-based smart farm management system and presents measures to improve the management and maintenance of clients in a large-scale deployment environment.

This paper is structured as follows: Section 2 describes the previously implemented smart farm system, Section 3 discusses improvement measures for efficient management and maintenance, and Section 4 presents the results and findings.

2. Smart Farm Management System

2.1 Previous System Configuration

As shown in Figure 1(a), the existing cloud-based smart farm system consisted of a locally operated smart farm client system, a cloud management system that integrated and managed multiple local smart farms connected via the Internet and stored collected environmental data, and a user application.

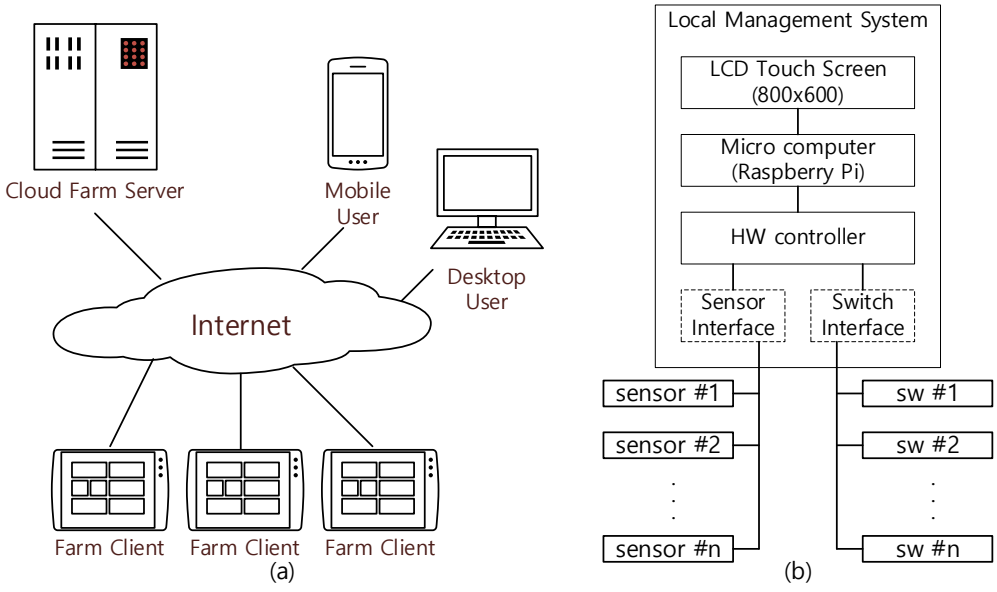

Figure 1. Configuration of the smart farm system [5]

The local smart farm system collected environmental data from unit facilities and controlled the cultivation environment. Its configuration is shown in Figure 1(b).

2.2 Local Management System

The local smart farm management system was designed to measure and collect various environmental data, such as illuminance, humidity, temperature, and CO_2 concentration, through sensor interfaces. It also controlled diverse environmental factors, such as lighting, heating, cooling, humidification, dehumidification,

circulation, and ventilation, through switch interfaces. As shown in Figure 2, the local smart farm unit was implemented as a closed-unit smart farm with a container structure to separate the growing conditions from the external environment.

(a) (b) (c)

Figure 2. Local smart farm unit: (a) outer view, (b) inner view, (c) management system[5]

The local management system used a Raspberry Pi 3 Model B+ as the microcomputer, and an 800 x 600-resolution LCD touchscreen was connected as an input/output device to implement the user interface. For stable hardware operation, a separate controller board was configured to control various sensor inputs and relay switches for device operation. This controller board was connected to the Raspberry Pi via a serial interface.

The program development environment for the local management system was set up with Linux-based Raspbian as the operating system, and MySQL was used as the database management system to store various settings and environmental data. Additionally, NGINX was used as the local web server to implement the web-based user interface (UI), and the web browser was run in kiosk mode to display the user interface on the LCD screen. The operating program for the system was web-based and implemented using PHP, HTML5, and JavaScript to improve the efficiency of the development work.

2.3 Cloud Management System

The cloud management system, which integrates and manages the local management systems, consists of a database server for storing environmental data and a web-based integrated management server. The basic server configuration was based on Amazon Web Services (AWS), Amazon's cloud computing solution. The database server used Amazon Aurora, a cloud-based MySQL-compatible relational database service with full MySQL compatibility, and the web-based management server was configured using Amazon EC2.

The cloud management system's database was designed to have the same structure as the local management system database, with the addition of a unique terminal number to distinguish each client system. Data transfer for synchronization from the local system to the cloud server used a REST architecture based on HTTP. Each terminal processed various information, such as real-time environmental data, environmental settings, and equipment operation specifications, in a JSON format and transmitted this information to the cloud web server, which synchronized the transmitted data with the database.

When a user remotely accesses the cloud server and operates the local system, the cloud server's settings need to be synchronized with the local management system. In this case, unlike the previous scenario, the synchronization data is transferred from the server to the client's local web server in a REST format, and the local web server synchronizes the transferred data with the system.

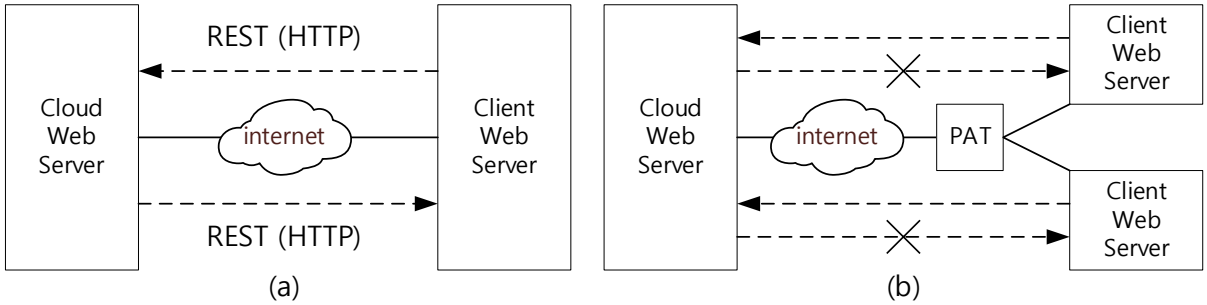

Figure 3. Architecture of synchronization between server and client

Figure 3 outlines the synchronization architecture between the server and the client. As shown in Figure 3(b), if the client is connected to a network address translation (NAT) or port address translation (PAT) device, such as a wired or wireless router, the nature of the HTTP protocol makes it impossible for the server to connect to the client. In most cases, local management systems do not have independent IP addresses and instead access the Internet through an IP router. Unless these issues are resolved, real-time synchronization via remote user access becomes difficult.

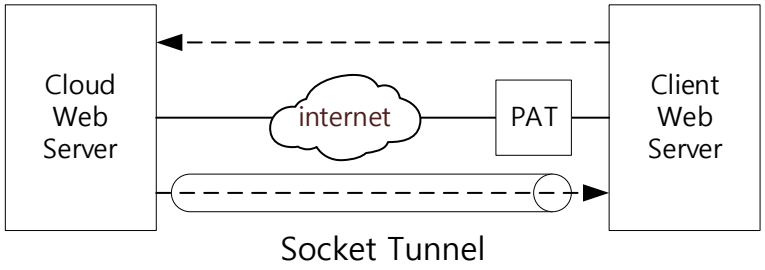

Figure 4. Socket Tunneling

The server-client synchronization issue was addressed by implementing socket tunneling from the client to the server, as shown in Figure 4. This setup enables the server to make a REST request via HTTP to the client through the tunnel, even if a PAT device exists. As a result, users can remotely execute commands through the cloud server. The commands are then synchronized and immediately executed on the local terminal.

3. Improving the Smart Farm Control System

The previous system was developed as a web application through a web server to ensure development efficiency and scalability. Due to the nature of web applications, synchronization issues may occur when multiple tasks are running simultaneously because unit tasks are performed asynchronously via a web server. In addition, as local smart farms are installed and operated in different regions, there is a need for a fast and efficient way to update the local management system to quickly respond to various operational requirements and issues that may arise in different operating environments. In such cases, there should be a means to remotely check and resolve issues without requiring physical visits.

3.1 Task and HW Synchronization

The local management system operates in an environment where various tasks can occur simultaneously.

These include tasks initiated by user interaction via the LCD screen, periodic background tasks registered in the system, and tasks requested from the cloud server. Due to the nature of web-based programs, tasks for each request are executed independently, and synchronization is not supported. For task synchronization, a file-based synchronization method can be used in PHP programs. However, since the client uses a DBMS for local data management, it is more reliable and efficient to implement task synchronization by storing synchronization information in the DB

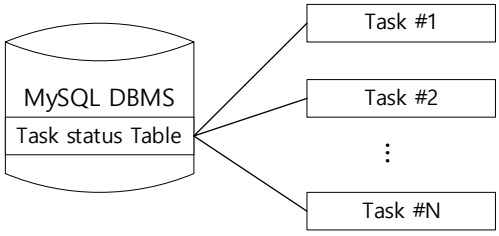

Figure 5. Synchronizing Tasks Using the DB

The local management system is connected to the HW switch to control external devices. If the power is interrupted during system operation, the HW switches will be reset. When the power is restored, the final switch state stored in the DB and the actual HW switch state will be different. To address this issue, a switch synchronization service was implemented to run when Linux boots.

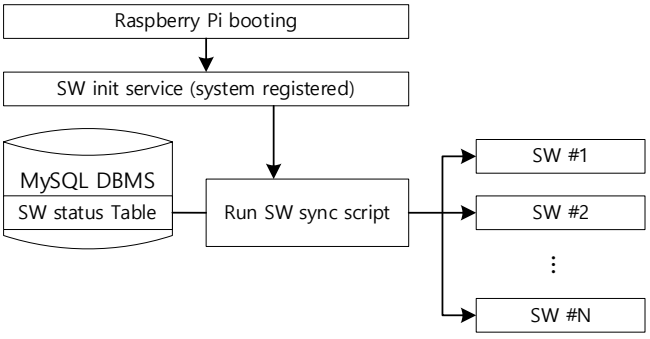

Figure 6. HW Switch Synchronization Service Structure

3.2 Software Fix and Updates

All Raspberry Pi devices use microSD cards as their primary storage device. The traditional method for distributing system software involves distributing the system's SD card image as a file and copying it to the SD card. However, this method has its drawbacks, such as the complexity of updating a new image and backing up existing data. Therefore, a user-friendly and efficient software distribution and update method is required, which is why this study proposes implementing over-the-air (OTA) software updates, as shown in Figure 7.

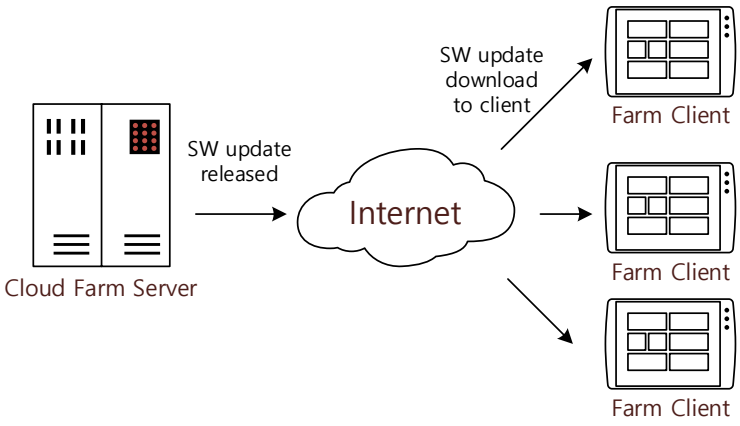

Figure 7. Software Update via OTA

In the case of web applications, program updates can be executed by copying added or modified files to the client's web document directory. However, updating individual files can complicate version control, so the entire file has been versioned. Due to the nature of web programming, the entire root directory of the web document can be compressed and distributed. By registering the compressed file on the server, as shown in Figure 8, the client can check the update information and proceed with the update. The server can also force the client to update if necessary.

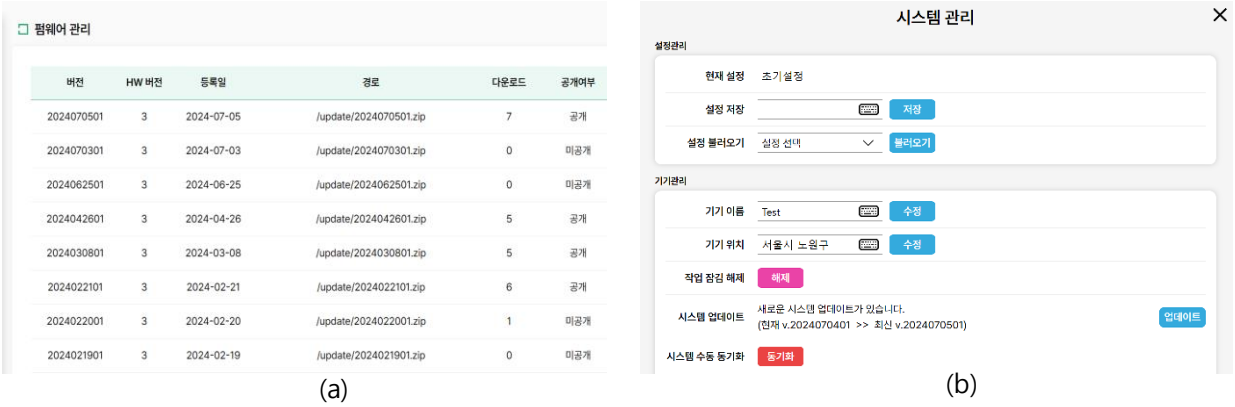

Figure 8. Software update: (a) management page on server, (b) setup page on client

There is also a need to implement a recovery function to prevent potential malfunctions or system freeze during the update process. The figure below illustrates the update process with a recovery function.

As shown in Figure 8(a), the downloaded update file is initially extracted to the update directory (www/update). Then, the web root directory (www/html), which contains the current version of the program, is moved to the backup directory (/www/old). After the update directory has been designated as the new web root, the update script is executed. Within the update script, any other necessary DB or system configuration changes are processed using shell scripts. Figure 8(b) shows the verification process of a program that is installed normally by the registered service upon booting Linux. If there are issues with the installation file due to an update or other causes, the previous version is reinstated first. Should there be any issues with the previous version, the program is switched to emergency recovery mode. In emergency recovery mode, after setting up the network, a simple menu is presented to proceed with software updates.

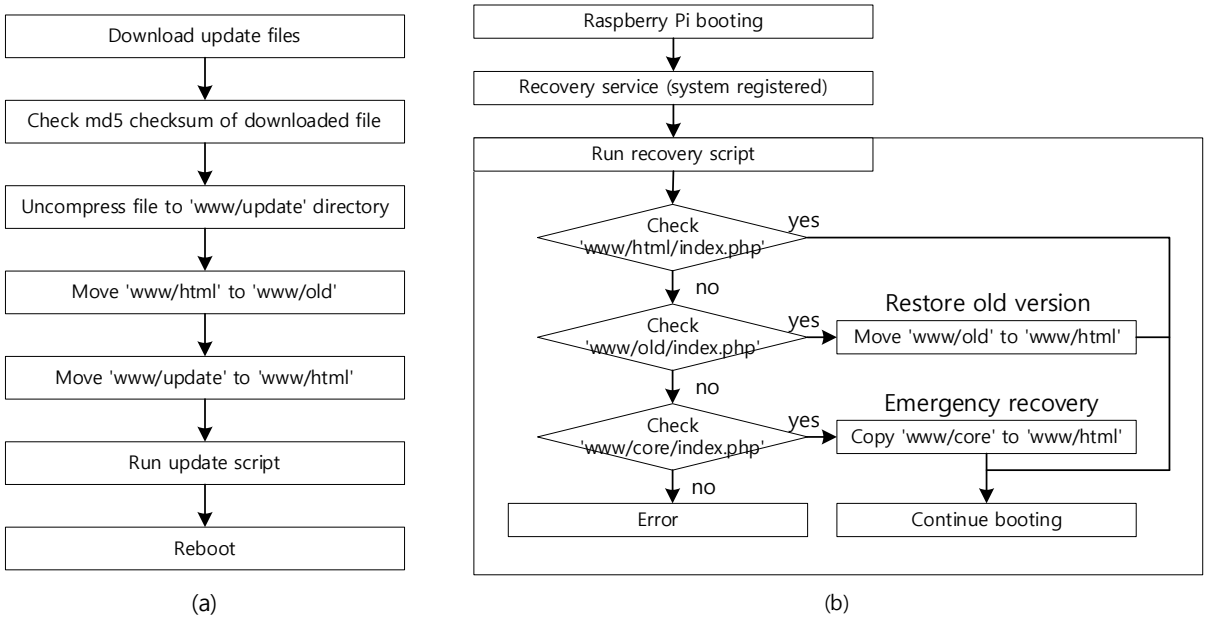

Figure 8. (a) Software update process, (b) recovery process

3.3 Remote Access Support

Small modular smart farms can be installed in different regions and exported to other countries. Maintenance issues for multiple smart farm clients installed nationwide could become a barrier to their widespread adoption. While hardware issues require on-site repairs, visiting each site to resolve operational issues and requirements can be costly. It is therefore necessary to monitor clients remotely and provide remote access to check and repair the system.

To facilitate maintenance, web-based DB management and maintenance functions have been implemented on the client to allow remote access to check the system's status and review various logs stored in the DB via the web. However, as described in Section 2.3, it is difficult to access the client's web server directly from the outside, and this can lead to security issues. Therefore, a secure shell (SSH) tunnel was created between the computer subject to maintenance and the cloud server as shown in Figure 10. Next, port forwarding was set up on the server to connect to the tunnel created for client access. This allows web access to the client through the server via the secure tunnel.

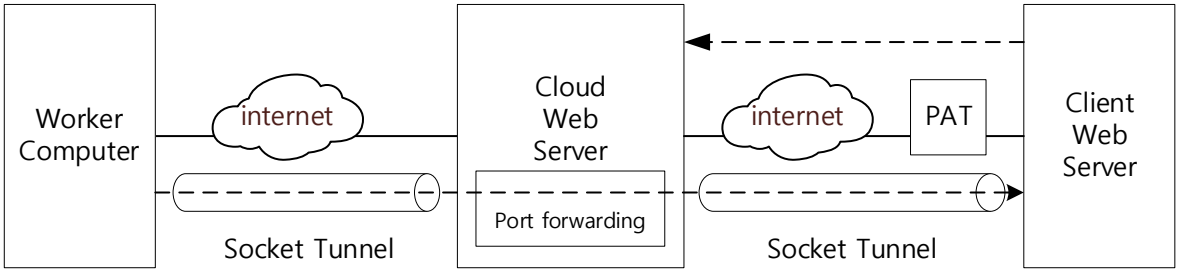

Figure 9. Client access via tunneling

If access to the entire client system is required, an additional SSH tunnel can be created from the client to the server to open direct SSH access to the client in the same way. Once SSH access is available, users can

view system-wide status and logs, including the operating system, and modify and test the program source directly. As shown in Figure 10, enabling remote maintenance via web and SSH access will reduce the burden of remote maintenance and increase the adoption of smart farm clients.

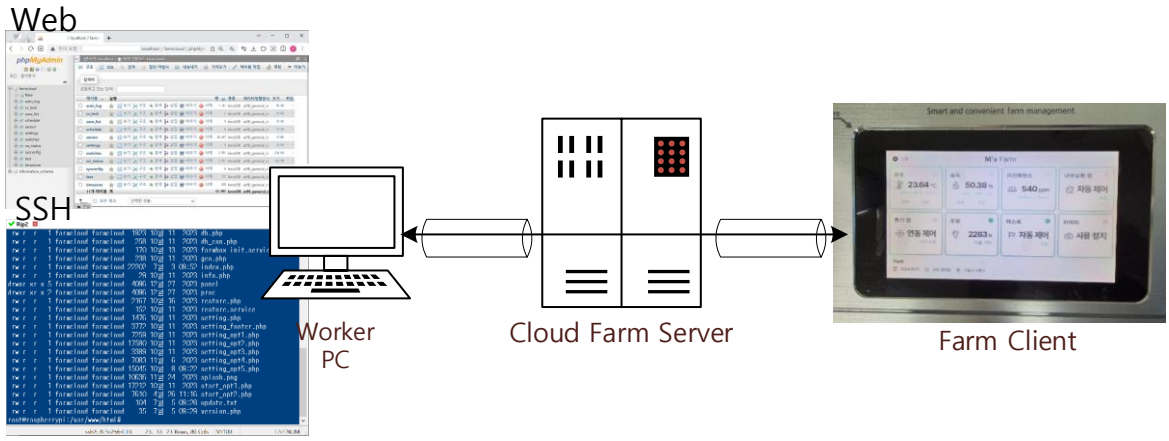

Figure 10. Maintenance client using the Web and SSH

4. Conclusion and Discussion

This study proposed measures to improve the efficient management and maintenance of a cloud-based smart farm control system, which is efficient for the distribution and operation of small-scale, standardized vertical smart farms. The existing system used Raspberry Pi (a general-purpose microcomputer) and a Linux-based OS to implement a local control system and introduced a web-based operating program to improve the efficiency and stability of development. The operation of individual programs using web-based applications and task scheduling in an operating system can improve stability, but it can also cause synchronization issues between tasks. Therefore, synchronization between individual programs has been efficiently solved using an embedded DB. A hardware synchronization service was also added to the Linux startup service to resolve status inconsistencies caused by hardware resets, such as power failures.

Maintenance issues for distributed terminals need to be considered to expand the distribution of smart farm clients and achieve nationwide distribution. To achieve this goal, this study proposed a method for implementing OTA software updates and enabling remote access maintenance to enable a rapid response to issues or new requirements. Adopting the proposed maintenance method is expected to expand the distribution of smart farm clients.

In recent years, various service models using kiosks have emerged. Since the proposed method is capable of implementing a local control system based on a kiosk connected to the cloud, it can be expanded into research on the design and implementation of a universal system applicable to various service models in the future.

References

[1] H. S. Kim, D. D. Lee and H. S. Kim, “Strategies and Tasks of ICT Convergence for the Creative Agriculture Realization(R736)”, Seoul: Korea Rural Economic Institute, 2014.

[2] Y. Lee and C. M. Heo, “A Study on the Influence of Acceptance Factors of ICT Convergence Technology on the Intention of Acceptance in Agriculture : Focusing on the Moderating Effect of Innovation Resistance”, Journal of

Digital Convergence, Vol. 17, No.9, pp. 115-126, 2019.
DOI: https://doi.org/10.14400/JDC.2019.17.9.115

[3] M. H. Ahn and C. M Heo, "The Effect of Technical Characteristics of Smart Farm on Acceptance Intention by Mediating Effect of Effort Expectation", Journal of Digital Convergence, Vol. 17, No. 6, pp. 145-157, 2019.
DOI: https://doi.org/10.14400/JDC.2019.17.6.145

[4] N. G. Yoon, J. S. Lee, G. S. Park and J. Y. Lee, "Korea smart farm policy and technology development status", Rural Resources, Vol. 59, No. 2, pp. 19-27, May 2017.

[5] Minseok Choi, "A study on the efficient Implementation method of cloud-based smart farm control system", Journal of Digital Convergence, Vol. 18, No. 3, pp. 171-177, 2019.
DOI: https://doi.org/10.14400/JDC.2020.18.3.000

[6] S. Qazi, B. A. Khawaja and Q. U. Farooq, "IoT-Equipped and AI-Enabled Next Generation Smart Agriculture: A Critical Review, Current Challenges and Future Trends", IEEE Access, Vol. 10, pp. 21219-21235, 2022.
DOI: https://doi.org/10.1109/ACCESS.2022.3152544

[7] Minseok Choi, "Smart Farm Management systemfor Improving Energy Efficiency", Journal of Digital Convergence, Vol. 19, No. 12, pp. 331-337, 2021.
DOI: https://doi.org/10.14400/JDC.2021.19.12.331

[8] Widianto, M., Ardimansyah, M., Pohan, H. and Hermanus, D, "A Systematic Review of Current Trends in Artificial Intelligence for Smart Farming to Enhance Crop Yield", Journal of Robotics and Control(JRC), Vol. 3, No 3, pp. 269-278, May 2022.
DOI: https://doi.org/10.18196/jrc.v3i3.13760

[9] Cambra Baseca, Carlos, Sandra Sendra, Jaime Lloret, and Jesus Tomas, "A Smart Decision System for Digital Farming", Agronomy Vol. 9, No. 5: 216, 2019.
DOI: https://doi.org/10.3390/agronomy9050216

[10] H. Y. Shin, H. K. Yim and W. T. Kim, "Intelligent Green House Control System based on Deep Learning for Saving Electric Power Consumption", Journal of IKEEE, Vol. 22, No. 1, pp. 53-60, 2018.
https://doi.org/10.7471/ikeee.2018.22.1.53

[11] Arshad, Jehangir, Musharraf Aziz, Asma A. Al-Huqail, Muhammad Hussnain uz Zaman, Muhammad Husnain, Ateeq Ur Rehman and Muhammad Shafiq, "Implementation of a LoRaWAN Based Smart Agriculture Decision Support System for Optimum Crop Yield", Sustainability, Vol.14, No. 2: 827, 2022.
DOI: https://doi.org/10.3390/su14020827

[12] Li Chen and Jae Eun Yoon, "Research on Spatial Layout Characteristics of Intelligent Farm", Design Research, Vol. 9, No. 2, pp.457-471, 2024.
DOI: https://doi.org/10.46248/kidrs.2024.2.457