ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 21-02-09
58

j.inst.Korean.electr.electron.eng.Vol.25,No.2,291~301,june 2021
https://doi.org/10.7471/ikeee.2021.25.2.291

# OPC UA Publisher와 DDS Subscriber의 상호운용성을 위한 게이트웨이 플랫폼

# Gateway platform for interoperability between OPC UA Publisher and DDS Subscribers

심 웅 빈*, 송 병 권**★, 신 준 호***

Woong-Bin Sim*, Byung-Kwen Song**★, Jun-Ho Shin***

## Abstract

OPC UA at the control and field level does not provide enough performance to replace the field bus. The OPC Foundation aims for a real-time and connection-less mechanism, and has added the OPC UA publish-subscribe model, a new specification that supports broker functions such as MQTT and AMQP, as the OPC UA Part 14 standard. This paper is about a gateway for interoperability between OPC UA publisher with the addition of OPC UA Part14 standard and DDS subscribers. Raspberry Pi 4 is used for the gateway proposed in this paper, and OpenDDS, an open source, is used for DDS. OPC UA publish-subscribe module used A-Open62541 publish-subscribe module, which additionally implements functions not provided by the corresponding source based on Open62541 publish-subscribe open source.

## 요 약

제어 및 필드 레벨에서의 OPC UA는 필드 버스를 대체할 만큼의 충분한 성능을 제공하지 못한다. OPC 협회는 실시간 및 비연결 메커니즘을 지향하고, MQTT나 AMQP와 같이 브로커 기능을 지원하는 새로운 사양인 OPC UA 발행-구독 모델을 OPC UA Part14 표준으로 추가하였다. 본 논문은 OPC UA Part14 표준을 추가한 OPC UA 발행자와 DDS 구독자 간에 상호운용성을 위한 게이트웨이에 관한 내용이다. 본 논문에서 제안한 게이트웨이는 라즈베리파이 4를 사용하였으며, DDS는 오픈소스인 OpenDDS를 사용하였다. OPC UA 발행-구독 모듈은 Open62541 발행-구독 오픈 소스를 기반으로 해당 소스에서 제공하지 않는 기능을 추가 구현한 것인 A-Open62541 발행-구독 모듈을 사용하였다.

*Key words : OPC UA Publish-Subscribe, OPC UAPart 14, Open62541, DDS, Gateway*

* Dept. of Electronics and Computer Engineering, Seokyeong University
** Dept. of Electronics Engineering, Seokyeong University
*** Smart Manufacturing Research Center, Korea Electronics Technology Institute
★ Corresponding author
E-mail : bksong@skuniv.ac.kr, Tel : +82-2-940-7739

※ Acknowledgment
This study was funded by the Ministry of Environment and supported by the Korea Institute of Environmental Industry and Technology's Safety Management Technology Development Project (2020002970006).
Manuscript received May. 13, 2021; revised May. 31, 2021; accepted Jun. 8, 2021.

## Ⅰ. 서론

오늘날 산업 시스템에서는 시스템 상호 간에 데이터 교환의 중요성이 점점 증가하고 있다. 최근 산업 자동화 및 제어 시스템은 산업용 사물인터넷(IIoT)이 적용된 지능형 기기가 포함되어있어, MQTT (Message Queuing Telemetry Transport), AMQP (Advanced Message Queuing Protocol), HTTP 및 DDS(Data Distribution Service) 등 데이터 교환을 위하여 다양한 미들웨어를 사용하고 있다. 이러한 미들웨어는 정보 모델이 서로 상이하고 통신 프로토콜이 다르기 때문에 직접 통신하기 어렵다. 예를 들어, OPC UA는 클라이언트-서버(요청-응답) 구조인 반면에, MQTT, AMQP 및 DDS는 발행-구독(Publish-Subscribe) 모델을 사용한다. MQTT나 AMQP을 사용하는 장치는 브로커를 사용하여 토픽 데이터를 연결하고 발행한다. 데이터 소비자 즉 구독자(subscriber)는 브로커에 연결하여 원하는 특정 토픽 데이터를 구독한다. 그러나 DDS는 브로커를 사용하지 않는 방식이다.

OPC UA는 자동화 피라미드(pyramid)의 상위 계층에서 사용되는 사실상 표준 통신 미들웨어 기술이고, Industry 4.0과 관련하여 가장 유망한 산업 자동화 통신 프로토콜 중 하나이다[2]. 이러한 OPC UA는 SOA(Service-Oriented Architecture) 기반의 연결 지향형(connection-oriented) 프로토콜을 사용하여 제어 시스템과 엔터프라이즈 레벨의 자동화 기기 간에 데이터 교환용으로 사용이 되고 있다[2].

현재 필드 레벨에서는 성능을 달성하기 위하여 이더넷(ethernet) 기반 필드 버스(field bus) 프로토콜(예, SERCOS, ProfiNET, CAN 등)이 사용된다. 그러나 이러한 프로토콜들은 동작 특성이 서로 다르기 때문에 상호운용성(Interoperability)에 문제가 발생한다. 만약 필드 레벨에 여러 종류의 프로토콜이 사용된다면 서로 다른 프로토콜 특성 때문에 프로토콜별로 MMI(Man Machine Interface) 등이 요구되고, 또한 전체 시스템 통합 시 프로토콜 상호 변환 기능을 제공하는 여러 종류의 게이트웨이가 필요하게 된다. 이러한 문제를 해결하려면 자동화 레벨 전체를 동일한 프로토콜로 통합하는 것이 필요하다. 이러한 요구사항을 만족하기 위해서, OPC 협회는 실시간 및 비연결 메커니즘을 지향하고, MQTT나 AMQP와 같이 브로커 기능을 지원하는 새로운 사양인 OPC UA Publish-Subscribe(이하 PubSub) 모델을 OPC UA Part14 표준으로 추가하였다[4].

본 논문은 OPC UA Part14 표준으로 추가한 A-Open62541 PubSub 미들웨어를 사용하는 발행자(publisher)와 DDS 구독자 간의 상호운용성(Interoperability)을 지원하기 위한 게이트웨이 설계 및 구현에 관한 것이다. 제안된 게이트웨이는 MQTT나 AMQP 브로커 내에 존재하여 브로커가 수신한 OPC UA 발행자 데이터를 DDS 메시지로 변경한 후 DDS 구독자로 전송한다.

본 논문에서 사용된 OPC UA PubSub 모듈은 Open62541 PubSub[5] 오픈소스를 기반으로, 보안키 서비스를 제외하고 OPC UA Part14 사양을 구현한 A-Open62541 PubSub을 사용하였다. A-Open62541 PubSub에서 지원하는 MQTT는 Open62541 PubSub에 적용된 MQTT-C[15]를 사용하지 않고, 일반적으로 산업 표준처럼 사용되는 Mosquitto[6]를 이용하였다. Open62541 PubSub은 AMQP를 지원하지 않지만, A-Open62541 PubSub은 RabbitMQ[7]와 Apache에서 제공하는 Qpid-Proton[8] 두 가지 모델을 지원한다. DDS는 오프 소스인 Open DDS[14]를 사용하였다. DDS 게이트웨이 모듈과 브로커는 라즈베리파이 4에서 구동되고, A-Open62541 PubSub 및 DDS Subscriber는 데스크탑 PC의 가상머신에서 실행했다.

본 논문의 구성은 다음과 같다. 본론의 2장은 게이트웨이 플랫폼 설계에 대한 내용이고, 3장은 DDS 게이트웨이 구현 및 실험에 대한 것이다. 마지막으로 결론 및 향후 연구 방향을 제시한다.

## Ⅱ. 본론

### 1. 관련연구

현재 OMG사에서는 OPC-UA/DDS 게이트웨이를 제공하는데, 전체 구조는 그림 1과 같다. OMG사의 OPC-UA/DDS 게이트웨이는 OPC UA client-server와 DDS RTPS를 연결해주는 게이트웨이를 지원하고있다. OPC UA-DDS 브릿지(Bridge)를 사용하면 DDS 어플리케이션이 OPC UA 서버 주소 공간에 있는 정보를 읽고 쓰고 구독할 수 있게 된다[1].

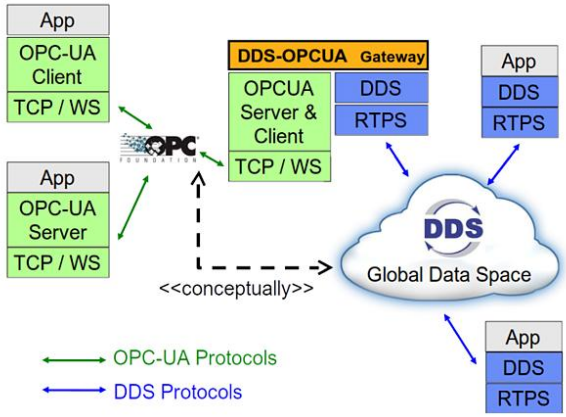


Fig. 1. OMG company DDS-OPC UA gateway structure.
그림 1. OMG사 DDS-OPC UA 게이트웨이 구조

그림 2와 같이 브릿지 한쪽에서는 OPC UA 서비스를 사용하여 서버에 연결하고 일련의 작업을 수행할 수 있는 OPC UA 클라이언트를 인스턴스화 한다. 다른 쪽에서는 게이트웨이가 DDS 측과 필요한 상호 작용을 처리할 수 있는 DataReader 및 DataWriter 를 인스턴스화하는 구조로 구성되어있다[1].

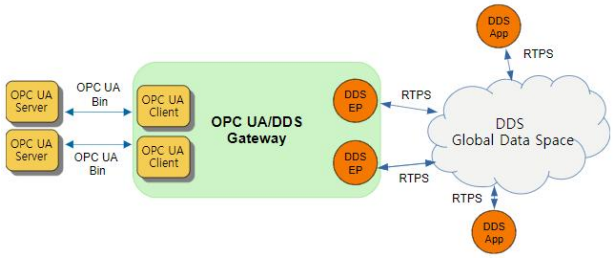


Fig. 2. OPC UA to DDS Bridge.
그림 2. OPC UA - DDS 브릿지

본 논문에서는 OPC UA 클라이언트-서버 구조의 게이트웨이가 아닌 OPC UA part14에 기반한 AMQP/MQTT PubSub 구조로 게이트웨이를 구현하였다. 또한 게이트웨이는 AMQP/MQTT 브로커 내부에 존재하여 OPC UA-DDS 게이트웨이의 기능을 수행 할 뿐만아니라 OPC UA PubSub 기능도 함께 수행할 수 있다.

## 2. DDS 게이트웨이 설계

### 가. OPC UA PubSub

OPC UA PubSub은 OPC UA Part 14[4]로 추가된 표준이다. 그림 3은 OPC UA 전체 프로토콜 스택을 나타낸다.

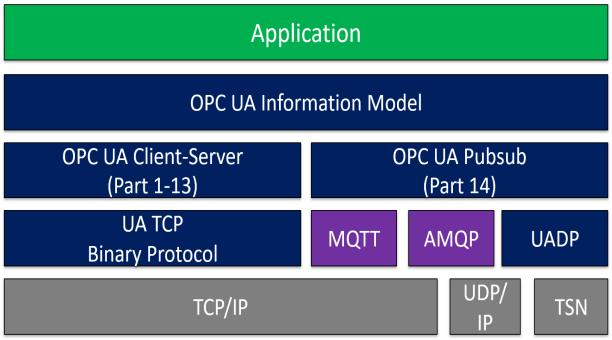


Fig. 3. OPC UA Protocol Stack.
그림 3. OPC UA 프로토콜 스택

현재 OPC UA 표준은 기존 클라이언트-서버 기반의 OPC UA에 OPC UA PubSub이 Part14로 추가된 형태로 확장되었다. OPC UA 통신 모델은 서버에 클라이언트가 추가로 연결될 때마다 새로운 세션(session)을 형성하는 구조이기 때문에 대량의 산업 장치에 서버가 연동되는 경우 성능에 문제가 생긴다.

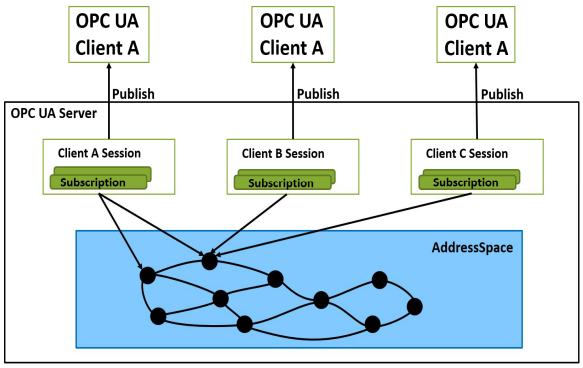


Fig. 4. OPC UA PubSub Communication Model[4].
그림 4. OPC UA PubSub 통신 모델[4]

그림 4는 OPC UA 서버 하나에 3개의 OPC UA 클라이언트가 연결되어 있고, OPC UA 클라이언트는 각각 OPC UA 서버의 'AdressSpace' 공간에서 각 노드들을 정보를 수집하기 위한 저장 공간인 'MonitoredItem'을 통해 구독 신청을 한 상태를 나타낸다. OPC UA 클라이언트는 더 이상 다른 서비스를 요청하지 않고 이미 구독해 둔 노드들에 대한 알림만을 수신하지만, OPC UA 서버는 'Subscription' 서비스를 수행하기 위해 각각의 클라이언트와 연결된 세션을 유지해야만 한다. 각각의 세션에 대해서 통신 주체인 서버와 클라이언트는 전송 버퍼링(buffering), 수신 확인, 데이터 재전송 등의 전송 관련 기능을

개별적으로 수행할 수밖에 없으므로, 이 세션의 수에 비례하여 필요한 통신 자원이 많이 소요된다. 따라서 하드웨어 사양 등의 이유로 통신 자원이 제한적인 서버의 경우는 동시 연결 가능한 클라이언트 개수를 제한하거나, 'MonitoredItem/Subscription'의 수를 제한받을 수밖에 없다.

OPC UA PubSub은 이러한 단점을 극복한 프로토콜이다. OPC UA PubSub은 두 가지 통신 모델을 제안하는데, 하나는 UDP/IP 멀티캐스트(multicast)이고, 다른 하나는 MQTT와 AMQP를 사용한 이용한 브로커 방식이다. OPC UA PubSub은 클라이언트-서버가 직접 연결되지 않고 브로커 및 IP 멀티캐스트 기반으로 연결되기 때문에 연결 개수가 증가하더라도 서버 즉 OPC UA 발행자의 통신 자원에 전혀 영향이 없다. 또한 OPC UA 발행자의 프로토콜 스택이 가볍기(lightweight) 때문에, 16/32 비트 저 사양 필드 레벨에서도 적용이 가능하다. 따라서, 기존 OPC UA에 발행-구독 기능이 추가됨으로써 필드 레벨에서부터 상위 레벨까지 모든 시스템 상호 간의 통신 프로토콜을 OPC UA 기반으로 통합할 수 있게 되었다. 결과적으로 공장 자동화 내의 모든 설비가 OPC UA 및 OPC UA PubSub으로 연계되어 하나의 유기체와 같이 완벽하게 작동할 수 있는 환경을 제공할 수 있다.

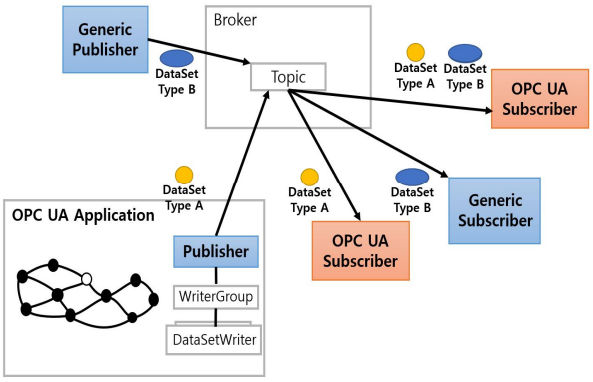


Fig. 5. OPC UA PubSub Broker Communication Model.
그림 5. OPC UA PubSub 브로커 통신 모델

그림 5는 OPC UA PubSub의 브로커 통신 모델을 나타낸다. OPC UA PubSub 표준안은 브로커 통신 방식으로 MQTT와 AMQP을 채택했고, 브로커를 사용하지 않을 경우에는 IP 멀티캐스트를 이용한다. MQTT와 AMQP로 전송할 경우에는 'mqtts://<domain name>[:<port>][/<path>], amqps://<domain name>[:<port>][/<path>]'와 같이 설정하며 MQTT 기본포트는 1883 AMQP 기본포트는 5672로 지정되어있다.

OPC UA PubSub의 메시지 인코딩 방식은 UADP (Unified Architecture Datagram Packet)와 JSON (JavaScript Object Notation) 두 가지 방식을 채택하고 있다. UADP는 최적화된 UA Binary Encoding을 적용한 것으로 메시지의 보안 기능도 제공된다. 이러한 UADP는 UDP/IP 멀티캐스트, 이더넷 및 브로커 기반 통신에 사용된다. JSON은 텍스트 형태의 가독성이 높은 방식으로서 일반적인 데이터 교환을 위한 인코딩 방식으로 사용이 된다.

나. DDS

DDS는 PubSub 기반의 데이터 중심(data centric) 미들웨어로, time/mission-critical 응용을 위하여 OMG에서 권고한 실시간 미들웨어다[10]. 이러한 DDS는 다양한 통신 서비스 품질(QoS : Quality-of-Service)을 제공한다.

DDS는 발행자와 구독자 간에 실제 데이터를 주고받는 토픽이 있다. 발행자, 구독자 및 토픽은 이를 제어하는 서비스 품질이 존재한다. 그림 6은 토픽의 예를 나타낸다.

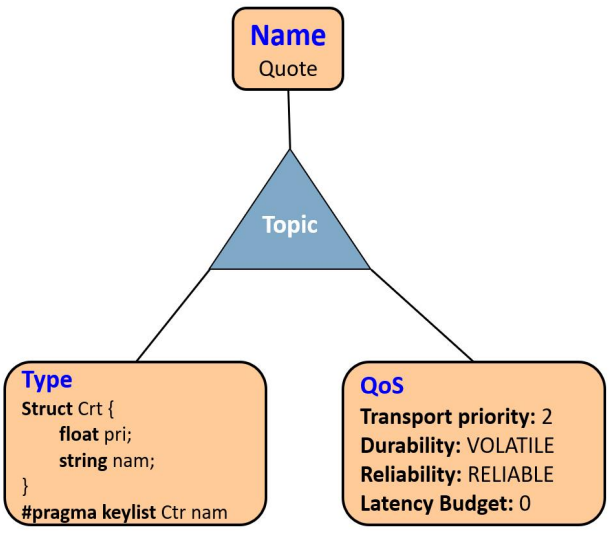


Fig. 6. Example of Topic[11].
그림 6. 토픽의 예[11]

다음 그림 7은 DDS entity의 상호 관계를 나타낸다.

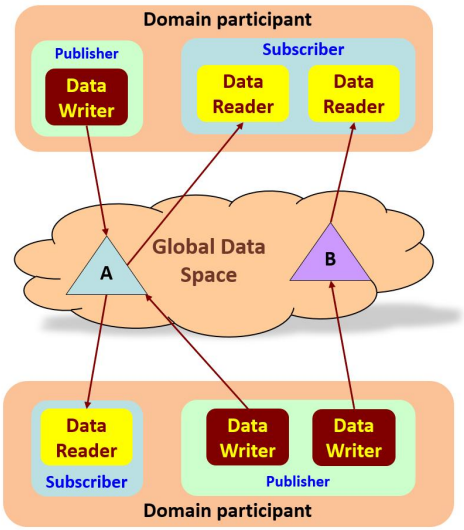


Fig. 7. DDS Entity Interrelationship.
그림 7. DDS 엔티티 상호관계

다. A-Open62541 PubSub

A-Open62541 PubSub은 Open62541 PubSub 오픈 소스를 기반으로, 해당 소스에서 제공하지 않는 기능을 추가 및 구현되어있는 라이브러리를 접근성이 좋은 라이브러리로 교체한 형태로 구현하였다. Open62541 PubSub은 AMQP를 지원하지 않지만, A-Open62541 PubSub은 RabbitMQ 및 Qpid-proton 두 가지 표준을 지원한다. 또한 현재 오픈소스에서도 지원하고있는 JSON과 MQTT는 가장 일반적으로 사용되고 C언어와의 호환성이 높은 JSON-C 및 Mosquitto로 대체하여 구현하였다. 그림 8은 A-Open62541 PubSub과 Open62541 PubSub에서 지원하는 기능을 나타낸다.

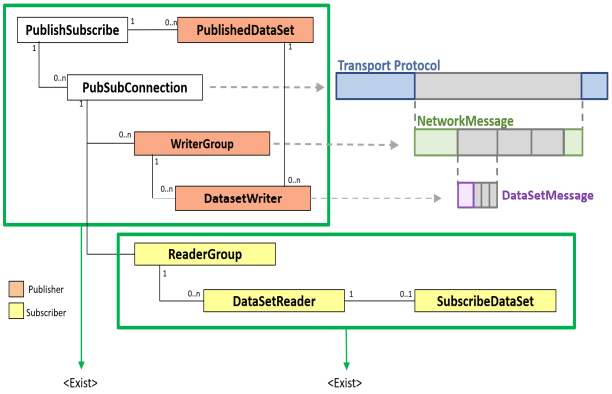


Fig. 8. A-Open62541 PubSub and Open62541 PubSub support features.
그림 8. A-Open62541 PubSub와 Open62541 PubSub 지원 기능

라. 게이트웨이

게이트웨이 플랫폼은 OPC UA 발행자 데이터를 DDS 메시지로 변환한 후 DDS 구독자에게 전달하는 기능을 수행한다. 그림 9는 게이트웨이 플랫폼 전체 시스템 구조를 나타낸다.

게이트웨이에서 OPC UA 발행 데이터와 DDS 메시지의 원시 타입(native type) 매핑을 위한 규칙은 표 1과 같다.

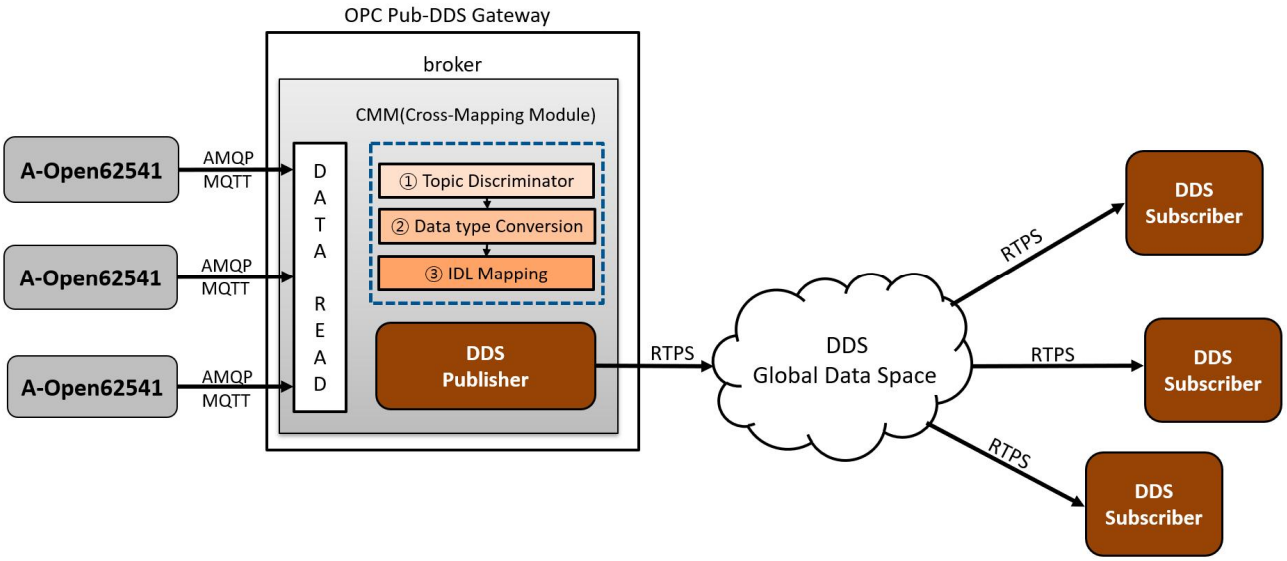


Fig. 9. Gateway overall system structure.
그림 9. 게이트웨이 전체 시스템 구조

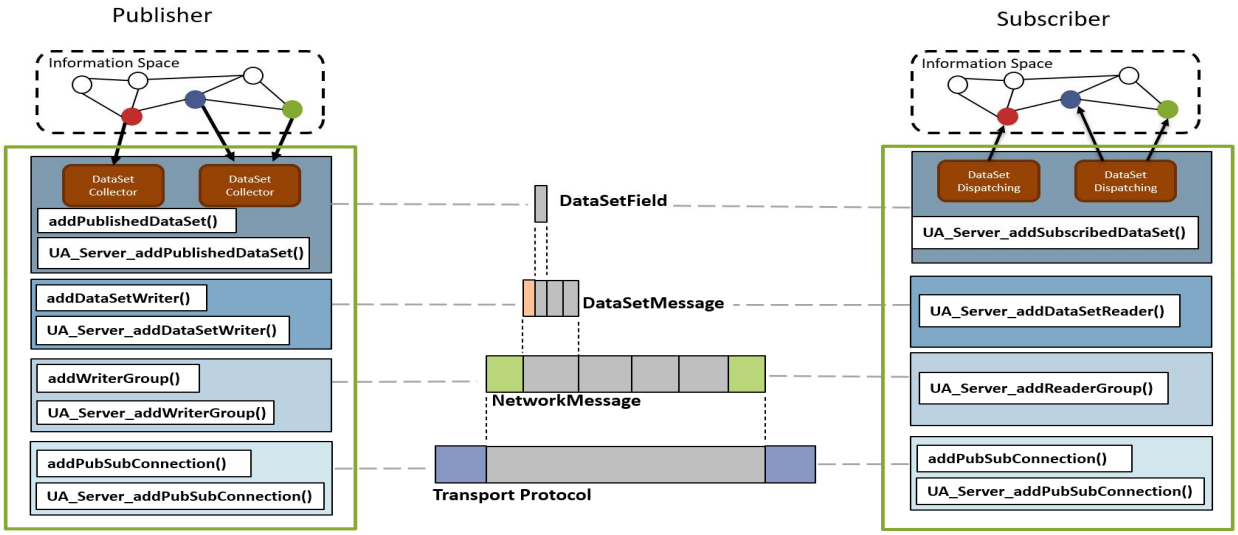


Fig. 10. A-Open62541 PubSub Protocol Stack.
그림 10. A-Open62541 PubSub 프로토콜 스택

Table 1. Mapping of OPC UA PubSub native types to DDS.
표 1. OPCUA PubSub type 매핑 표

| OPC UA PubSub Type | DDS Type |
|---|---|
| Boolean | Boolean |
| Sbyte | Byte |
| Byte | Byte |
| Int16 | Int16 |
| UInt16 | Uint16 |
| Int32 | Int 32 |
| UInt32 | Uint32 |
| Float | Float32 |
| Double | Float64 |
| String | String |
| DateTime | Int64 |
| ByteString | Sequence(Byte) |

데이터 타입 변환은 CMM(Cross-Mapping Module)에서 수행되고, 그 기능은 아래와 같다.

* Topic Discriminator : 브로커로 송신된 데이터 및 토픽을 읽어서, 토픽이 'DDS Gateway' 이면 'Data Type Conversion' 기능을 호출하고 아니면 브로커에 연결된 OPC UA 구독자에게 전달한다.
* Data Type Conversion : OPC UA 발행 데이터는 인코딩된 타입으로 전송되었기 때문에, DDS 구독자가 수신하기 위해서는 해당 타입을 DDS IDL(Interface Definition Language)로 변환해야 하는데, 해당 기능을 수행한다.
* IDL Mapping : 변경된 데이터를 키값과 데이터로 구성된 토픽에 대입하고, 키값은 OPC UA 발행 데이터 길이로 설정한다.

## 3. 구현 및 실험

가. A-Open62541 PubSub

A-Open62541 PubSub은 1절 다항에서 언급을 한 바와 같이 Open6254 PubSub 오픈 소스를 기반으로 AMQP 기능을 추가하는 형태로 구현하였고, JSON과 MQTT는 일반적으로 사용되는 JSON-C와 Mosquitto로 구현했다. 그림 10은 A-Open62541 PubSub의 프로토콜 스택을 나타낸다.

그림 11과 그림 12는 A-Open62541 PubSub의 AMQP API을 나타낸다.

| Open62541 PubSub Mapping func | A-Open62541 PubSub(RabbitMQ) |
|---|---|
| UA_PubSubChannelAMQP_open() | amqp_new_connection()<br>/*Allocate and initialize a new amqp_connection_state_t object*/ |
| | amqp_tcp_socket_new()<br>/*A TCP socket connection. Create a new TCP socket.*/ |
| UA_PubSubChannelAMQP_regist() | amqp_bytes_malloc_dup()<br>/*Duplicates an amqp_bytes_t buffer.*/ |
| | amqp_queue_bind()<br>/*amqp bind*/ |
| | amqp_basic_consume()<br>/*Subsribe a message from the broker*/ |
| UA_PubSubChannelAMQP_send() | amqp_basic_publish()<br>/*Publish a message to the broker*/ |
| | amqp_bytes_free()<br>/*Frees an amqp_bytes_t buffer*/ |
| UA_PubSubChannelAMQP_close() | amqp_channel_close()<br>/*Closes a channel*/ |
| | amqp_connection_close()<br>/*Closes the entire connection*/ |
| | amqp_destroy_connection()<br>/*Destroys an amqp_connection_state_t object*/ |

Fig. 11. A-Open62541 PubSub RabbitMQ API.
그림 11. A-Open62541 PubSub RabbitMQ API

| Open62541 PubSub Mapping func | A-Open62541 PubSub(Qpid-proton) |
|---|---|
| UA_PubSubChannelAMQP_open() | pn_event_connection()<br>/*Get the connection associated with an event.*/ |
| | pn_session()<br>/*Factory for creating a new session on a given connection object.*/ |
| | pn_connection_set_container()<br>/*Set the AMQP Container name advertised by a connection object.*/ |
| | pn_connection_open()<br>/*Open a connection.*/ |
| UA_PubSubChannelAMQP_regist() | pn_event_delivery()<br>/*Get the delivery associated with an event.*/ |
| | pn_link_recv()<br>/*Receive message data for the current delivery on a link.*/ |
| UA_PubSubChannelAMQP_send() | pn_message_body()<br>/*Get and set the body of a message.*/ |
| | pn_data_put_binary()<br>/*Puts a PN_BINARY value.*/ |
| | pn_data_exit()<br>/*Sets the current node to the parent node and the parent node to its own parent.*/ |
| | pn_message_send()<br>/*Encode and send a message on a sender link.*/ |
| UA_PubSubChannelAMQP_close() | pn_connection_close()<br>/*Close a connection.*/ |

Fig. 12. A-Open62541 PubSub Qpid-proton API.
그림 12. A-Open62541 PubSub Qpid-proton API

나. 게이트웨이

게이트웨이 플랫폼은 MQTT와 AMQP 브로커 내부에서 구동한다. 주요 API는 다음과 같다.

* handle_connect() : OPC UA 발행자에서 Mosquitto 브로커로 전달된 데이터를 수신하는 API로 데이터 및 토픽을 확인한다.
* amqp_basic_get() : OPC UA 발행자에서 RabbitMQ 브로커로 전달된 데이터를 수신하는 API로 데이터 및 토픽을 확인한다.
* queue_receive() : OPC UA 발행자에서 Qpid-proton 브로커로 전달된 데이터를 수신하는 API로 데이터 및 토픽을 확인한다.
* check_gateway_topic() : 토픽이 'DDS Gateway' 인지 확인 후 'DDS Gateway'이면, 수신한 데이터를 change _type() API 인자로 전달한다.
* change_type() : 인코딩된 OPC UA 발행 데이터를 DDS에서 사용할 수 IDL 타입으로 변환한다.
* Input_IDL_data() : IDL을 이용하여 DDS 토픽 생성

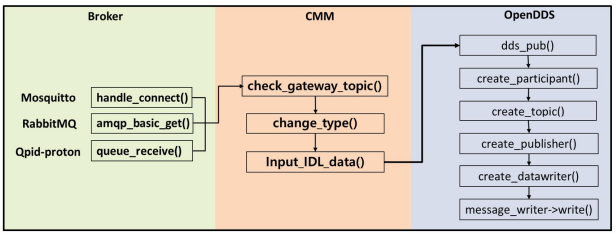


Fig. 13. DDS Gateway Overall Software Structure.
그림 13. DDS 게이트웨이 전체 소프트웨어 구조

그림 14는 IDL 매핑 과정의 예이고, 그림 15는 게이트웨이 수행과정을 의사 코드로 나타낸 것이며 QoS 설정은 그림 16에 나타낸다.

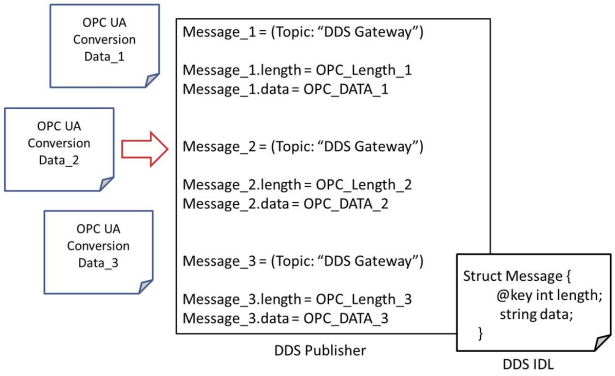


Fig. 14. Example of DDS Gateway IDL Mapping Course.
그림 14. DDS 게이트웨이 IDL 매핑 과정의 예

```
Connect Publisher:
        if(Topic == 'DDS Gateway'){
                if(DDS type ≠ recieve_data type){
                        DDS_message = (DDS type) recieve_data;
                        IDL Mapping(DDS_message);
                        Send DDS_message to DDS_Sub;
                }else{
                        IDL Mapping(recieve_data);
                        Send recieve_data to DDS_Sub;
                }
        }
        else{
                Send recieve_data to OPC_Sub;
        }
}
```

Fig. 15. DDS Gateway pseudo code.
그림 15. DDS 게이트웨이 의사코드

```
// Create DataWriter
DDS::DataWriterQos pin_qos;
publisher->get_default_datawriter_qos(pin_qos);


pin_qos.history.kind = DDS::KEEP_ALL_HISTORY_QOS;

pin_qos.ownership.kind = DDS::SHARED_OWNERSHIP_QOS;

pin_qos.destination_order.kind = DDS::BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS;

pin_qos.reliability.kind = DDS::RELIABLE_RELIABILITY_QOS;
pin_qos.reliability.max_blocking_time.sec=0;
pin_qos.reliability.max_blocking_time.nanosec=100000;

DDS::DataWriter_var writer =
  publisher->create_datawriter(topic,
                               pin_qos,
                               0,
                               0);
```

Fig. 16. DDS Gateway QoS settings.
그림 16. DDS 게이트웨이 QoS 설정

다. 테스트베드 및 실험

게이트웨이는 라즈베리파이 4상에서 구동되고, OPC UA 발행자와 DDS 구독자는 데스크탑 PC의 가상머신에서 실행된다. 그림 17과 표 2는 테스트베드와 시스템 사양을 나타낸다.

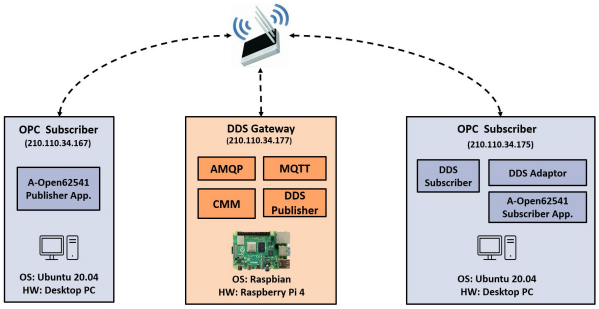


Fig. 17. Test-bed.
그림 17. 테스트베드 구성도

Table 2. Test-bed System Specifications.
표 2. 테스트베드 시스템 사양

| Component | OPC UA Publisher | DDS Gateway | OPC Subscriber over DDS |
|---|---|---|---|
| OS | Ubuntu 20.04 | Rasbian GNU/Linux 10 | Ubuntu 20.04 |
| Tool | A-Open62541 Pub | Mosquitto v2.0.4<br>RabittMQ v3.8.2<br>Qpid-proton v0.33.0<br>OpenDDS v3.17 | A-Open62541 Sub<br>OpenDDS v3.17 |
| Language | C | C, C++ | C, C++ |
| RAM | 4GB | 4GB | 4GB |
| Capacity | 30GB | 32GB | 30GB |
| CPU | Intel Core i7 | ARMv7 Processor rev3 | Intel Core i7 |

PoC(Proof-of Concept) 개발을 위한 데이터 전송 시나리오는 아래와 같다.

* MQTT 기반 OPC UA 발행자 데이터 발행 (Source IP: 210.110.34.167)
* Broker 수신 (Destination IP: 210.110.34.177)
* Broker내의 DDS 발행자가 UDP로 RTPS 메시지 발행 (Source IP: 210.110.34.177)
* DDS 구독자가 UDP 기반 RTPS 메시지 수신 (Destination IP: 239.255.0.2)

그림 18은 OPC UA 발행자에서 게이트웨이 플랫폼으로의 MQTT 메시지 흐름을 와이어샤크를 사용해서 모니터링 것이고, 그림 16\8은 게이트웨이로부터 DDS 구독자로의 데이터 흐름을 나타낸 것이다. 그림 18과 그림 19를 보면 OPC UA 발행자에서 랜덤 데이터인 실수 값 58.0을 JSON 타입으로 보내면, 게이트웨이에서는 해당 타입을 DDS 타입으로 매핑한 후 동일한 값인 58.0으로 DDS 구독자에게 전송하는 것을 확인할 수 있다.

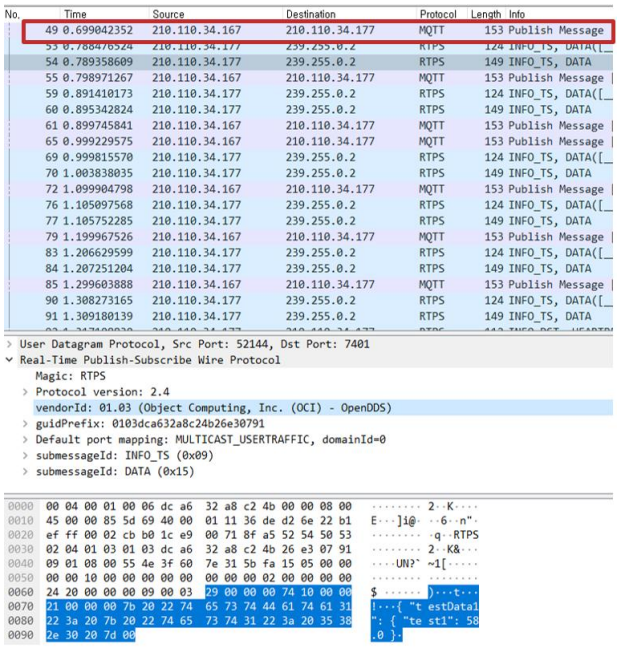


Fig. 18. Wireshark between OPC UA Publisher and Gateway(MQTT).
그림 18. OPC UA 발행자와 게이트웨이 사이의 모니터링 화면(MQTT)

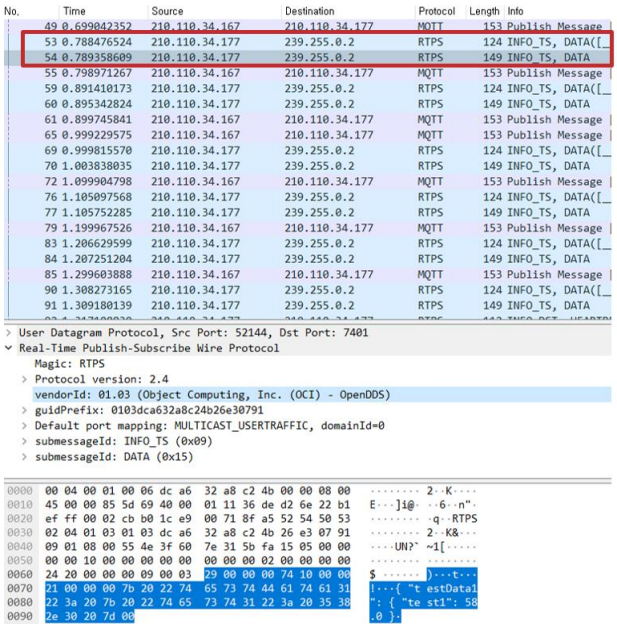


Fig. 19. Wireshark between Gateway and OPC UA Subscriber (MQTT).
그림 19. 게이트웨이와 DDS 구독자 사이의 모니터링 화면 (MQTT)

그림 20, 21은 위의 그림 18, 19를 AMQP로 나타낸 것이다.

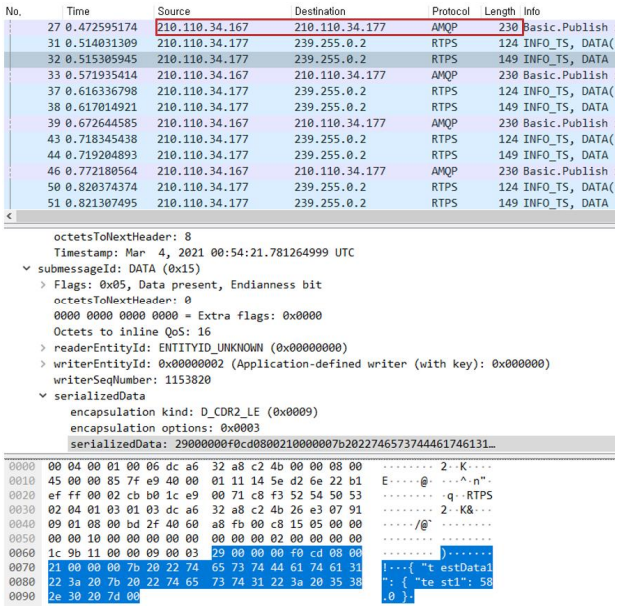


Fig. 20. Wireshark between OPC UA Publisher and Gateway (AMQP).

그림 20. OPC UA 발행자와 게이트웨이 사이의 모니터링 화면(AMQP)

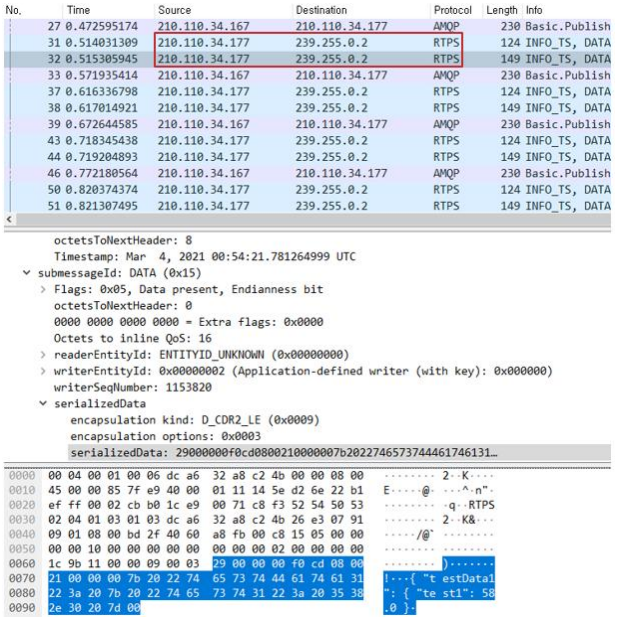


Fig. 21. Wireshark between Gateway and OPC UA Subscriber (AMQP).

그림 21. 게이트웨이와 DDS 구독자 사이의 모니터링 화면(AMQP).

게이트웨이 성능 분석은 처리량(throughput) 기반으로 수행하였다. 그림 22는 OPC UA 발행자와 DDS 구독자가 각각 1개의 경우에 대한 처리량이고, 그림 23은 OPC UA 발행자 10개와 DDS 1개의 경우를 나타낸다. 두 조건 모두 데이터의 크기를 85bytes, 125bytes, 190bytes로 하였고, 데이터 간격은 100ms로 설정하였다.

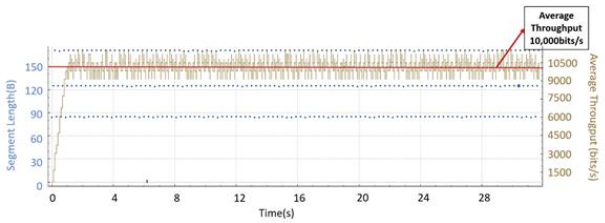


Fig. 22. publisher 1 - subscriber 1 Throughput.

그림 22. publisher 1 - subscriber 1인 경우의 처리량

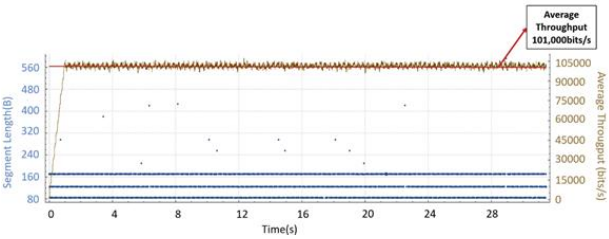


Fig. 23. publisher 10 - subscriber 1 Throughput.

그림 23. publisher 10 - subscriber 1인 경우의 처리량

그림 22의 평균 처리량은 10,000bit/s이며, 그림 21은 평균 처리량이 101,000bit/s로 나타났다. 즉 OPC UA Publisher의 개수에 비례하여 처리량이 증가하는 것을 확인하였다.

## III. 결론

본 논문은 OPC UA Publisher 및 DDS Subscriber 간의 상호운용성 제공을 위한 DDS 게이트웨이에 대한 것이다. 본 논문에서 제안한 DDS 게이트웨이의 주 기능은 OPC UA Publisher로부터 전달된 데이터 타입을 DDS 메시지로 변환하는 것이다. 본 논문에서 사용된 A-Open62541 PubSub은 Open62541 PubSub을 기반으로 AMQP 기능을 구현하였고, JSON parser와 MQTT는 산업계에서 일반적으로 사용되는 JSON-C 및 Mosquitto로 교체하여 구현하였다. 그리고 성능분석을 통해서, DDS 게이트웨이의 처리량과 지연시간을 알아보았다.

OPC UA PubSub은 필드 레벨에서의 실시간성을 보장하는 프로토콜로써, OPC UA와 함께 공장 자동화 레벨 전체를 OPC UA라는 동일한 통신 프로토콜로 통합할 수 있도록 해주고, 또한 MQTT나 AMQP의 연계를 통해, 클라우드를 기반으로 한 IIoT(Industrial IoT) 응용 서비스를 보다 쉽게 구

축할 수 있도록 한다.

기존 연구는 OPC UA와 DDS와의 게이트웨이에 대한 것이 대부분이다. 따라서, 기존에 연구된 결과와 본 논문에서 제시된 기법을 통해, OPC UA 및 OPC UA PubSub 즉 OPC UA 표준을 지원하는 모든 디바이스는 DDS Subscriber와 상호운용성을 확보할 수 있게 되었다.

현재 구현된 A-Open62541 PubSub은 OPC UA Part14에서 권고한 SKS(Security Key Service)를 지원하지 않기 때문에, 향후에는 오픈 소스 SSL (Secure Socket Layer)[12] 및 OAuth2.0[13]을 기반으로 SKS를 구현할 예정이고 또한 TSN(Time Sensitive Network)을 지원하는 임베디드 시스템 [9]에서도 포팅을 할 예정이다.

## References


[1] OMG, “About The OPC-UA/DDS gateway specification version 1.0,” https://www.omg.org/spec/DDS-OPCUA/1.0/About-DDS-OPCUA/

[2] A. Eckhardt, S. Muller, and L. Leurs, “An evaluation of the applicability of OPC UA Publish Subscribe on factory automation use cases,” *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp.1071-1074, 2018.
DOI: 10.1109/ETFA.2018.8502445

[3] F. Prinz, et al, “Configuration of application layer protocols within real-time i4. 0 components,” *2019 IEEE 17th International Conference on Industrial Informatics (INDIN),* pp.971-976, 2019.
DOI: 10.1109/INDIN41052.2019.8972255

[4] *OPC Unified Architecture Specification Part 14: PubSub*, OPC Foundation, 2019. [Online]. Available: https://opcfoundation.org/developertools/specifications-unified-architecture/part-14-pubsub.

[5] Open62541, “open62541: an open source implementation of OPC UA,” https://open62541.org

[6] Eclipse, “Eclipse Mosquitto,” https://mosquitto.org

[7] RabbitMQ, “Messaging that just works - RabbitMQ,” https://www.rabbitmq.com/

[8] Apache, “Apache Qpid Proton - The Apache Software Foundation!,” https://qpid.apache.org/proton

[9] TTTech Industrial, “Edge IP Solution - TTTech Industrial,” https://www.tttech-industrial.com/products/slate/edge-ip-solution/

[10] Object Management Group, “Data Distribution Service (DDS) | Object Management Group,” https://www.omg.org/omg-dds-portal/

[11] I. Etxeberria-Agiriano, I. Calvo, & F. Pérez, “Providing soft real-time capabilities to business applications,” *7th Iberian Conference on Information Systems and Technologies (CISTI 2012),* pp.1-6, 2012. DOI: 10.1109/MCOM.2008.4463780

[12] SpaceNet AG., “OpenSSL,” https://www.openssl.org/

[13] Aaron Parecki, “OAuth 2.0 – OAuth,” https://oauth.net/2/

[14] Object Computing, Inc. (OCI), “OpenDDS,‘ https://opendds.org/

[15] LiamBindle, “LiamBindle/MQTT-C: A portable MQTT C client for embedded systems and PCs alike,” https://github.com/LiamBindle/MQTT-C


## BIOGRAPHY


**Woong-bin Sim** (Member)

2020 : BS degree in Electrical Engineering, Seokyeong University, Seoul, Korea
2020-Present : MS degree in Electronics and Computer Engineering, Seokyeong University, Seoul, Korea

**Byung-kewn Song** (Member)

1984 : BS degree in Electronics Engineering, Korea University
1986 : MS degree in Electronics Engineering, Korea University
1995 : Ph.d degree in Electronics Engineering, Korea University
1995~ : Professor, Department of Electronics Engineering, SeoKyeong University

**Jun-ho Shin** (Member)

1998 : BS degree in Electronics Engineering, Ajou University.
2000 : MS degree in Electronics Engineering, Ajou University.
2004~ : Principal Researcher, Korea Electronics Technology Institute