공 학 석 사 학 위 논 문

임베디드 보드와 오픈 소스 소프트웨어를 이용한 산업용 장비의 OPC-UA 통신

2021년 8월

부 경 대 학 교 산 업 대 학 원

기 계 시 스 템 및 조 선 공 학 과

채 수 인

공 학 석 사 학 위 논 문

임베디드 보드와 오픈 소스 소프트웨어를 이용한 산업용 장비의 OPC-UA 통신

지도교수 장 지 성

이 논문을 공학석사 학위논문으로 제출함.

2021년 8월

부 경 대 학 교 산 업 대 학 원

기 계 시 스 템 및 조 선 공 학 과

채 수 인

채 수 인의 공학석사 학위논문을인준함

2021 년 8 월 27 일

주 심 지 상 원 (인)

위 원 정 석 호 (인)

위 원 장 지 성 (인)

목 차

List of Figures

List of Tables

OPC-UA communication of industrial equipment using embedded board and open source software

Su-In Chae

Department of Mechanical Systems and Naval Architecture Engineering, Pukyong National University Graduate School of Industry

Abstract

Recently, various software and hardware technologies are converging through the 4th industrial revolution and the distribution and spread of smart factories that can be intellectualized and optimize the entire production process for the innovation of the manufacturing industry are being made widely. In addition, the need for standard communication among equipments for the convergence of SW technology and hardware device in the manufacturing process is increasing and according to this trend, various communication systems and application technologies are being studying. Specially, communication operating system based on OPC-UA(Open Platform Communication Unified Architecture) is in the spotlight. On this study, OPC-UA communication platform applied to open source software based on Raspberry-pi embedded board is developed and tested the performance evaluation in real industrial systems to enable the data exchange mutually in the various industrial devices and software environment used in industrial field smart manufacturing environment.

This study consists of five chapters and the summary of each chapter is as follows.

On the first chapter, it explains the problems with the configuration of the communication protocol for each manufacturer and the controllers

provided by manufacturers in the industrial field, and the necessity of industrial protocol integration. Furthermore, the definition and configuration of OPC communication is also explained. On the second chapter, it explains the configuration of OPC software that has been commercialized and sold and it contains the outline of the Open62541 OPA-UA library. On the third chapter, it explains the configuration of the OPC-UA test bed developed with the embedded board and open source software. In addition, 4 types of test items are selected and then tested and verified the developed OPC-UA server and 6 types of PLC for each manufacturer.

제 1 장 서 론

제조업 혁신을 위해 생산 전 과정의 지능화 및 최적화가 가능한 스마트 공장(smart factory)의보급 및 확산이 널리 이루어지고 있다. 이와 더불어 제조 공정에서의 원활한 SW기술 융복합을 위해 기기간 표준 기반 통신의 필요성이 높아지고 있으며, 이러한 추세에 따라 OPC-UA(Open Platform CommunicationUnified Architecture)가 주목을 받고 있다[1]. OPCUA는 IEC 62541 표준[2]에 의해 명세되어 있으며 자동화시스템에 구성되는 서로 다른 다양한 제어 디바이스 장치들이 데이터 교환을 가능하게 한다. OPC-UA의 플랫폼 독립적인 아키텍쳐는 임베디드 플랫폼부터 고성능 시스템까지 다양한 규모의 시스템이 상호 접근 가능한 환경을 제공한다[3]

본 논문은 산업현장 스마트제조 환경에서 사용 되어지는 다양한 산업용 장비 및 소프트웨어 환경에서 상호 데이터 교환을 가능하게 하기위해 Raspberrypi 임베디드 보드기반 오픈 소스 소프트웨어를 적용한 OPC-UA 통신 플랫폼을 개발하여 실제 산업용 시스템에서의 성능평가를 실험하였다.

1.1 산업용 프로토콜 통합의 필요성

산업용 장비는 Mitsubishi Electric, Siemens, LS Electric, Allen-Bradley 등 여러 종류의 장비가 각자의 독자적인 통신규격을 갖고 설비간의 통신이나 PC 응용프로그램과의 통신에 사용되고 있다. 이렇게 사용할 경우, 여러 가지 문제가 발생할 가능성이 예상된다.

첫째. 산업현장에서 규격 통일로 인한 매몰비용이 발생한다. 예를 들어, A

사의 제어장비로 공장을 구성했을 경우, 차후에 제어장비를 살 때 B사의 제어장비가 더 좋은 성능과 가격을 갖고 있음에도 불구하고 기존에 A사의 장비로 시스템을 구축해 놓았기 때문에 이종간 통신이 되지 않는다면 A사의 제품으로 시스템을 계속 구축해갈 수 밖에 없다.

둘째. 매몰비용과 동일한 문제로, 장비 제조사가 도산하거나 수입에 문제가 생기거나 할 경우 기존 시스템의 확장이 불가능하게 되며, 장비에 문제가 생길 경우 시스템의 일부 교체가 아닌 전체를 교체해야 하는 대 작업이 필요하게 될 수도 있다.

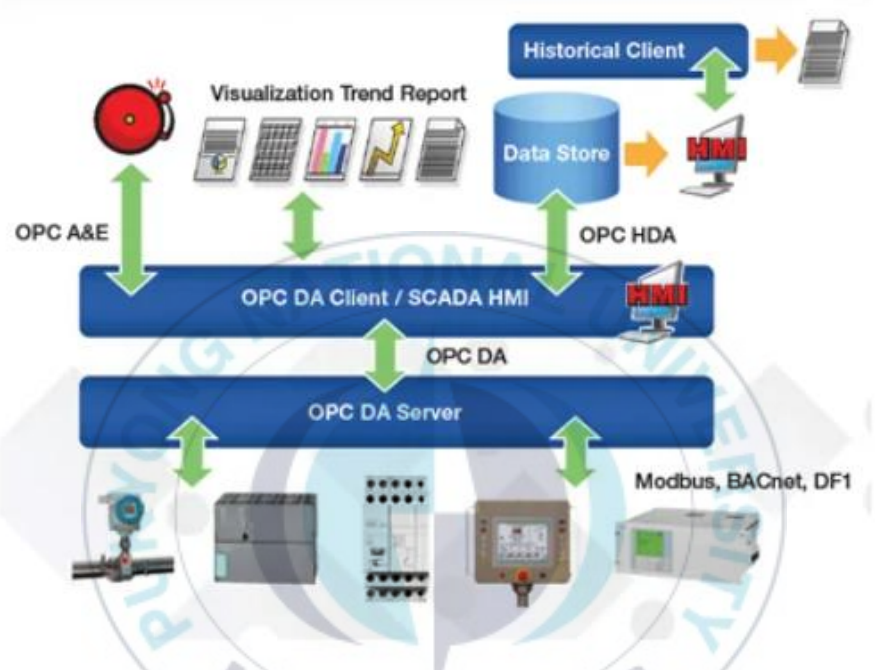
마지막으로, 빅데이터 구축등을 목표로 클라우드에 데이터를 수집하거나 혹은 인수합병한 공장을 모니터링 시스템에 포함시키거나 할 경우, 통신 규격에 따라 어플리케이션을 추가로 확장해야 할 필요성이 생긴다. 그만큼 개발 비용 및 시간이 추가로 들어가게 된다.

최근에 나오는 제어장비들은 여러 프로토콜을 지원하고 그 중엔 Modbus/TCP, EtherNet/IP와 같은 산업용 범용 프로토콜을 지원한다는 경우도 많으나, 기존에 구축된 설비들 같은 경우 프로토콜을 추가로 확장하는 것이 쉽지 않거나 불가능하다. 따라서 산업용 표준 프로토콜을 제시하되, 기존의 프로토콜을 표준 프로토콜로 변환하는 방법 또한 함께 연구되어야 할 것이다.

1.2 OPC-DA(OPC Classic)

OPC는 산업자동화 분야의 각 업계로부터 안전하고 신뢰성있는 데이터 교환을 목적으로 상호운용을 위한 표준규격이다. 1996년 첫 표준규격이 발표 되었으며, 기존에 개발된 PLC고유의 프로토콜(Modbus, Profibus등)을

표준화된 인터페이스로 추상화 하는 것이 목적이었다. 결과, 최종사용자(End-User)는 각 PLC에 일일이 접근하지 않아도 추상화된 데이터 교환이 가능해 졌다. Fig. 1.1은 산업현장에서 운영되어 지는 OPC 클래식 아키테처 구성 예시를 나타내고 있으며 서버와 연결된 산업용 장비 (PLC, 센서 계측장비)와 클라이언트를 통해 구성되는 Data Store 구성을 확인할 수 있다.



Fig, 1.2 OPC Classic Architectures

초기의 OPC(현재의 OPC Classic)는 프로토콜 통합의 목적을 가지고 탄생했고 실제로 그 역할을 수행해 왔으나, 실제 산업에 적용해 본 결과 아래와 같은 이슈들이 발생하였다.

- 플랫폼 의존성: OPC Classic의 통신 방식은 Microsoft Windows의 DCOM(Distributed Component Object Model)기술에 의존며. 때문에 서버와 클라이언트 양측의 Application이 윈도우 환경에서 작동해야만 하는 문제점이 확인되었다..

- 보안성 : 2000년 1월, 오스트레일리아 퀸즐랜드 주의 마루치 지자체에서

발생한 하수 처리 시스템에 대한 사이버 공격에 의해 하수 처리 시설이 오동작 하는 사건이 있었던 이후로[4], 산업 현장의 제어 시스템에 대한 보안 이슈가 급증하기 시작 하였으며 OPC Classic 은 자체적인 보안 규격을 갖고 있지 않기에, 보안 이슈에 대해서는 각 어플리케이션 별로 대응하는 문제점이 확인되었다. 이는 어플리케이션 간의 호환성을 크게 해치게되었다.

- 방화벽을 통한 외부 접근: DCOM의 통신 방식은 서버측에서 동적 포트를 할당해 주는 방식으로, 정책적으로 접근 가능한 포트를 관리하는 방화벽을 통과하기 어렵다. 그러나 IoT, 클라우드 방식의 빅 데이터 수집 등 산업현장에서 인터넷을 통한 정보 교환 요구가 늘어나고 있어 방화벽을 통한 외부에서의 안전한 데이터 엑세스가 필요하다는 문제점이 확인되었다.

1.3 OPC-UA

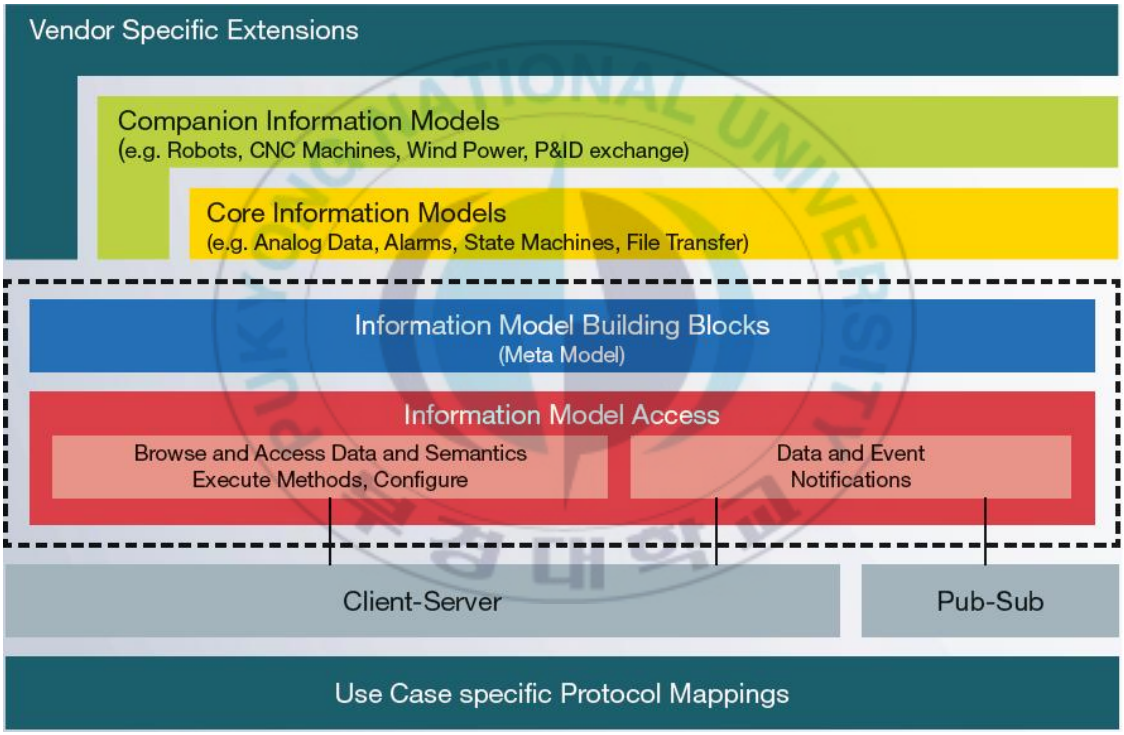
상기한 이슈를 바탕으로 새로운 통합 아키텍쳐에 대한 논의가 활발히 진행되었으며, 2008년에 OPC-UA 표준이 발표되었으며. OPC-UA는 다음과 같은 특징을 확인하였다.

- 기능의 동등성: 기존 OPC Classic에서 가능했던 기능들을 모두 구현할 수 있어야 한다(Discovery, Address, Subscription, Event, Method 등)
- 플랫폼 비의존성: 하드웨어나 OS 종류에 관계 없이 구축이 가능해야 한다.
- 보안성: 인증서 또는 ID와 패스워드로 인증된 사용자만이 통신이 가능할 수 있어야 하며, 통신에 이용되는 패킷은 암호화 되어 스니핑 등의 공격에서 보호될 수 있어야 한다.
- 확장성: 새로운 전송 프로토콜, 보안 알고리즘, 암호화 표준, 어플리케이션 서비스에 대해서 기존 제품에 대한 하위호환성을 유지할 수 있어야

한다.(새로운 OPC-UA제품이나 새 버전은 기존 OPC-UA제품과의 호환성을 유지해야 한다)

- 정보 모델링: 데이터를 정보로 바꾸면서, 복잡하고 다층구조적인 모델로도 확장이 가능하다.

Fig. 1.2는 OPC-UA의 추상정보 모델링 구조를 표현하고 있으며 사용 사례별 프로토콜 매핑이 이루어지며 클라이어트 서버 와 펍-서브는 정보모델에 접근하여 데이터들을 교환하는 구조로 구성된다.



Fig, 1.3 Abstract Information Modeling Structure of OPC-UA

OPC-UA의 표준은 IEC-62541 표준으로 정의되어 있으며, OPC Foundation은 해당 표준을 지켜서 제작된 소프트웨어 또는 시스템을 인증해 주고 있다.

제 2 장 선행 연구

2.1 기존 OPC 소프트웨어를 이용한 프로토콜 변환

Fig. 2.1은 OPC Classic 서버의 대표 제품인 KepwareServerEX Ver 6.9와 미츠비시 FX5U-64M 모델을 통신 연결한 구성으로, MELSEC 프로토콜을 OPC-DA 통신방식으로 변경하여 OPC Quick Client에서 읽기/쓰기가 가능한 것이 확인되었다.

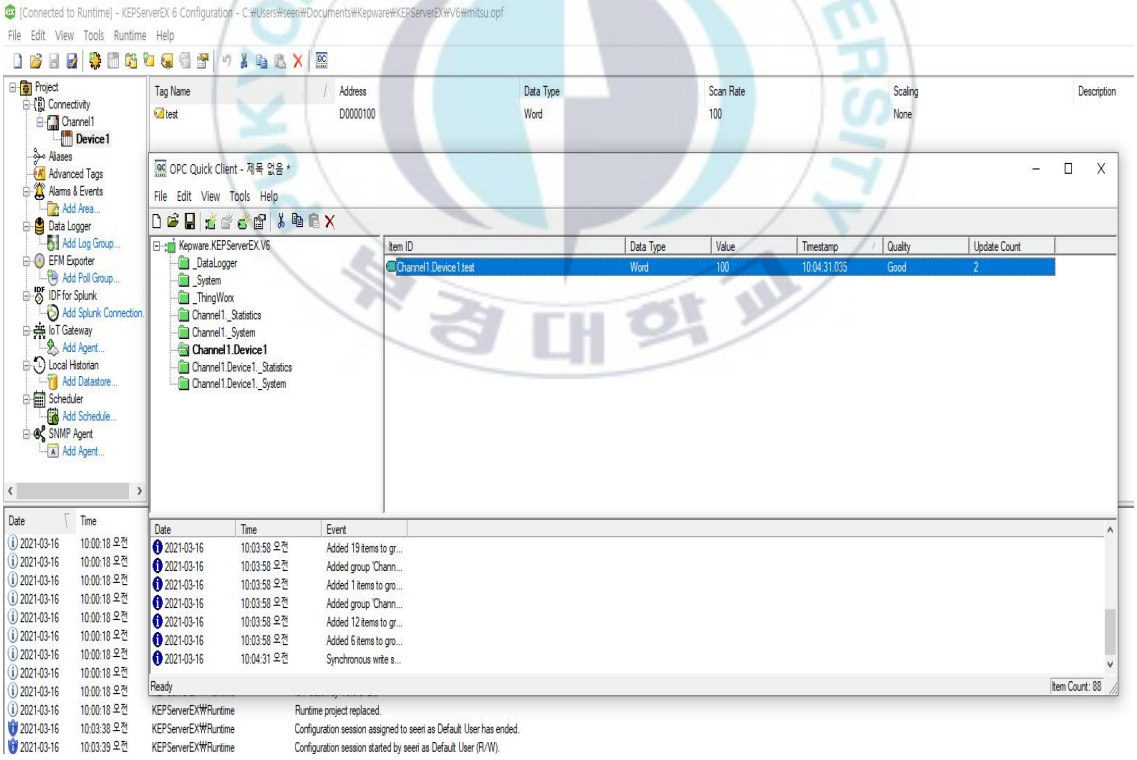


Fig. 2.1 Read/write communication with Mitsubishi FX5U using KepwareServerEx

해당 제품을 포함한 기존 OPC Classic 서버 소프트웨어들은 OPC-UA 통신 역시 지원하기는 하나, 소프트웨어가 윈도우 기반이라 프로토콜 변환에 고가의 데스크탑과 윈도우 OS를 구매해야만 시스템을 운영할 수 있다는 점을 확인할 수 있었다.

2.2 Open62541 OPC-UA 라이브러리 개요

Open62541은 C 기반의 오픈소스 OPC UA 라이브러리 엔진이다. 독일의 프라운호퍼 사를 포함한 여러 단체 및 개인이 개발과 유지보수를 진행하고 있으며, 라이센스는 Mozilla Public License v2.0을 따른다. open62541에는 아래와 같은 장점이 있다.

- 오픈 소스이므로 무료로 사용할 수 있다.
- OPC Foundation의 공식 인증을 받은 소프트웨어로, 안정적이다.
- C 기반으로 컴파일 결과가 가벼워, 비교적 고사양의 하드웨어가 필요하지 않다.
- OS 독립적으로 사용 가능하다.
- 라즈베리파이를 공식적으로 지원한다.

Fig. 2.2는 임베디드 대상을 위한 open6254기반 OPC-UA클라이언트 서버에 대한 아키텍처 다어어그램 구성을 나타내었다.

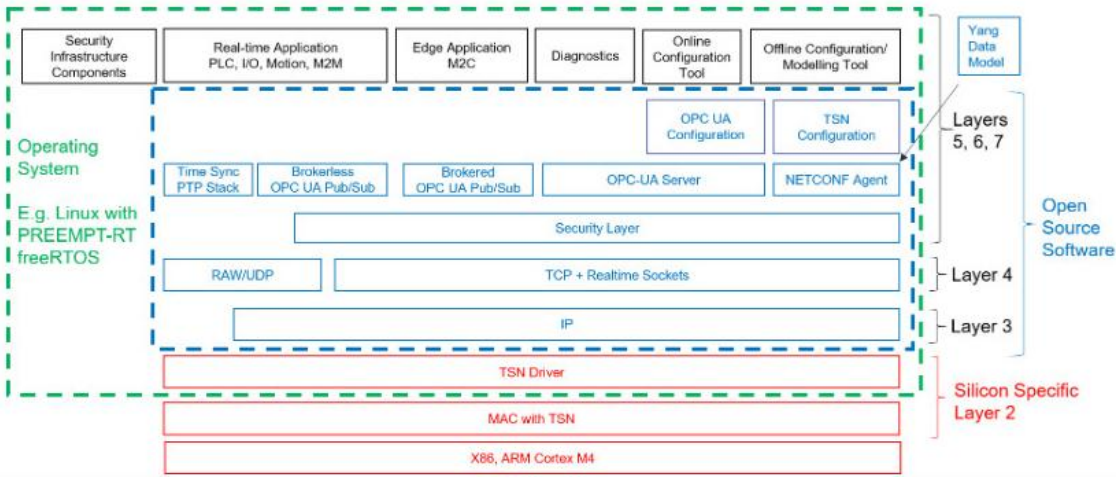


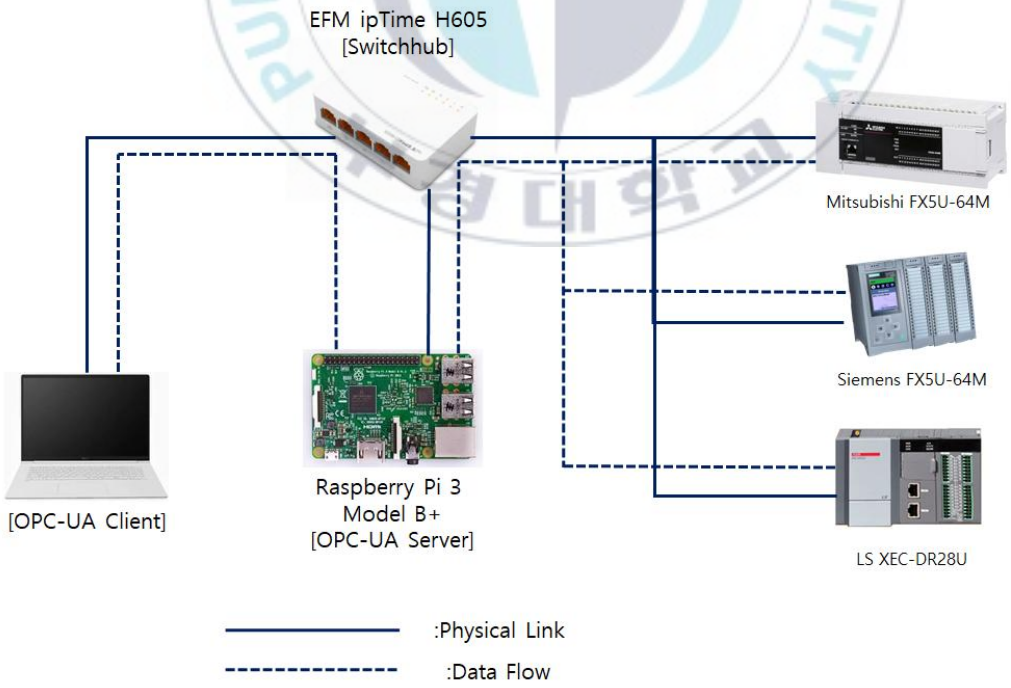
Fig. 2.2 Architecture diagram for open62541-based OPC UA client server SDK for embedded targets

본 논문에서는 OPC-UA가 갖고 있는 플랫폼 비의존성이라는 특성을 살려 임베디드 보드와 리눅스 기반의 OS, 오픈소스 라이브러리인 open62541을 이용해 다수의 산업용 시스템 구성의 PLC와 OPC-UA로 통신하는 과정을 구현하며 성능평가 항목으로 OPC-서버 장비간 통신 속도, 이기종 장비 호환 여부, 명령 전송 정확성, 통신시스템 안정성을 실험하였다.

제 3 장 연구방법 및 절차

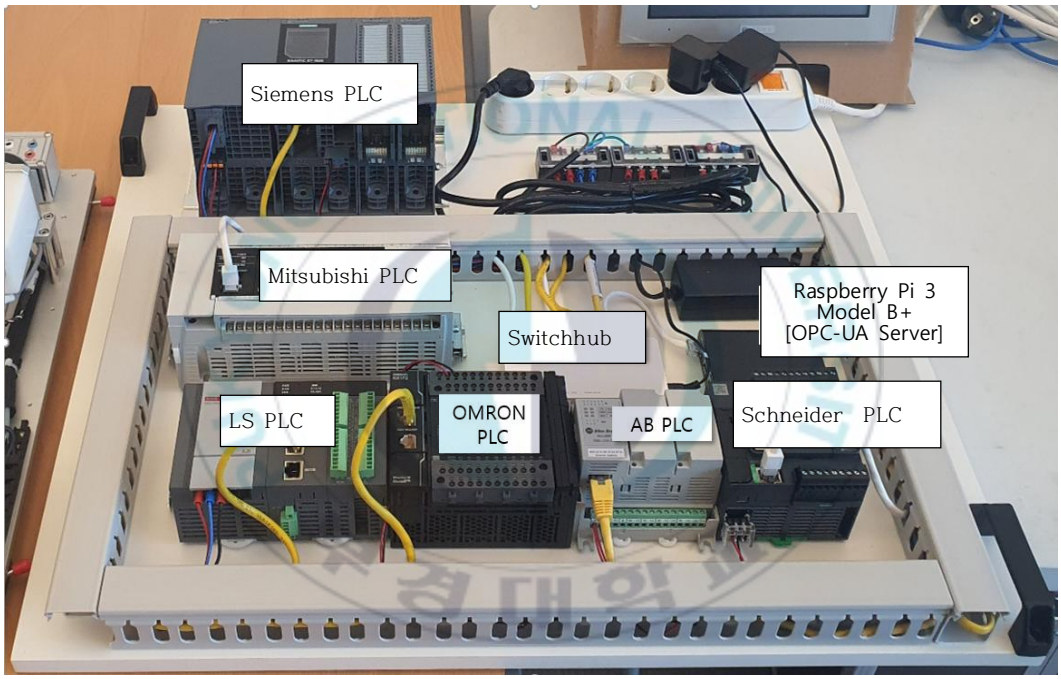
3.1 테스트베드 구성

Fig 3.1에서는 OPC-UA를 테스트 할 테스트 베드의 이론적 구성도이다. 서버가 될 라즈베리 파이에 OPC-UA 서버를 포팅하고, 클라이언트 PC 및 서로 다른 종류의 PLC들과 스위치허브를 통해 이더넷으로 서로 연결한다. 라즈베리 파이에서 구동되는 OPC-UA 서버는 각 PLC별 프로토콜로 통신하여 읽어온 데이터를 OPC-UA 규격으로 변환하며, 클라이언트 PC에서 제 3 사에서 개발한 OPC-UA 클라이언트 앱을 통해 OPC-UA 서버와 통신 및 읽기/쓰기를 테스트하게 된다.



Fig, 3.1 Theoretical configuration of testbeds

라즈베리 파이 3 Model B: OPC-UA 서버는 영국의 라즈베리 파이 재단에서 학교등에서 기초 컴퓨터 과학의 교육을 증진시키기 위해 개발한 신용카드 크기의 싱글보드 컴퓨터이다. 라즈베리파이의 가장 큰 특징은 35$라는 저렴한 가격으로, ARM 프로세서를 사용하여 저전력 대비 준수한 컴퓨팅 성능을 보인다. OS는 데비안 계열의 리눅스 운영체제인 라즈베리 파이 전용 운영체제 Raspberry Pi OS 사용을 권장하며, 해당 OS는 라즈베리 파이 재단 홈페이지에서 무료로 다운로드 받을 수 있다.



Fig, 3.2 Test bed physical configuration

Table 3.1에서 기존 OPC 소프트웨어를 이용한 시스템 구축비용과 본 연구에 구현되는 테스트베드 구축비용의 원가비교를 나태낸다. Table 3.1를 통해 기존 판매되는 제품의 KepwareServerEx 구성비용을 살펴보면 약 4800,000원 비용이 발생되며, 개발하는 Open62541 라이브러리를 활용한 테스트베드

는 약 49,000원 비용이 발생하여 저렴한 비용을 통해 시스템을 구축할 수 있다는 점을 알 수 있다.

Table 3.1 The difference between the cost of building a system using existing OPC software and the cost of building a test bed

	KepwareServerEX		Open62541	
	Product	Price(won)	Product	Price(won)
Hardware	ASUS PN40 J4025	₩327,650	Raspberry Pi 3B	₩49,630
OS	Windows 10 Pro		Raspberry Pi OS	₩0
Software	KepServerEx 6 (Manufacturing Suite)	₩4,476,494 ($3983)	open62541	₩0
Total		₩4,804,144		₩49,630

본 연구에 적용되는 PLC 구성은 Table 3.2 내용과 같다.

Table 3.2 PLC list

No.	Name of PLC	Manufacturer	Model
1	PLC #1	Mitsubishi	FX5U-64M
2	PLC #2	SIEMENS	S7-1500
3	PLC #3	Rockwell	Micro820
4	PLC #4	LS	XEC-DR28U
5	PLC #4	OMRON	NX1P2
6	PLC #5	Schneider	TM221CE16T

3.2 PLC별 통신 프로토콜 분석

3.2.1 MELSEC 프로토콜

MELSEC 프로토콜은 미츠비시에서 생산된 PLC(Q 시리즈, FX시리즈

등)PLC를 외부 장치(PC등)에서 통신하기 위해 사용하는 프로토콜이다. Fig 3.3은 미츠비시 Q시리즈에서 쓰이는 3E Frame의 메시지 송수신 예시를 설명하고 있으며 각각의 세부내용은 확인 할 수 있다.

Request

Subheader	Access Route	Request Data Length	Monitoring Timer	Request Data				
				Command: Batch Read	Subcommand: WORD	Head Device		Number of Device Points: 3
						Device Number: 100	Device Code: Data Register	
50 00	00 FF FF 03 00	0C 00	00 00	01 04	00 00	64 00 00	A8	03 00

Response

Subheader	Access Route	Response Data Length	End code: Normal	Response Data		
				D100	D101	D102
D0 00	00 FF FF 03 00	08 00	00 00	64 00	00 00	00 00

Fig. 3.3 Protocol Data Unit of MELSEC 3E Frame

미츠비시 Q시리즈에서 쓰이는 3E Frame 프로토콜의 구조는 아래와 같다.

- Subheader: Request 메시지는 0x5000, Response 메시지는 0xD000
- Access Route: 요청 메시지를 보낸 단말(PC)로부터 PLC까지의 네트워크 거리를 나타며, PC와 PLC가 직접 통신할 경우 Access Route는 0x00FFFF0300 구성된다.
- Request Data Length: 요청 메시지의 길이를 나타낸다.

Request Data+Monitoring Timer의 길이를 바이트 단위로 나타낸 값이다.

- Monitoring Timer: 읽기/쓰기 프로세스가 완료되어 대답이 돌아오기 까지 기다리는 시간을 나타낸다.
- Request Data
- Command: 읽기/쓰기 등의 명령문 코드이다.

0104: Batch Read

-Subcommand: 명령을 보조하는 코드이며 코드의 내용은 명령문에 따라 달라진다.

-Head Device: 읽기/쓰기 명령문에서 접근할 데이터 주소명을 나타낸다.

- End Code: 요청 메시지가 정상일 경우 0, 그렇지 않을 경우 에러 코드를 반환한다.

- Response Data: 응답메시지이며 메시지의 형태는 요청 메시지에 따라 달라진다.

3.2.2 Siemens S7 프로토콜

Siemens사의 SIMATIC S7 시리즈는 보통 PROFIBUS/PROFINET이라는 전용 프로토콜을 쓰지만, 이더넷 통신을 해야 할 때는 S7 프로토콜을 사용한다. S7 프로토콜은 ISO on TCP(RFC1006)에 의존하기 때문에 직접 구현하기 어렵다. 따라서 본 논문에서는 Fig 3.4구성의 오픈소스 라이브러리인 snap7을 이용하여 통신을 구현하였다.

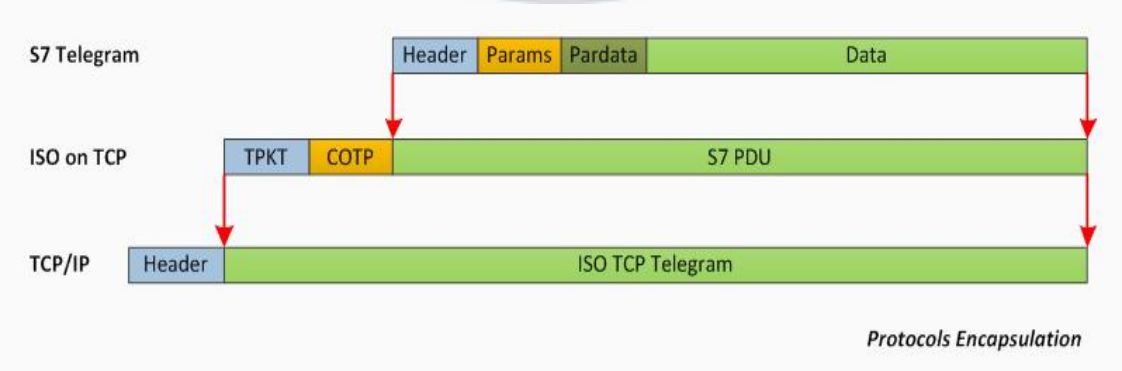


Fig. 3.4 Encapsulation of Siemens S7 Protocol

3.2.3 LS XGT-FeNet 프로토콜

Fig 3.5은 LS 일렉트릭(구 LS산전)의 전용 프로토콜로 , XGT, XGB, XEC등의 LS사 전용 PLC와 데이터 통신을 위해 구축한 프로토콜이다. Fig 3.5은 아래는 연속읽기/쓰기 메시지의 구성을 나타내는 예시이다.

Request

Application Header								Application Instruction		
Company ID ("LSIS-XGT")	PLC Info	CPU Info	Source of Frame	Invoke ID	Length of application instruction	Fenet Position	BCC	Command: Read Request	Data Type: Block Type	Reserved
4C 53 49 53 2D 58 47 54 00 00	00 00	A0	33	00 00	14 00	00	00	54 00	14 00	00 00

Application Instruction			
Block Number	Block		Data Length
	Variable Length	Variable: "%DW100"	
01 00	06 00	25 44 57 31 30 30	06 00

Response

Application Header								Application Instruction		
Company ID ("LSIS-XGT")	PLC Info	CPU Info	Source of Frame	Invoke ID	Length of application instruction	Fenet Position	BCC	Command: Read Response	Data Type: Block Type	Reserved
4C 53 49 53 2D 58 47 54 00 00	06 02	B4	11	00 00	12 00	01	00	55 00	14 00	00 00

Application Instruction			
Error State	Block Number	Block	
		Data Length	Data
00 00	01 00	06 00	64 00 00 00 00 00

Fig, 3.5 Protocol Data Unit of FeNet

LS 일렉트릭(구 LS산전)의 전용 프로토콜의 구조는 아래와 같다.

- Application Header: 메시지의 헤더를 나타낸다.
- Company ID: 10byte의 고정메시지로 구성된다.(“LSIS-XGT”)
- PLC Info: PLC의 상태 정보로 요청메시지에는 표시하지 않는다.
- CPU Info: 메시지를 전송하는 PLC의 기종을 의미한다.
- Source of Frame: 프레임의 방향을 나타낸다.

0x33 - 요청메시지, 0x11 - 응답메시지

- Invoke ID: 프레임의 순서를 나타내는 번호이며 설정해주지 않아도 무방하다.
- Length: Application Instruction의 길이(바이트)를 의미한다.
- Fenet Position: PLC의 슬롯, 랙 정보. PC는 0으로 표시한다.
- BCC: Application Header의 체크섬을 의미한다.
- Command: 명령어

Table 3.2는 데이터 읽기 쓰기에 필요한 명령어 구성을 나타내고 있다.

Table 3.3 Read Write Command Description

	read	writing
request	0x5400	0x5800
answer	0x5500	0x5900

- 데이터 타입: 읽기/쓰기에 쓰이는 변수의 길이 단위를 의미며, 연속 읽기/쓰기의 경우에는 단위가 Byte로 고정되며, 길이는 데이터 주소 뒤에 별도로 표기한다.
- Block Number: 읽기/쓰기를 진행할 변수 블록의 개수이며 연속 읽기/쓰기일때는 1로 고정으로 설정한다.
- Variable Length: 블록에 속한 변수 문자열의 길이를 의미한다.
- Variable: 읽기/쓰기를 진행할 변수의 주소를 ASCII 문자열로 표시하며, 첫 번째 문자: “%”, 변수 문자열임을 표시, 두 번째 문자: 메모리 영역이다. 메모리 영역에 대해서는 PLC별로 구성이 다른게 설정된다. 세 번째 문자: 메모리의 주소 단위를 설명하고 나머지 문자: 메모리의 주소 오프셋. 오프셋의 크기는 주소 단위에 따라 달라진다.(“%DW100” = “%DB200”)
- Error State: 요청 메시지가 정상일 경우 0, 그렇지 않을 경우 에러 코드를 반환한다.

- Data Length: 읽기 요청의 경우 읽어온 블록의 데이터의 길이를 의미한다.
- Data: 읽기 요청으로 읽어온 블록의 데이터를 의미한다.

3.3 OPC-UA 서버 프로그램의 구성

본 연구에서 사용되는 Raspberry Pi 임베디드는 비동기식 통신 구축을 위해 서버 프로세스와 모니터링 프로세스를 나누어 설계되어 동작하게끔 코딩하였다. Fig 3.6 구성으로 OPC-UA서버는 open62541 라이브러리를 사용하여 구성하였으며, 외부 제조사별 PLC에 대해 데이터 쓰기를 실행하며, PLC내부 데이터 리스트에 대한 읽기를 실행하는 방식으로 개발 OPC-UA서버에 전달된다. 전달된 데이터는 OPC-UA 서버내 Object 구성에 의해 데이터는 교환 및 전달되는 방식으로 운영되는 구성으로 설계하였다.

- 서버가 생성되면, 연결할 PLC의 종류 만큼 Object Node, dat 파일, 모니터 프로세스를 생성한다.
- PLC마다 할당된 변수만큼 Variable Node를 생성한다.
- OPC-UA 서버의 역할: 읽기 요청이 들어오면 dat파일을 읽어 데이터를 변수에 할당한다. 쓰기 요청이 들어오면 변수의 정보와 쓰기 값을 req파일에 저장한다.

-모니터 프로세스의 역할 : req파일이 있는 지 확인하고 있으면 PLC에 쓰기 명령을 전송한다.

req파일이 없으면 읽기 명령을 전송하고 수신받은 데이터를 dat 파일에 저장코딩한 파일을 라즈베리파이에 넣고 gcc로 컴파일한다.

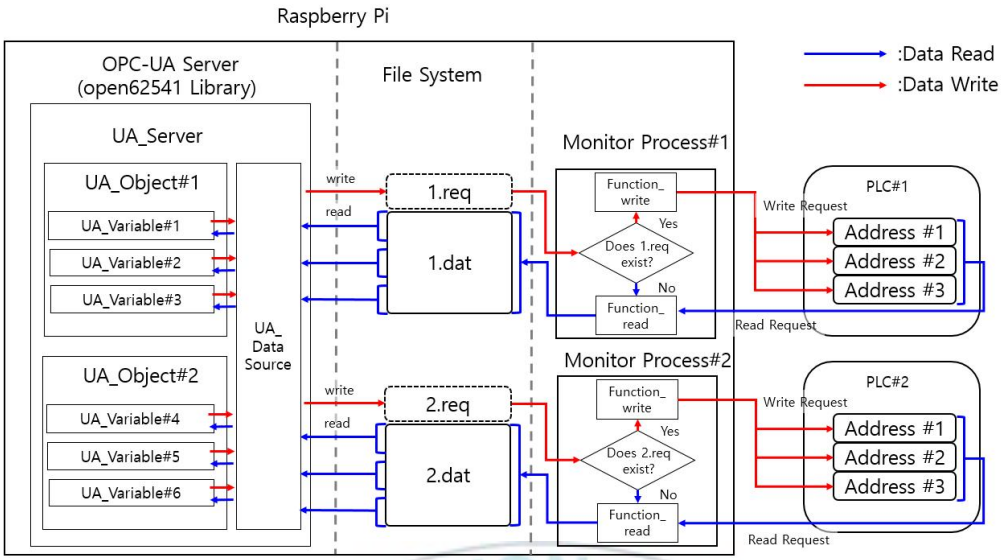


Fig. 3.6 Structure diagram of data exchange with PLC using open62541 library

Fig 3.7은 개발 OPC_UA에 대한 동작 순서를 나타내고 있다 GL_OPC를 실행하면 실행파일과 같은 디렉토리에 있는 첫 번째 GLUAServer.conf파일을 읽기를 실행한다.(파일이 없거나 유효하지 않으면 종료) 두 번째 Server 객체를 생성한다. (UA_Server_new(), UA_ServerConfig_setMinimal() 세 번째 GLUAServer.conf파일에 설정된 대로 PLC 또는 센서 대수에 맞춰 ObjectNode를 생성하다. 네 번째 GLUAServer.conf파일에 설정된대로 데이터 변수들을 생성, ObjectNode의 개수만큼 GLUAMonitor 프로세스를 실행한다. 다섯 번째 OPC 서버 프로세스를 시작하며 서버 프로세스 종료전 GLUAMonitor 프로세스들을 먼저 종료시튼 순서로 구동되어 진다.

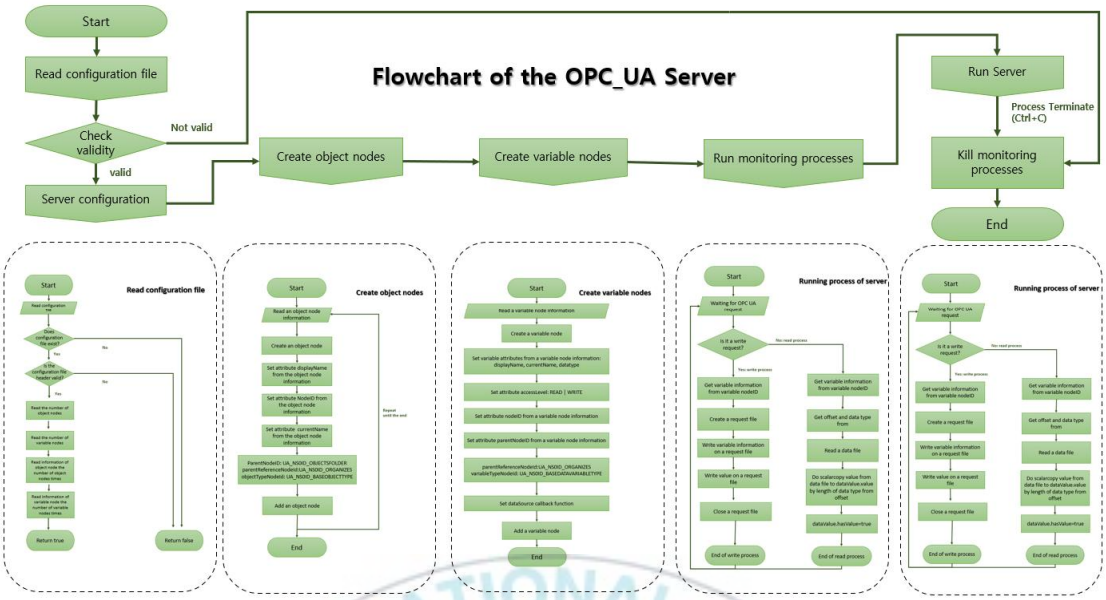


Fig. 3.7 Algorithm of OPC-UA server and monitoring process

3.4 테스트 진행

(1) 통신 시간 테스트: OPC-UA 서버와 각 PLC간의 요청/응답 패킷을 캡쳐하여, 서버로부터의 요청이 각 PLC에 얼마나 빠르게 전달되는지 확인한다.
(2) 장비 호환성 테스트: 각 PLC 내부 어드레스의 값을 변경하였을 때, OPC-UA 서버의 변수 노드의 값이 변하는 지 확인한다.
(3) 전송 정확성 테스트: OPC-UA 서버의 변수 노드의 값을 변경하였을 때, 각 PLC의 주소 값이 변하는 지 확인한다.
(4) 안정성 테스트: OPC-UA 서버와 PLC간 네트워크 연결을 30초 이상 물리적으로 분리 시켰다가 다시 연결시켰을 때, 서버와 PLC간의 연결이 회복 되는지 확인한다.

Table 3.4은 OPC-UA개발에 적용된 6종류의 PLC 구성(Mitsubishi_FX5U, SIEMENS_S7-1500,OMRON_NX1P2,Schneider_TM221CE16T, Rockwell_Micro820, LS_xec-DR28U)과 PLC 제조사별 사용 되는 프로그램에 대한 설명을 나타내고 있다.

Table 3.4 Program list

No.	Name of PLC	Node id	Model	Name of program
1	PLC #1	ns=1, i=50000	Mitsubishi (FX5U-64M)	GX-WORKS3
2	PLC #2	ns=1, i=50001	SIEMENS (S7-1500)	Totally Integrated Automation Portal
3	PLC #3	ns=1, i=50002	OMRON(NX1P2)	Sysmac Studio
4	PLC #4	ns=1, i=50003	Schneider (TM221CE16T)	Ecostruxure Machine Expert - Basic
5	PLC #5	ns=1, i=50004	Rockwell (Micro820)	Connected Component Workbench
6	PLC #6	ns=1, i=50005	LS(XEC-DR28U)	XG5000

Table 3.5은 개발 OPC_UA 서버와 각각의 제조사별 PLC 6종에 대한 장비간의 응답 시간을 측정한 내용이다. 시험절차 구성은 1) OPC-UA서버, PLC(6종)을 구동 2)클라이언드 PC에서 PuTTY를 사용하여 OPC-UA 서버에 접속 3) OPC-UA 서버에 Tshark를 사용하여 OPC-UA서버와 PLC #1간의 패킷을 3s 이상 캡처 4)Wireshark를 사용하여 절차 3)에서 생성한 패킷 파일을 분석 후 수식 “ 응답 시간(ms) = B - A,”을 사용하여 응답 시간을 계산함 ※ A : OPC서버에서 PLC로 송신한 패킷 Time, B : PLC로부터 수신한 패킷의 Time를 나타냄, 5)절차 3) ~ 4)를 총 5회 반복한 후 응답 시간의 평균을 구함 6) 절차 3)의 PLC를 PLC#2 ~ #6으로 변경하면서 절차 3) ~ 5)를 총 6회 반복 후 평균 응답시간을 구함

Table 3.5 Transmission rate between OPC server and equipment

Test purpose	Measuring response time between OPC-UA server and PLC equipment
Test procedure	1) Drive OPC-UA server and PLC(6 EA). 2) Connect client PC to OPC-UA via PuTTy. 3) Capture the packet more than 3s between OPC-UA and PLC #1 via TShark in the OPC-UA server. 4) Calculate the response time by using formula below after analyzing the packet file created in step 3) via Wireshark. ※response time(ms) = B - A A : Time of packet sent from OPC server to PLC B : Time of packet received from PLC 5) Calculate the average of response time after repeating 5 times from step 3) to 4). 6) Calculate the average of response time of PLC(6 EA) after repeating 6 times from step 3) ~ 5) while changing PLC of step 3) to PLC #2 ~ #6.
Test standard	within 50ms
Test result	average 15ms

Table 3.6은 개발 OPC-UA 서버 장비간 통시 속도 시험 결과이다.

Table 3.6 Test result of transmission rate between OPC server and device

	PLC #1	PLC #2	PLC #3	PLC #4	PLC #5	PLC #6	average
response time(ms)	5	7	5	5	23	47	15

Table 3.7 Test result of response time between OPC-UA server and PLC #1

	One time	Two times	Three times	Four times	Five times	average
response time(ms)	5	4	5	5	4	5

Table 3.8 Test result of response time between OPC-UA server and PLC #2

	One time	Two times	Three times	Four times	Five times	average
response time(ms)	7	6	8	7	6	7

Table 3.9 Test result of response time between OPC-UA server and PLC #3

	One time	Two times	Three times	Four times	Five times	average
response time(ms)	24	24	20	24	24	23

Table 3.10 Test result of response time between OPC-UA server and PLC #4

	One time	Two times	Three times	Four times	Five times	average
response time(ms)	29	53	51	51	51	47

Table 3.11은 개발 OPC_UA 서버와 이기종 장비 호환 여부를 테스트 하는 것으로 PLC에서 변경한 데이터가 OPC-UA 서버에 반영되는지 확인하게 된다. 시험절차 구성은 1) OPC-UA서버, PLC(6종) 구동 2) 클라이언트 PC에서 Integration Objects OPC-UA Client를 사용하여 Test.ouc 파일을 실행 3) 시험용 PC에서 GX-WORKS3 프로그램을 사용하여 PLC #1의 값을 0~100 사이의 임의의 값으로 변경함 4) PLC별 Node ID 및 프로그램 리스트 표를 참고하여 Integration Objects OPC UA Clien에서 PLC #1에 해당하는 Node ID의 Value 값이 변경되는지 확인함 5) 절차 3) ~ 4)를 총 5회 반복 후 수식 “성공률(%) (A / B) * 100”을 사용하여 성공률을 계산함 ※A : 정상동작한 횟수, B : 절차 3)에서 PLC의 값 변경 시도한 횟수를 나타냄, 6) PLC별 Node ID 및 프로그램 리스트 표를 참고하여 3) ~ 5)를 총 6회 반복 후 평균 성공률을 구한다.

Table 3.11 Compatibility of heterogeneous equipment

<table>
<tr><td>Test purpose</td><td>Checking whether the modified data on PLC applies to OPC-UA server.</td></tr>
<tr><td>Test procedure</td><td>1) Drive OPC-UA server and PLC(6 EA).
2) Execute test.ouc file using Integration Objects OPC UA Client in client PLC.
3) Change the value of PLC #1 to arbitrary value from 0 to 100 via GX-WORKS3 program on a test PC.
4) Check whether the Value of Node id corresponding to PLC #1 changes on the Integration Objects OPC UA Client. Please refer to table “Node id and program list.
5) Calculate the success rate by using formula below after repeating 5 times from step 3) to 4).
※ Success rate(%) = (A / B) * 100
A : The number of normal operation.
B : The number of attempts to change the value of PLC on step 3)
※ Normal operation : In case of matching modified Value on step 4) with changed arbitrary value on step 3).
6) Calculate the average of success rate after repeating 6 times from step 3) to 5) by modifying PLC and program. Please refer to table “Node id and program list .

<table>
<tr><th>No.</th><th>Name of PLC</th><th>Node id</th><th>Name of program</th></tr>
<tr><td>1</td><td>PLC #1</td><td>ns=1, i=50000</td><td>GX-WORKS3</td></tr>
<tr><td>2</td><td>PLC #2</td><td>ns=1, i=50001</td><td>Totally Integrated Automation Portal</td></tr>
<tr><td>3</td><td>PLC #3</td><td>ns=1, i=50002</td><td>Sysmac Studio</td></tr>
<tr><td>4</td><td>PLC #4</td><td>ns=1, i=50003</td><td>Ecostruxure Machine Expert - Basic</td></tr>
<tr><td>5</td><td>PLC #5</td><td>ns=1, i=50004</td><td>Connected Component Workbench</td></tr>
<tr><td>6</td><td>PLC #6</td><td>ns=1, i=50005</td><td>XG5000</td></tr>
</table>
[Node id and program list]</td></tr>
<tr><td>Test standard</td><td>100%</td></tr>
<tr><td>Test result</td><td>average 100%</td></tr>
</table>

Table 3.12은 개발 OPC-UA 이기종 장비 호환 여부 시험결과이다.

Table 3.12 Test result of compatibility of heterogeneous equipment

	PLC #1	PLC #2	PLC #3	PLC #4	PLC #5	PLC #6	average
The number of normal operation	5	5	5	5	5	5	-
The number of attempts to change the value of PLC	5	5	5	5	5	5	-
Success rate(%)	100	100	100	100	100	100	100

Table 3.13은 개발 OPC_UA 서버에서 변경한 데이터가 PLC에 반영되는지 확인하기 위한 시험이다. 시험절차 구성은 1) Table 3.9 이기종 장비 호환 여부 시험에서 이어서 진행함 2) 클라이언트 PC의 Integration Objects OPC UA Client에서 Table 3.9 이기종 장비 호환 여부 시험에 제공한 PLC별 Node ID의 Value 값을 0 ~ 100 사이의 임의의 값으로 변경함. 3) 시험용 PC에서 GX-WORKS3 프로그램을 사용하여 PLC #1의 값이 변경되었는지 확인함. 4) 절차 2) ~ 3)을 총 5회 반복 후 수식 “정확성(%) = (A / B) * 100”을 사용하여 정확성을 계산함 ※ A : 정상 동작한 횟수 , B : 절차 2)에서 변경한 임의의 값이 절차 3)에서 변경된 값과 일치하는 경우를 나타냄
5) Table3.9 이기종 장비 호환 여부 시험에서 제공한 PLC별 Node ID 및 프로그램 리스트 표를 참고하여 PLC 및 프로그램을 변경하면서 절차 2) ~ 4)를 총 6회 반복 후 평균 정확성을 구한다.

Table 3.13 Accuracy of Command Transmission

Test purpose	Checking whether the modified data on OPC-UA server applies to PLC.
Test procedure	1) Proceed after “Table 3.9 Compatibility of heterogeneous equipment” test. 2) Change the value of Node id corresponding to PLC #1 to arbitrary value from 0 to 100. Please refer to “Table 3.2” of “5.2 Compatibility of heterogeneous equipment” test in the Integration Objects OPC UA Client of client PC. 3) Check whether the value of PLC #1 changes via GX-WORKS3 program on a test PC. 4) Calculate the accuracy by using formula below after repeating 5 times from step 2) to 3). ※ Accuracy(%) = (A / B) * 100 A : The number of normal operation. B : The number of attempts to change the value of PLC on step 2) ※ Normal operation : In case of matching modified Value on step 3) with changed arbitrary value on step 2). 5) Calculate the average of accuracy after repeating 6 times from step 2) to 4) by modifying PLC and program. Please refer to table “Table 3.2” of “5.2 Compatibility of heterogeneous equipment”.
Test standard	100%
Test result	average 100%

Table 3.14은 개발 OPC-UA 명령 전송 정확성 시험결과이다.

Table 3.14 Test result of accuracy of command transmission

	PLC #1	PLC #2	PLC #3	PLC #4	PLC #5	PLC #6	average
The number of normal operation	5	5	5	5	5	5	-
The number of attempts to change the value of PLC	5	5	5	5	5	5	-
Success rate(%)	100	100	100	100	100	100	100

Table 3.15은 개발 OPC_UA 서버와 PLC간의 통신이 끊어진 후 다시 연결되었을 때 OPC UA서버가 작동하는지 확인하기 위한 시험이다. 시험절차 구성은 1) Table 3.13 명령 전송 정확성 시험에 이어서 진행함 2) OPC UA 서버와 PLC #6의 통신을 30s 이상 단절시킨 후 다시 연결 3) Integration Objects OPC UA Client를 사용하여 PLC #6에 해당하는 Node ID의 Value 값을 0 ~ 100사이의 임의의 값으로 변경함 4) 시험용 PC에서 XG5000 프로그램을 사용하여 PLC #6의 값이 변경되었는지 확인함 5) 절차 2) ~ 4)를 총 5회 반복후 수식 “안전성(%) = (A / B) * 100”을 사용하여 안정성을 계산함※ A = 정상 동작한 횟수 , B = 통신 단절을 시도한 횟수를 나타냄

Table 3.15 Safety of communications system

Test purpose	Checking whether OPC-UA works when the communication between OPC-UA server and PLC disconnects and reconnects.
Test procedure	1) Proceed after “Table 3.13 Accuracy of Command Transmission” test. 2) Reconnect the communication between OPC-UA server and PLC #6 after disconnecting them more than 30s. ※ Method for communication interruption : Remove UTP cable connected to PLC #6. 3) Change the value of Node id corresponding to PLC #6 to arbitrary value from 0 to 100 via Integration Objects OPC UA Client. 4) Check whether the value of PLC #6 changes via XG5000 program on a test PC. 5) Calculate the safety by using formula below after repeating 5 times from step 2) to 4). ※ Safety(%) = (A / B) * 100 A : The number of normal operation. B : The number of attempts the communication interruption. ※ Normal operation : In case of matching modified Value on step 4) with changed arbitrary value on step 3).
Test standard	100%
Test result	100% [100 = (5 / 5) * 100]

Table 3.16 개발 OPC-UA 통신시스템 안정성 시험결과이다.

Table 3.16 Test results for communication system stability

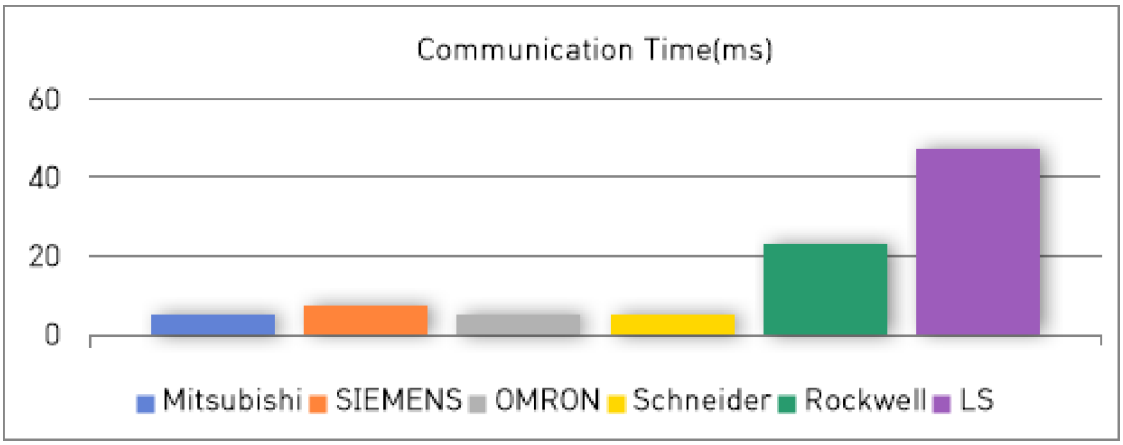
	PLC #1	PLC #2	PLC #3	PLC #4	PLC #5	PLC #6	average
The number of normal operation	5	5	5	5	5	5	-
The Number of attempts to disconnect communication	5	5	5	5	5	5	-
Success rate(%)	100	100	100	100	100	100	100

제 4 장 테스트 결과 및 고찰

본 연구는 임베디드 보드와 오픈 소스 소프트웨어를 이용한 산업용 장비의 OPC-UA 구현 연구로써 “OPC-UA가 갖고 있는 플랫폼 비의존성이라는 특성을 살려 임베디드 보드와 리눅스 기반의 OS, 오픈소스 라이브러리인 open62541을 이용해 다수의 PLC와 OPC-UA로 통신하는 과정을 구현하였다. 테스트 진행을 위해 산업현장에서 사용중인 6개 제조사의 PLC를 이용하였으며 성능 테스트 진행 항목으로 첫 번째 OPC 서버 장비간 통신속도, 두 번째 이기종 장비 호환여부 확인, 세 번째 명령 전송 정확성 확인, 마지막으로 통신시스템 안전성을 확인하였다.

Table 4.1에서 OPC 서버 장비간 통신속도에 대한 결과를 나타내고 있으며 통신 속도에 차이가 발생된 부분은 제조사별 PLC통신 속도 설정 및 CPU 환경설정 범위 및 임베디드보드의 사양에 따른 차이에서 발생되는 요소이며 OPC-UA 통신 시험기준으로 한 설정한 50ms범위 이내에 측정 되었음을 알수 있다.

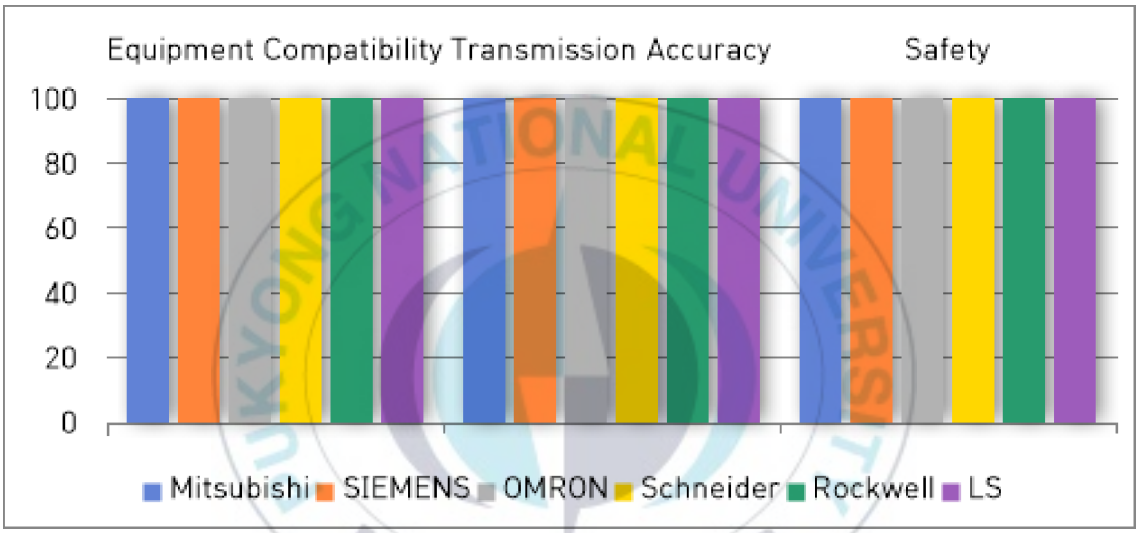
Table 4.1 Communication Time Test Results



	Mitsubishi	SIEMENS	OMRON	Schneider	Rockwell	LS
Communication Time	5ms	7ms	5ms	5ms	23ms	47ms

Table 4.2에서 측정한 이기종 장비 호환여부, 명령 전송 정확성, 통신시스템 안전성에 대한 항목은 모두 만족한 결과를 확인할 수 있었다.

Table 4.2 Equipment Compatibility,Transmission Accuracy,Safety Test Results



	Mitsubishi	SIEMENS	OMRON	Schneider	Rockwell	LS
Equipment Compatibility	100%	100%	100%	100%	100%	100%
Transmission Accuracy	100%	100%	100%	100%	100%	100%
Safety	100%	100%	100%	100%	100%	100%

제 5 장 결 론

본 논문에서는 산업현장 표준으로 사용 되어지는 OPC-UA 기준 오픈소스 소프트웨어를 이용하여 임베디드 보드에 탑제하여 MELSEC, SIEMENS, LS ELECTRIN PLC 각각의 기종에 대한 통신 구현을 실험하였다.

실험을 통해 구현된 프레임 워크가 정상 작동함을 확인하고, 이에 대한 성능을 측정하였다. 성능 평가 구성으로 OPC 서버-장비간 통신속도, 이기종 장비 호환 여부, 명령 전송 정확성, 통신시스템 안정성에 대해 확인할 수 있었다. 또한 각 PLC의 프로토콜과 OPC-UA서버와의 통신을 윈도우와 데스크탑 기반이 아닌 라즈베리 파이와 리눅스 기반으로도 가능하다는 것을 확인 할 수 있었다. 이로써 기존의 설비들을 큰 비용을 들이지 않고도 OPC-UA로 변환 가능하다는 것을 확인할 수 있었다.

향후 연구로는 임베디드 보드 기반 OPC-UA통신을 이용하여 산업현장에서 구축한 스마트팩토리 시스템 구성에 적용 설비데이터를 바탕으로 하는 빅데이터 수집 플랫폼을 개발하고자 한다.

참고 문헌

[1] OPC Unified Architecture Specification - Part 1: Overview and Concepts, OPC Foundation, 2015.

[2] OPC Unified Architecture - Part 1: Overview and Concepts, IEC 62541-1, Oct. 2016.

[3]J. Imtiaz and J. Jasperneite, "Scalability of OPCUA down to the chip level enables, "internet of things"," Proc. of the 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 500-505, Jul. 2013.

[4]J. Slay, M. Miller, Lessons learned from the Maroochy water breach, Critical Infrastructure Protection,, vol. 253/2007, Springer, Boston, 2007, pp. 73-82