J. Korean Inst. Electr. Electron. Mater. Eng.
Vol. 35, No. 5, pp. 509-515 September 2022
DOI: https://doi.org/10.4313/JKEM.2022.35.5.14
ISSN 1226-7945(Print), 2288-3258(Online)

Early Stage Report : Undergraduate Research

스마트팩토리를 위한 데이터 수집 관리 프로그램 개발

김현진, 김진사
조선이공대학교 자동화시스템과

Data Collection Management Program for Smart Factory

Hyeon-Jin Kim and Jin-Sa Kim
Department of Automatic System, Chosun College of Science & Technology, Gwangju 61453, Korea

(Received June 30, 2022; Revised July 5, 2022; Accepted July 6, 2022)

Abstract: As the 4th industrial revolution based on ICT is progressing in the manufacturing field, interest in building smart factories that can be flexible and customized according to customer demand is increasing. To this end, it is necessary to maximize the efficiency of factory by performing an automated process in real time through a network communication between engineers and equipment to be able to link the established IT system. It is also necessary to collect and store real-time data from heterogeneous facilities and to analyze and visualize a vast amount of data to utilize necessary information. Therefore, in this study, four types of controllers such as PLC, Arduino, Raspberry Pi, and embedded system, which are generally used to build a smart factory that can connect technologies such as artificial intelligence (AI), Internet of Things (IoT), and big data, are configured. This study was conducted for the development of a program that can collect and store data in real time to visualize and manage information. For communication verification by controller, data communication was implemented and verified with the data log in the program, and 3D monitoring was implemented and verified to check the process status such as planned quantity for each controller, actual quantity, production progress, operation rate, and defect rate.

Keywords: Smart manufacture, PLC, Cortex-m, Raspberry pi, Iot, Hmi

제조업 분야에서 ICT를 기반으로 하는 제 4차 산업혁명이 진행되면서 스마트팩토리 구축에 대한 관심이 크다. 특히 전세계가 인구노령화에 따른 숙련공의 부족과 코로나 19 사태로 제조업체의 경영난 등이 커지면서 스마트팩토리가 관심을 받고 있다. 고객수요에 따라 유연하고 맞춤형 생산이 가능한 스마트팩토리 생산방식은 고객 만족과 동시에 신규시장확보로 기업의 수익성을 증가시킬 뿐 만 아니라 비용절감에 따른 이익증대에도 기여할 것으로 예상되기 때문이다. 그러나 국내의 경우 스마트팩토리 전담 조직인 스마트 제조혁신기획단은 2022년도까지 스마트팩토리 3만개를 보급할 계획을 가지고 있으나 국내의 스마트팩토리 구축한 기업의 70%가 초기단계에 머물러 있는 실정이다 [1].

스마트팩토리가 일반화되기 위해서는 엔지니어와 장비들간 네트워크 통신을 통하여 실시간으로 자동화된 공정을 진행함으로서 공장의 효율성을 극대화하고 구축된 IT 시스템의 연계가 원활해야 한다. 특히, 이기종 설비에서 실시간 데이터를 수집하여 저장하고 방대한 양의 데이터를 분석하고 시각화하여 필요한 정보를 활용할 수 있어야 한다.

따라서 본 연구에서는 스마트팩토리 도입 전에 자동화 시스템이 구축되어 있는 경우가 많으므로 인공지능(AI), 사

✉ **Jin-Sa Kim; kimjs@cst.ac.kr**

물인터넷(IoT) 및 빅데이터(big data) 등의 연결을 통한 스마트팩토리 구축을 위하여 범용으로 사용되고 있는 PLC (MELSEC iQ-F FX5), 아두이노(Arduino), 라즈베리파이(Raspberry-Pi), 임베디드시스템(Embedded System) 등 4종의 제어기를 구성하여 실시간으로 데이터를 수집 및 저장하여 정보를 관리할 수 있는 프로그램을 개발하고자 한다.

스마트팩토리 도입 전에 자동화시스템이 구축되어 있는 경우가 많으므로 IT시스템과 연계하여 이기종 시스템의 데이터를 수집하여 관리하기 위하여 그림 1과 같이 제어기를 구성하였다. 개별산업에서 맞춤형 제품의 수요증가로 단일공정에서 연속공정으로 전환하기 위하여 PLC제어기를 채택하고 있다. PLC는 기존의 릴레이 제어반을 반도체 소자로 대체한 제어장치로 릴레이 시퀀스와 같은 복잡한 배선이 필요없이 프로그램을 통해 병렬처리 할 수 있는 편리한 개발환경을 지니고 있어 자동화시스템에 도입되고 있다 [2]. 본 연구에서는 PLC 제어기 중 MELSEC iQ-F FX5 시리즈를 사용하였으며 전압허용범위는 AC 85 V ~ 264 V으로 10ms의 순간정전에 대해서 동작을 계속할 수 있고, 소비전력은 30 W로 유지비용 절감의 효과가 있다.

사물인터넷, 3D프린팅 및 웨어러블 기술 등에 다양하게 적용하기 위하여 아두이노와 라즈베리파이가 많이 활용되고 있다. 각종 입출력장치와 인터넷, 스마트폰, TV 등에 연동하여 제어를 할 수 있기 때문이다. 아두이노는 윈도우와 리눅스 등의 OS환경에서 드라이버 설치가 가능하고 프로그래밍뿐만 아니라 디버깅도 가능하다. 또한 파이썬, C언어, BASIC, 델파이 등의 소프트웨어와 결합이 쉽다. 다양한 객체지향 라이브러리를 제공해주고 있고 거의 모든 센서를 사용할 수 있도록 지원하고 있어 많이 이용되고 있다. 본 연구에서는 디지털 입출력을 14개를 지원하고 6개의 아날로그 입력과 USB연결, 리셋버튼과 파워잭을 지원하는 아두이노를 사용하였다. 라즈베리 파이는 각종 센서나 저가의 인터넷 카메라 등의 사용할 수 있는 소형 저가형 컴퓨터로 영상처리나 딥러닝과 같은 인공지능등에도 활용이 가능하다 [3,4]. 또한 산업현장에서 임베디드시스템에 가장 많이 사용되는 있는 Cortex-M4을 사용하였고, Cortex-M4는 최소형으로 저비용의 장점이 있어 최저전력의 제어장치에 최적화되어 있다 [5].

PLC제어기로부터 데이터를 수집하기 위한 PLC제어기의 입출력 할당은 표 1에 나타내었고, PLC제어기의 입출력 상태를 확인하기 위하여 래더도를 작성하여 그림 2에 나타내었다.

Table 1. I/O allocation for PLC controller.

No	Digital in		Digital out	
	Address	Allocation	Allocation	Address
0	X0	D502.0	D500.0	Y0
1	X1	D502.1	D500.1	Y1
2	X2	D502.2	D500.2	Y2
3	X3	D502.3	D500.3	Y3
4	X4	D502.4	D500.4	Y4
5	X5	D502.5	D500.5	Y5
6	X6	D502.6	D500.6	Y6
7	X7	D502.7	D500.7	Y7
8	X8	D502.8	D500.8	Y8
9	X9	D502.9	D500.9	Y9
10	X10	D502.A	D500.A	Y10
11	X11	D502.B	D500.B	Y11
12	X12	D502.C	D500.C	Y12
13	X13	D502.D	D500.D	Y13
14	X14	D502.E	D500.E	Y14
15	X15	D502.F	D500.F	Y15

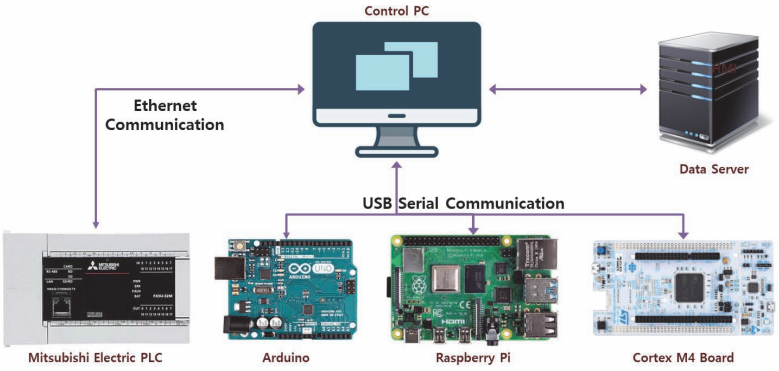

Fig. 1. Configuration of controller.

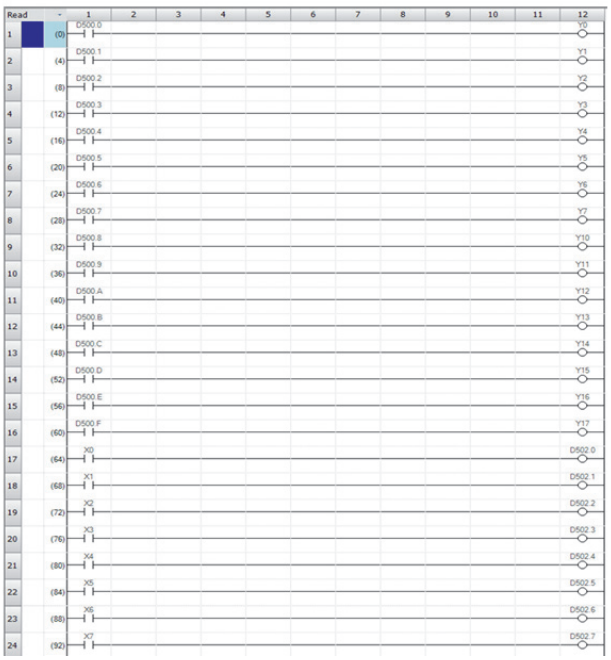

Fig. 2. Ladder program for PLC control.

Table 2. I/O allocation for Arduino controller.

No	Digital in		Digital out	
	Address	Allocation (Binary signal)	Allocation (Binary signal)	Address
0	D2	0 or 1	0 or 1	D8
1	D3	0 or 2	0 or 2	D9
2	D4	0 or 4	0 or 4	D10
3	D5	0 or 8	0 or 8	D11
4	D6	0 or 16	0 or 16	D12
5	D7	0 or 32	0 or 32	D13

No	Analog in	
	Address	Allocation (10 Bit)
0	A0	0 ~ 1023

```
#include <Arduino.h>
uint8_t currentDigitalInput = 0;
uint8_t prevDigitalOutput, currentDigitalOutput = 0;
uint8_t tempBuffer[200];
String recivedata;
//PacketSerial myPacketSerial;
uint8_t ReadDigitalPins(void);
void WriteDigitalPins(uint8_t data);
void onPacketReceived(const uint8_t* buffer, size_t size);

}

void loop() {
  currentDigitalInput = ReadDigitalPins();
  Serial.print(currentDigitalOutput);
  Serial.print(",");
  Serial.print(currentDigitalInput);
  Serial.print(",");
  Serial.print(analogRead(0));
  Serial.print("₩r");
  recivedata = Serial.readString();
  if(recivedata != ""){
    currentDigitalOutput = recivedata.toInt();
    WriteDigitalPins(currentDigitalOutput);  }
  delay(100);
}
uint8_t ReadDigitalPins(void){
  uint8_t data=0;
  data = (digitalRead(2) & 0x01);
  data |= (digitalRead(3) & 0x01)<<1;
  data |= (digitalRead(4) & 0x01)<<2;
  data |= (digitalRead(5) & 0x01)<<3;
  data |= (digitalRead(6) & 0x01)<<4;
  data |= (digitalRead(7) & 0x01)<<5;
  return data; }

void WriteDigitalPins(uint8_t data){
  digitalWrite(8, data & 0x01);
  digitalWrite(9, (data>>1) & 0x01);
  digitalWrite(10, (data>>2) & 0x01);
  digitalWrite(11, (data>>3) & 0x01);
  digitalWrite(12, (data>>4) & 0x01);
  digitalWrite(13, (data>>5) & 0x01); }

void onPacketReceived(const uint8_t* buffer, size_t size){
  int sendsize = 10;  }
```

Fig. 3. Program for I/O control.

아두이노 제어기로부터 데이터를 수집하기 위하여 제어기의 입출력 할당은 표 2에 나타내었고, 입출력 제어를 위한 프로그램은 그림 3에 나타내었다.

라즈베리파이 제어기로부터 데이터를 수집하기 위하여 제어기의 입출력 할당은 표 3에 나타내었고, 입출력 제어를 위한 프로그램은 그림 4에 나타내었다.

임베디드 제어기로부터 데이터를 수집하기 위하여 제어기의 입출력 할당은 표 4에 나타내었고, 입출력 제어를 위한 프로그램은 그림 5에 나타내었다.

스마트팩토리의 데이터 수집관리를 위하여 제어기에서 보낸 데이터를 받기 위한 수신부 프로그램은 그림 6과 같다.

스마트팩토리를 위한 데이터를 수집하고 관리하기 위하여 제어기들은 C# 프로그램을 통하여 제어기에서 보낸 데이터를 받도록 하였으며 프로그램 화면 구성은 그림 7에 나타내었다.

데이터 수집관리를 위하여 프로그램에서 제어기쪽으로 송신하기 위한 송신부는 프로그램내에서 체크박스를 누르면 실행되어 확인할 수 있도록 하였으며 송신부 프로그램은 그림 8과 같다.

스마트팩토리를 구현하기 위하여 데이터 수집관리 프로그램의 데이터 저장 및 송신부와 UI는 그림 9에 나타내었고, 각 제어기에서 받은 데이터를 배열 버퍼에 저장한 후 버퍼의 데이터를 프로그램 UI (data Log.) 및 서버로 송신하도록 하였다.

제어기별 통신검증을 위하여 그림 9와 같이 프로그램 내 Data Log로 데이터 통신을 구현하여 검증하였으며 제어기별 계획수량, 실적수량, 생산진도, 가동률, 불량률 등의 공정상태를 그림 10과 같이 3D 모니터링을 구현하여 검증하였다.

Table 3. I/O allocation for Raspberry-Pi control.

No	Digital in		Digital out	
	Pin number	Allocation (Binary signal)	Allocation (Binary signal)	Pin number
0	No. 7	0 or 1	0 or 1	No. 29
1	No. 13	0 or 2	0 or 2	No. 31
2	No. 15	0 or 4	0 or 4	No. 32
3	No. 16	0 or 8	0 or 8	No. 33
4	No. 18	0 or 16	0 or 16	No. 37
5	No. 22	0 or 32	-	-

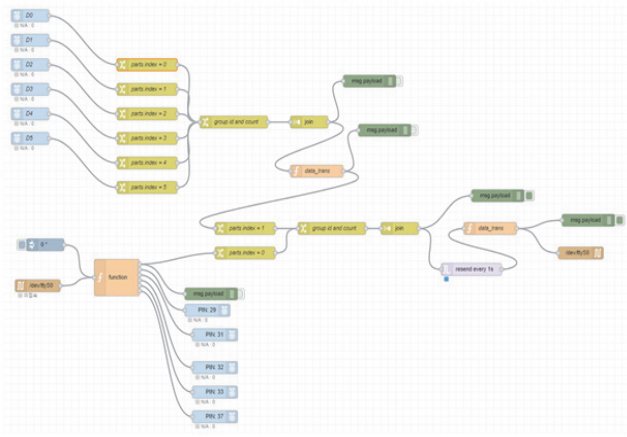

Fig. 4. Program for for Raspberry-Pi control.

```
#include "string.h"
  uint8_t rx_Data=0;
  uint8_t rx_Data_Buffers[100];
  uint8_t rx_Data_Buffers_counter=0;
  uint8_t tx_Data_Buffers[100];
  GPIO_PinState INPUT_state[6];
  GPIO_PinState OUTPUT_state[6];
  uint8_t input_data = 0;
  uint8_t output_data = 0;
  uint16_t adc_data = 0;
  HAL_UART_Receive_IT(&huart3, &rx_Data, 1);
while (1){
      INPUT_state[0] = HAL_GPIO_ReadPin(D2_GPIO_Port, D2_Pin);
      INPUT_state[1] = HAL_GPIO_ReadPin(D3_GPIO_Port, D3_Pin);
      INPUT_state[2] = HAL_GPIO_ReadPin(D4_GPIO_Port, D4_Pin);
      INPUT_state[3] = HAL_GPIO_ReadPin(D5_GPIO_Port, D5_Pin);
      INPUT_state[4] = HAL_GPIO_ReadPin(D6_GPIO_Port, D6_Pin);
      INPUT_state[5] = HAL_GPIO_ReadPin(D7_GPIO_Port, D7_Pin);
      input_data = 0;
      input_data = INPUT_state[0] | INPUT_state[1] << 1 | INPUT_state[2] << 2
            | INPUT_state[3] << 3 | INPUT_state[4] << 4
            | INPUT_state[5] << 5;
      OUTPUT_state[0] = output_data & 0x01;
      OUTPUT_state[1] = output_data>>1 & 0x01;
      OUTPUT_state[2] = output_data>>2 & 0x01;
      OUTPUT_state[3] = output_data>>3 & 0x01;
      OUTPUT_state[4] = output_data>>4 & 0x01;
      OUTPUT_state[5] = output_data>>5 & 0x01;
      HAL_GPIO_WritePin(D8_GPIO_Port, D8_Pin, OUTPUT_state[0]);
      HAL_GPIO_WritePin(D9_GPIO_Port, D9_Pin, OUTPUT_state[1]);
      HAL_GPIO_WritePin(D10_GPIO_Port, D10_Pin, OUTPUT_state[2]);
      HAL_GPIO_WritePin(D11_GPIO_Port, D11_Pin, OUTPUT_state[3]);
      HAL_GPIO_WritePin(D12_GPIO_Port, D12_Pin, OUTPUT_state[4]);
      HAL_GPIO_WritePin(D13_GPIO_Port, D13_Pin, OUTPUT_state[5]);
      HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
      adc_data = HAL_ADC_GetValue(&hadc1);
      HAL_ADC_Stop(&hadc1);
      sprintf(tx_Data_Buffers, "%d,%d,%d\r", output_data, input_data, adc_data);
      HAL_UART_Transmit(&huart3, tx_Data_Buffers, strlen(tx_Data_Buffers), 10);
      HAL_Delay(1000);
 } }
 static void recieveRXDataEvent(uint8_t *date){
    output_data=atoi(date);   }
 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *uartHandle){
   if(uartHandle->Instance == huart3.Instance){
        rx_Data_Buffers[rx_Data_Buffers_counter] = rx_Data;
        if(rx_Data == '\r'){
            rx_Data_Buffers[rx_Data_Buffers_counter+1] = '\0';
            recieveRXDataEvent(rx_Data_Buffers);
            rx_Data_Buffers_counter=0; }
        else{ rx_Data_Buffers_counter++; }
   HAL_UART_Receive_IT(&huart3, &rx_Data, 1);  } }
```

Fig. 5. Program for control embedded system.

Table 4. I/O allocation for embedded controller.

No	Digital in			Digital out		
	Pin number /User label	Pin name	Allocation (Binary signal)	Allocation (Binary signal)	Pin name	Pin number /User label
0	D2	PF15	0 or 1	0 or 1	PF12	D8
1	D3	PF13	0 or 2	0 or 2	PD15	D9
2	D4	PF14	0 or 4	0 or 4	PD14	D10
3	D5	PF11	0 or 8	0 or 8	PA7	D11
4	D6	PF9	0 or 16	0 or 16	PA6	D12
5	D7	PF13	0 or 32	0 or 32	PA5	D13

No	Analog in	
	Address	Allocation (10 Bit)
0	A0	0 ~ 1,023

ICT를 기반으로 하는 제4차 산업혁명이 진행되면서 고객수요에 따라 유연하고 맞춤형 생산이 가능한 스마트팩토리 구축이 일반화될 전망이다. 이를 위해서는 장비들간 네트워크 통신을 통하여 실시간으로 자동화된 공정을 진행하여 공장의 효율성을 극대화하고 구축된 IT 시스템의 연계가 가능하여야 한다. 또한 이기종 설비들에서 실시간 데이터를 수집하여 저장하고 방대한 양의 데이터를 분석하여 시각화하여 필요한 정보를 활용할 수 있어야 한다. 따라서 본 연구에서는 인공지능, 사물인터넷, 빅데이터 등의 기술 연결이 가능한 스마트팩토리 구축을 위하여 범용으로 사용되고 있는 PLC, 아두이노, 라즈베리파이, 임베디드시스템 등 4종의 제어기를 구성하여 실시간으로 데이터를 수집하고 저장하여 정보를 시각화하여 관리할 수 있는 프로그램개발에 관하여 연구하였다. 제어기별 통신검증을

```
private void OnSerialDataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        if (AllDeviceCom.SerialPort[(int)type].BytesToRead > 0)
        {
            byte[] buffer = new byte[AllDeviceCom.SerialPort[(int)type].BytesToRead];
            AllDeviceCom.SerialPort[(int)type].Read(buffer, 0, buffer.Length);
            ReceiveBuffer.AddRange(new List<byte>(buffer));
        }
    }
```

Fig. 6. Program of receive part for data collection of controller.

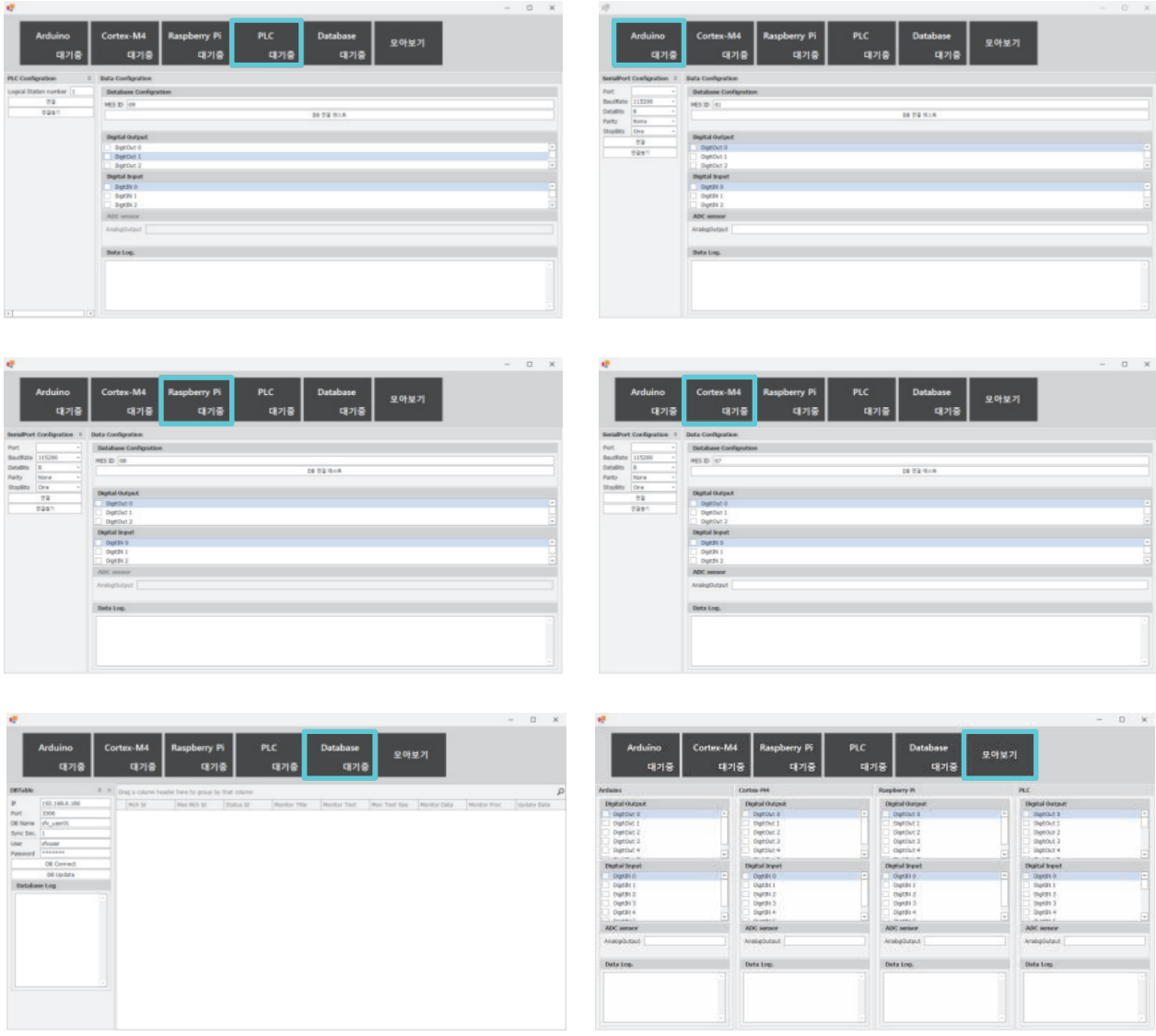

Fig. 7. Screen composition for data collection of controller.

```
private void OutputItemCheckChanged(object sender,
DevExpress.XtraEditors.Controls.ItemCheckEventArgs e)
        {
            CheckedListBoxControl checkedListBox = (CheckedListBoxControl)sender;
            DigitalData item = (DigitalData)checkedListBox.GetItem(e.Index);
            SetDataLogMemoEditText($"The {item.name} item with the {item.id} value is
{item.pinState}.");
            //digitalInputDataArray[e.Index].pinState = item.pinState;
            plc_output_data = 0;

            for (int num=0; num< DataModel.digitalOutputDataArrayList[(int)type].Length;
num++)
            {
                if(DataModel.digitalOutputDataArrayList[(int)type][num].pinState)
                {
                    plc_output_data |= 1 << num;
                }
            }
            if(PLCConDockPanel.Visibility ==
DevExpress.XtraBars.Docking.DockVisibility.Visible)
            {
                int ret = AllDeviceCom.ActUtlType.SetDevice("D500", plc_output_data);
```

Fig. 8. Program of transmitter part for sending data to the controller.

Fig. 10. Implementation of 3D modeling for verification.

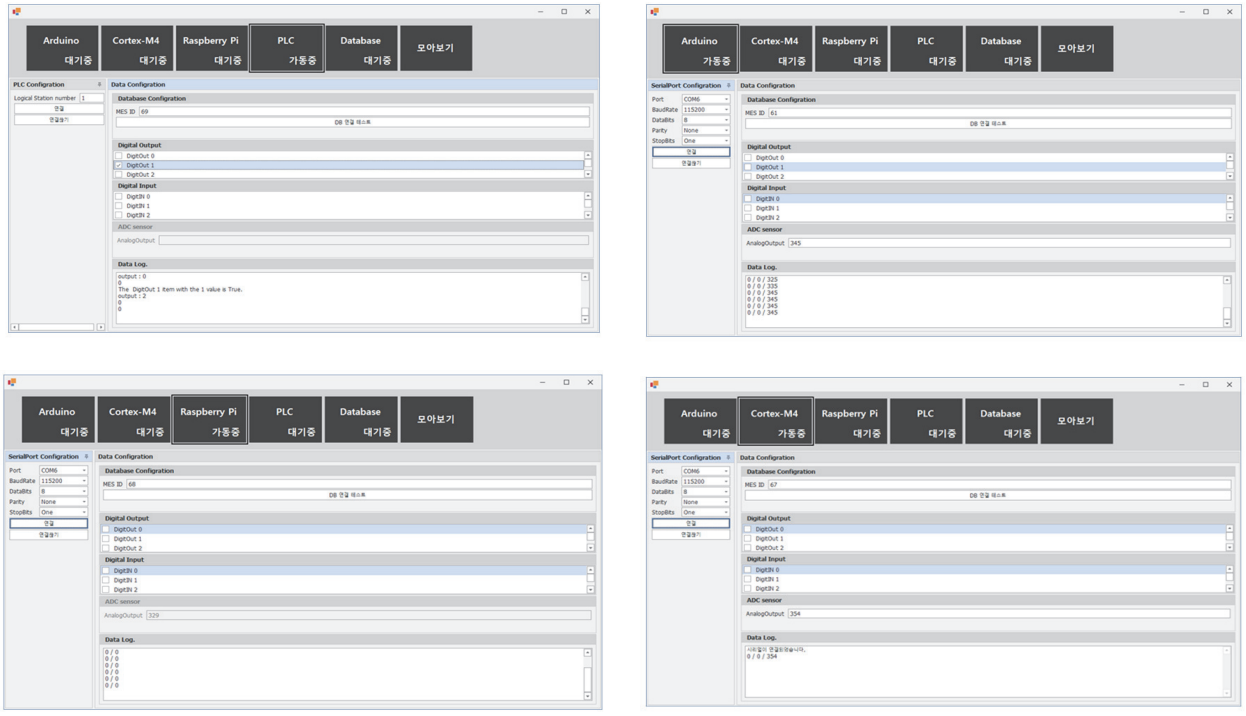

Fig. 9. Implementation of Program for data collection and management.

위하여 프로그램 내 Data Log로 데이터 통신을 구현하여 검증하였으며 제어기별 계획수량, 실적수량, 생산진도, 가동률, 불량률 등의 공정상태를 확인하기 위하여 3D 모니터링을 구현하여 정보를 시각화하여 관리할 수 있도록 하여 검증하였다. 향후 데이터 수집 및 관리를 통하여 빅데이터 관리와 AI 솔류션과 연계할 수 있을 것이다.

ORCID

Jin-Sa Kim https://orcid.org/0000-0001-9316-3092

REFERENCES

[1] H. Kim, H. Huh, J. W. Kang, and J. Boo, *J. Soc. Korea Ind. Syst. Eng.*, **42**, 252 (2019). [DOI: https://doi.org/10.11627/jkise.2019.

42.3.252]

[2] S. B. Lee, T. H. Park, and K. S. Han, *J. Inst. Control Rob. Syst.,* **20**, 741 (2014). [DOI: https://doi.org/10.5302/J. ICROS.2014. 13.1969]

[3] E. S. Lee, *The Journal of Korean Practical Arts Education*, **27**, 1 (2021). [DOI: https://doi.org/10.29113/skpaer.2021.27.1.001]

[4] H. W. Park and Y. W. Shin, *Journal of The Korea Society of Computer and Information*, **25**, 17 (2020). [DOI: https://doi.org/ 10.9708/jkscr.2020.25.11.017]

[5] W. Y. Lee, D. S. Son, J. J. Oh, and J. H. Yu, *J. Pract. Eng. Educ.*, **11**, 283 (2019). [DOI: https://doi.org/10.14702/JPEE.2019.283]