MASTER

Enabling Omron's IPC for Industry 4.0, M2M and IIoT

Pujari, P.

Award date:
2016

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
Enabling Omron’s IPC for Industry 4.0, M2M and IIoT

Author:
Praveen Pujari

Supervisors:
Dr. J.J.Lukkien
Dr. R.H.Mak
Dr. M.C.W. Geilen
Mr. Dingeman Knaap

August 24, 2016
Acknowledgements

Apart from the great supervision and ideas of Mr. Dingeman Knaap, Dr. J.J.Lukkien, Dr. R.H.Mak and Dr. M.C.W. Geilen I would also like to extend my thanks to:

Mr. Fred Scheffer For his support on OPC-UA development tools and software integration.

Mr. Jasper Spanjers For his support on Sysmac IPC platform and software integration.

Mr. Corne Groen For introducing me to Omron products, tool and software.

Mr. Hendrik Beyeman For his support with Sysmac IPC development flow.
## Contents

**Abbreviations**

**Terms and Definitions**

### 1 Introduction

1.1 Background ........................................... 8
1.2 Motivation ........................................... 9
1.3 Scope of Project ................................. 10
1.4 Outline of the thesis ......................... 10

### 2 Case Study - Poultry Processing

### 3 Communication Protocols in Industrial Automation

3.1 Industrial Automation Levels ................. 18
3.1.1 Device Level ................................ 18
3.1.2 Control Level ................................ 18
3.1.3 SCADA/MES Level ............................ 19
3.1.4 ERP Level .................................... 19
3.2 OPC-UA for IIoT and Industry 4.0 ......... 20
3.2.1 OPC-Classic ................................ 20
3.2.2 OPC - Unified Architecture ............ 21
3.3 LWM2M for IIoT and Industry 4.0 ........... 24
3.4 Real-Time Control and Information Exchange .
3.4.1 Control ........................................ 25
3.4.2 Information Exchange ......................... 25

### 4 Information Modelling

4.1 OPC-UA Information Models .................. 27
4.2 Representation of Information models ....... 29
4.3 Companion Specifications ..................... 30
4.4 Sysmac IPC: Information Modelling ........ 30

### 5 Communication Architectures Using Sysmac IPC

5.1 Sysmac-IPC: Features .......................... 34
5.2 Inter-OS Communication ....................... 35
5.2.1 Secure communication over Virtual Ethernet 36
5.2.2 Publish-Subscribe Data Exchange ....... 38
5.3 External Communication ....................... 46
5.3.1 Client-Server ................................ 46
5.3.2 Publish-Subscribe (pub-sub) .............. 47

### 6 LWM2M/OPC-UA Protocol Translator

6.1 Introduction ....................................... 48
6.2 Comparison between OPC-UA and LWM2M ....
6.2.1 LWM2M .......................................... 49
6.2.2 OPC-UA ................................................................. 52
6.3 Implementation .......................................................... 54
   6.3.1 Translating OPC-UA services to LWM2M operations ............... 57
6.4 Evaluation ................................................................. 61

7 Results and Recommendations ................................................ 62
   7.1 Results ................................................................. 62
   7.2 Recommendations .................................................. 62

8 Conclusions ................................................................. 64

References ................................................................. 65
Abbreviations

AI - Analog Input
AMQP - Advanced Message Queuing Protocol
AO - Analog Output
API - Application Program Interface
B2M - Business to Machine
C2C - Controller to Controller
CIP - Common Industrial Protocol
CoAP - Constrained Application Protocol
COM - Componet Object Model
CPS - Cyber Physical Systems
DA - Data Access
DCOM - Distributed Componet Object Model
DI - Digital Input
DO - Digital Output
ERP - Enterprise Resource Planning
FINS - Factory Interface Network Service
GPOS - General Purpose Operating System
HDA - Historical Data Access
ICT - Information and Communication Technology
IIoT - Industrial Internet of Things
IO - Input/Output
IoT - Internet of Things
M2M - Machine to Machine
MES - Manufacturing Execution System
MQTT - Message Queuing Telemetry Transport
OLE - Object Linking and Embedding
OPC-Classical - OLE for Process Control
OPC - UA Open Process Connectivity-Unified Architecture
PLC - Programmable Logic Controller
RTOS - Real Time Operating System
SCADA - Supervisory Control and Data Acquisition
SDK - Software Development Toolkit
SOA - Service Oriented Architecture
SSL - Secure Socket Layer
TLS - Transport Layer Security
TSN - Time Sensitive Networking
URI - Uniform Resource Identifier

Terms and Definitions

Communication Latency: End to end communication delay between client and server or publisher and subscriber.

Component Object Model: Component Object Model (COM) is a binary-interface standard for software components introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in a large range of programming languages. More information can be found in this book [1].

Cyber-Physical Systems (CPS): CPS are integrations of computation, networking, and physical processes. Embedded computers and networks monitor and control the physical processes, with feedback loops where physical processes affect computations and vice versa.

Hypervisor: Hypervisor which is also known as a virtual machine monitor, is a layer of software that hosts other operating systems. There are two types, Type-1 is called as bare metal hypervisor, which runs on the hardware directly and hosts other operating systems. Type-2 runs on an operating system hosts guest operating systems.

Industry 4.0 [6]: Industry 4.0 or the fourth industrial revolution, is a collective term embracing a number of contemporary automation, data exchange and manufacturing technologies. The four phases in industrial revolution are,

- Industry 1.0 - Water/steam power
- Industry 2.0 - Electric power
- Industry 3.0 - Computing power
- Industry 4.0 - Internet of Things (IoT) power

The following are the technology areas that underpin Industry 4.0 and every utility or enterprise which wishes to keep up with the trend must be aware of.

- Horizontal and vertical system integration
- The Internet of Things
- Cybersecurity
- The cloud
- Big data analytics
- Simulation
- Additive manufacturing
- Augmented Reality
- Smart robots

**Internet of Things**: The Internet of Things (IoT) is the network of physical objects, devices, vehicles, buildings and other items which are embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

**Interoperability**: Interoperability is the ability of a product or system, to work with other products or systems.

**Machine to Machine (M2M)**: Machine to Machine communication refers to the direct interaction between two devices using any communication channel. With recent developments in light weight communication protocols in Internet of Things, it is possible to achieve direct communication between any two devices without the need for a relaying device in between. For example, the information gathered by a sensor can be directly sent to an application that uses it, which was not the case before.

**Object Model**: A collection of objects or classes through which a program can examine and manipulate some specific parts of its world. In other words, the object-oriented interface to some service or system. Such an interface is said to be the object model of the represented service or system.

**ODVA**: A global association whose members are comprised of the world’s leading automation companies. ODVA’s mission is to advance open, interoperable information and communication technologies in industrial automation.

**ReST- Representational State Transfer** [2]: It is an architecture style commonly used in distributed resource oriented applications such as the internet. Protocols adhering to ReST style are usually called ReSTful and follow client-server communication style. ReSTful protocols have four important properties, namely

- **Addressability**: Addresses are assigned to resources that act as their identifiers and that contains enough information to establish communication and operate on resources.

- **Statelessness**: By stateless it means that the server does not store any state about the client session on the server side. Statelessness is one of the properties of ReSTful protocols, which makes it possible for the server to service a large number of clients simultaneously from a set of infinite number of clients. HTTP and CoAP are examples of stateless communication protocols. The client’s application state is not stored on the server, but passed from the client to every place that needs it. That is where the ST in ReST comes from, State Transfer. The state of the client is transferred instead of storing it on the server. This makes it possible to scale to millions of concurrent users. Otherwise million users would need a million sessions to be stored on a server.

- **Connectedness**: Explains how resources reference each other in their representations via their identifying addresses.
Uniform Interface: The same set of services are used to interact with all resources. For example, the available operations in HTTP (GET, POST, PUT, DELETE, and so on) are the same for every website.

**Scalability:** Scalability as described by André B. Bondi in [3], is the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement.

**Service:** In the context of this project “service” is a function exposed by server and which can be used by clients running on a single device or on different devices over a communication network. Example: Read Service - allows clients to read information contained within the server.

**Service-oriented architecture:** An architectural design pattern in which application components provide services (functions) to other components via a communication protocol, typically over a network.
1 Introduction

1.1 Background

With advancements in ICT, CPS, embedded electronics, machine learning, cloud computing, etc. current industries are heading towards the fourth industrial revolution aka Industry 4.0 or Smart Factory. Industrial Automation is one of the driving forces for the technological developments that we have seen in the last decade.

Figure 1: Multilevel automation architecture [16]

Figure 1 shows a multilevel Industrial Automation Architecture, which consists of four levels namely Device Level, Control Level, MES Level and ERP Level. Functions, responsibilities and communication protocols of each of these levels will be discussed with respect to the Poultry Processing Industry as a case study in chapter 2. Communication protocols in Industrial Automation typically revolve around real-time control and information exchange of process data within the same level or across multiple levels. The latency requirements imposed by these communication protocols on the communication network vary across the levels. Communication Protocols used in the Industrial Automation will be discussed in detail in chapter 3.
OMRON Industrial Automation functions as a partner to help innovate worldwide manufacturing with precision sensing technology, actuators, PLCs, IPC, safety and robotics. OMRON focusses on improving productivity in many industries. OMRON is developing a new product called Sysmac IPC as shown in Figure 2 which will be used in the multilevel automation architecture shown in Figure 1.

It consists of a hardware platform on which it runs two operating systems, namely a GPOS and a RTOS. Apparently, the characteristics of the hardware platform remain unspecified in this section, but will certainly become important once real-time constraints enter the picture. In this project Windows OS is used for general purpose and RTOS is used for real-time control and the choice of operating systems may vary in future. A software PLC runs on RTOS which handles the real-time control of processes in a factory. A software bus exists between the operating systems for exchange of PLC data. Details of the software bus will be discussed in chapter 5.

1.2 Motivation

With Industry 4.0 and IIoT taking the center stage in Industrial Automation, Omron wishes to contribute new communication architectures and features which are relevant to Industry 4.0 and IIoT environments using the Sysmac IPC. These features may include but not limited to support for traceability applications, uniformity of protocols enabling the ease of integration, support for monitoring and logging applications etc. The purpose of this project is to answer questions such as,
1. Where does the Sysmac-IPC fit in a multilevel Industrial Automation Architecture and how can industries benefit from it?

2. What are the communication protocols to be used by the Sysmac-IPC in the context of real-time information reporting in Industry 4.0 and IIoT domains?

3. How to model the information in and out of the Sysmac-IPC? What are the important considerations to be made while selecting an information model based on communication protocols such as OPC-UA?

4. How to exchange data between the two operating systems in Sysmac-IPC securely and at the same time achieve high data throughput? What are the possible integration environments and communication architectures for the Sysmac-IPC?

5. How to incorporate the advantages of IoT protocols such as low resource requirements within Industry 4.0 applications?

6. What are the future integration possibilities based on ongoing research in the field of communication architectures & infrastructures?

1.3 Scope of Project

The project studies the questions raised above in a context where the communication protocol has already be chosen to be OPC-UA, a popular protocol for Industry 4.0 and IIoT. Proper motivation for selecting OPC-UA as the protocol when compared to other protocols is provided in the chapter 3 of this report. New architectural solutions using the Sysmac-IPC and the OPC-UA protocol is provided. The architectural solutions provided aim to address the information modelling challenges and other requirements of a typical poultry processing industry, gathered through a case study. In an effort to combine IoT and Industry 4.0 a protocol translator which translates application level services of the OPC-UA protocol to communicate with a device using IoT protocol (LWM2M) is provided.

1.4 Outline of the thesis

Chapter 2 provides a case study of Poultry Processing industry which helps in understanding the requirements of Industry 4.0 and IIoT. Chapter 3 briefly discusses the role of communication protocols and their requirements in the context of this project. Chapter 4 addresses the information modelling challenges associated with communication protocols in the Industrial Automation domain and provides suitable solutions. Chapter 5 provides new communication architectures using the Sysmac-IPC and brief evaluation of their performance in terms of CPU utilization, network latency etc. Chapter 6 discusses the implementation of protocol translation or extension of an IoT protocol using a Industry 4.0 protocol to explore possibility of new high level applications at lower costs. Chapter 7 provides the results of the project followed by conclusions and future recommendations in chapter 8.
2 Case Study - Poultry Processing

This chapter details the current scenario of Industrial Automation and future requirements by considering poultry processing as an example. The poultry processing has various stages as shown in Figure 3.

Each of these stages shown in the flow diagram involves a lot of equipments or machinery as shown in Figure 4, hence control of the machinery to achieve a desired task, such as killing (slaughtering) or de-feathering is a primary goal of Industrial Automation devices and networks.

Figure 3: Poultry Processing: Flow Diagram
Traditionally the control of equipments used to realize a process was the main goal of Industrial Automation. Nowadays this goal is extended with the need to monitor such a process in real time from any location along with local control of individual stages to achieve overall control and optimization of the entire poultry processing. The devices, communication protocols, and network performance required to achieve this vary with type of industries. There are two important aspects in the Industrial Automation namely process control and information exchange. Some protocols such as Profinet/IO, EtherCAT, Modbus, etc. are used for control while protocol such as OPC-UA is mainly used for information exchange. Chapter 3 focusses on these things.

We studied the requirements of some of the biggest players (companies) in food processing all over the world. They thrive to maintain the competitive edge and business continuity by providing solutions that increase production capacity, maximum performance and guaranteed up-time. They highly invest in R&D in order to provide efficient, effective and sustainable solutions to their customers. All these companies which manufacture equipments used in poultry processing and provide end to end solutions are now focussed on technological developments in the Industrial Automation, especially the idea of “Smart Factory”. Smart Factory enables access to data from the factory through the internet which makes planning, maintenance and monitoring of the poultry processing possible from a central location. They want
their machinery to be integrated with the complete poultry supply chain: Following and planning the whole process from hatching to the supermarket. For example, visualization of the real-time data gathered from all the machinery about the status of machines, number of birds being processed, expected yield, etc. would help better planning of delivery, guarantee higher up-time, change machine settings to adapt to the demanded yield, etc. Consider a scenario where a user inputs expected yield from MES/ERP system and the whole automation process is smart enough to adjust the machine settings to meet the required yield. Consider other possible scenario where a machine located at the customers factory issues a service-request to the concerned parties, when it needs preventive maintenance.

For example, consider a typical poultry processing operation. There are several automated steps as shown in Figure 3 that occur throughout the process such as stunning, killing, evisceration, water chilling, weighing, and packaging. This is by no means an exhaustive list for a poultry process, but allows us to visualize how a networked system can benefit this type of automated process.

First, one must consider the number of sensors that exist in these processes and the information they convey, such as equipment status, water temperature, air temperature, weight and packaging material remaining. The complexity involved in managing this type of operation is tremendous, and some of the parameters have the potential to cause a substantial impact on the overall throughput of the facility. For example, if the chiller water temperature was too high, a large number of products could be affected, thus bringing the process to a halt.

It is apparent that data collection and analysis can add considerable value to this process. The expected data volumes are high, but not exactly known yet. Using OPC-UA protocol for communication with MES/ERP/Cloud is strongly being considered. OPC-UA is easily portable on-to small embedded devices which makes uniformity of communication protocols possible. Current OPC-UA specification is client-server style, it is yet to define publish-subscribe communication style which is still in draft stage. Once released OPC-UA publish-subscribe architectures can be used in these industries to further optimize the real-time communication by providing flexible architectural solutions for resource constrained environments. We want to show that Omron can be a perfect partner to companies providing industrial automation infrastructure in the future by proposing suitable solutions in this regard. Using the Sysmac-IPC for control and higher level applications helps customers to achieve better performance in terms of communication latency, saves the costs for separate SCADA/Monitoring application PCs and provides user friendly configuration through GPOS (Windows).

The technical requirements of poultry processing industry, addressed by this project are briefly explained below.

- **Interoperability and easy integration:** Big players target total line (example: intake of birds till the packaging of meat) solutions rather than single machines as they need to control the whole process. Due to historical issues (separated developments, different technologies on different parts of the line, etc.) the line architecture is complicated with multiple networks, proprietary systems and communication protocols etc. which makes further developments difficult. These companies can really benefit from an architecture which enables uniformity with respect to the communication protocols and easy integration using standardized interoperable protocols such as OPC-UA.

- **Cyber Security:** As the machines get more connected, cyber security becomes more
important which forces poultry processing industries to upgrade to modern, secure communications. As the low level data in the field equipment (PLC) becomes more valuable (e.g. For high level applications such as monitoring and traceability), in which factories need to have data integrity & management practices and related infrastructure on the line. Using secure protocols such as OPC-UA (confidentiality & authentication) in their communication architecture solutions really helps which is one of the most important reasons why OPC-UA has gained popularity all over the world.

• **Horizontal (C2C) and vertical integration (B2M):** End users need to integrate their supply chain and manufacturing with on-line marketing systems as well. If supermarket learns on Tuesday that the weather will be sunny during the weekend, they must be able to directly switch to barbecue related products for the weekend without having a big backlog of other products. PLCs controlling individual processes must have some kind of Controller to Controller (C2C) communication to achieve the desired goals and few or relevant PLCs should also be integrated with MES/ERP or Cloud based applications. Hence, architectural solutions using M2M (in general) communication protocol such as OPC-UA on the Sysmac-IPC not only solves the problem, but also brings in additional possibilities of using RTOS for C2C, Windows for B2M and so on in one device or system.

• **Traceability:** The processing steps and conditions to which the final food product has been subjected need to be retrievable. For example, if someone falls sick or complaints about the quality of the meat, it should be possible to trace back to the equipments or the process involved in producing that pack of meat. Realizing such high level applications need collection of huge data from lower levels in industrial automation and efficient ways of accessing such a collection of data. Such applications can benefit from OPC-UA Historical Data Access information model.

• **Real-time monitoring:** For continuous monitoring of poultry processing lines which are involved in producing chicken legs, weighing machines can be placed at the end of each line. The weight of every chicken leg is reported to a monitoring system which compares them with the weight of previous chicken leg from the same line and chicken legs from other lines. An alarm can be triggered if there is an anomaly and the line can be immediately fixed. Such real-time monitoring applications require time-stamped data which contain enough information to locate the cause of that anomaly. The cost of such a system depends on the weighing machines and samples of data per second. Let’s assume a new chicken leg is produced every 10 seconds, which requires an update rate of 6 samples/minute for each line. Assume there are 100 lines, which will collectively require an update rate of 600 samples/second. Since the communication infrastructure has to be shared with real-time control applications, monitoring data is expected to consume less network resources. For time-stamped historical data access one can go with the OPC-UA communication protocol in their weighing machines, however LWM2M communication protocol in this scenario fits the best. LWM2M is for low resource devices and it is ReSTful hence consumes less network bandwidth in this scenario than session based & non-ReSTful OPC-UA. In the project proposes a combination of these two protocols and how a monitoring system can benefit from this. Further, using TSN over Ethernet offers prioritization of data streams, time critical control can be given higher priority over monitoring application’s data stream.
• **Information Modelling:** The data in a poultry processing farm have to reach various geographically distributed higher level applications such as monitoring, tracing, control, analysis, business planning, etc. Users expect process transparency from the received data for example, Factory1/Line1/Motor1/RPM. Also the data to various high level applications from different vendors expect a certain level of standardized representation in terms of data types (int, float, date-time, etc.). Modelling of information is a rigorous task, which can benefit from user friendly tools. This project proposes OPC-UA information models and tools to be used with the Sysmac-IPC in Chapter 4.

• **Accountability:** If an operator changes certain parameters knowingly or unknowingly which results in catastrophic situations or financial losses, the automation system should be able to assist an investigator to find the person responsible. Data authorization and role based access control help provide accountability. The OPC-UA communication protocol satisfies such requirements.

The communication protocol in chapter 3, information models 4, protocol translation 6 and architectures 5 proposed through this project are driven by the above requirements.
3 Communication Protocols in Industrial Automation

This chapter discusses the answers to research questions 1 & 2 in motivation section 1.2. As a part of master’s thesis preparation, industrial communication protocols used in various levels of automation hierarchy namely Device level, Control level, MES level and ERP level were studied. This chapter briefly discusses these protocols, in addition a comparison in terms of security, inter-operability, performance, features etc. will be made. Finally, a bit more detailed information about the OPC-UA protocol and the motivation behind selecting this protocol to be used with the Sysmac-IPC will be provided.

Figure 1 shows the four levels in Industrial Automation and Figure 5 shows communication protocols used by Omron in their Industrial Automation architectures with the Sysmac-IPC in picture where SCADA/MES and Control levels reside on one Sysmac-IPC device. Each of these levels have their own requirements and features which are briefly discussed in this chapter. An important issue to be noted is the number of different protocols used across various levels, which affects the ease of integration and uniformity.

Sysmac-IPC fits on Control Level and SCADA/MES Level in the hierarchy, hence it needs to interact with other Control and SCADA Level devices (horizontal) with in the plant or factory. Also, Sysmac-IPC needs to interact with ERP systems (Vertically up) and Fieldbus devices (Vertically down) Device Level.
Figure 5: Multilevel automation architecture using Sysmac-IPC
3.1 Industrial Automation Levels

Consider the poultry processing industry as an example and try to map its functions/tasks on various levels in the Industrial Automation. For example, a bird entering a processing factory goes through various stages such as slaughtering, de-feathering, washing, evisceration etc before the processed meat is packed. Each of these stages have a number of sensing, actuating and mechanical devices which need to be coordinated and controlled to achieve a high level task. The whole process is recorded and a log of everything that is happening within the plant is required for analysing, improving the process etc. which is achieved using SCADA/MES systems. A poultry processing company would have several factories to supply globally and they would prefer to have access to relevant information for financial planning, billing, resource planning etc. from a central corporate office which are achieved using ERP systems or Cloud based applications.

3.1.1 Device Level

EtherCAT, Profinet, ModbusTCP, EtherNet/IP etc. are the most commonly used Ethernet based communication protocols at this level. All these protocols are standardized and hence devices using the same protocol are inter-operable. However, integration problems still exist due to huge list of fieldbus protocols. The Device Level consists of complex devices which are wired to sensors and actuators. For example an EtherCAT slave device has a number of analog/digital IO modules and communicates to an EtherCAT master device in the Control Level or within the Device Level. For example, temperature of a boiler is sensed by a temperature sensor and will be sent to a PLC via EtherCAT slave device.

The communication protocols are expected to support update rates less than 100µs over Ethernet. Communication security at the Device Level was not a serious concern until recently, however the scenario is different now where message authentication and confidentiality are required. The DeviceNET or Profinet standards have not defined any means of secure communication, users depend on external measures such as firewall, access-control etc. to secure the communication. Communication protocols such as EtherCAT are widely used in industry because of its flexible topology, redundancy, high scalability, sub nano-seconds time synchronization etc. Omron’s Sysmac-IPC supports EtherCAT interface on RTOS.

Horizontal communication between devices in this level is insignificant because the devices interact directly with PLCs vertically.

3.1.2 Control Level

The Control Level has hardware/software PLCs which control the process such as slaughtering, washing, temperature control etc. with the help of devices in the Device Level. PLCs have programs with inputs from the Device Level and take a control action such as turning on/off a machine through program outputs communicated back to the Device Level. Automated control actions typically are hard real-time in nature which happen within 3 to 5 milliseconds. The Sysmac-IPC on the controller side has data update cycle of 500µs and synchronization within 1µs. PLCs also include manual control for maintenance activities.
which are soft real-time in nature. Profinet and EtherCAT are commonly used protocols for real-time control over Ethernet in PLCs.

Apart from control related activities PLCs also report the underlying process information to SCADA/MES or Monitoring applications. Information exchange between the Control Level and SCADA/MES Level require latency between 100 milliseconds to 1 second. OPC classic, OPC-UA and EtherNet-IP are commonly used protocols for information exchange and supervisory control between the Control Level and SCADA levels. Communication to/from PLCs has to be secured because an attacker with malicious intention can cause huge financial losses if an undesired control action were to be performed. Hence, access-control to PLCs, user-authentication over messages and confidentiality of messages are desired. In scenarios where the Control Level is completely isolated from higher levels, network security is not so crucial.

Horizontal communication between devices in this level is mainly between other other PLCs which is also known as C2C communication. One or more PLCs can collectively perform a control action or PLCs may just communicate with redundant PLCs.

### 3.1.3 SCADA/MES Level

SCADA/MES systems monitor and control the whole poultry processing. PLCs from the Control Level which are controlling individual tasks such as slaughtering, de-feathering, washing etc. will communicate with SCADA/MES/Monitoring systems which enable a complete control/overview of poultry processing. Access to SCADA/MES applications are limited by user accounts in Operating Systems. Communication between the Control Level and SCADA level has to be secured. OPC-UA is commonly used because of its security and other features which will be discussed later in this section. The devices at this level communicate with ERP/Cloud based applications through the Internet where an ERP system can initiate work orders, improve production etc. The vertical communication uses OPC-UA or HTTP protocols and communication latency up to few seconds/minutes is tolerated.

Horizontal communication between devices in this level is mainly between SCADA, MES, Monitoring systems etc. for information exchange or redundancy.

### 3.1.4 ERP Level

ERP systems or Cloud based applications provide high level overview and control over the business. For example, if weather prediction for next day is warm and sunny an ERP system can trigger production of more barbecue meat than usual. The real-time communication at this level can have latency up to few seconds.

Now that we have a basic understanding of individual levels in the Industrial Automation, let us look into IoT and Industry 4.0 real-time communication protocols.
3.2 OPC-UA for IIoT and Industry 4.0

OPC-UA is probably the hottest topic recently in Industrial Automation, Internet of Things and Industry 4.0. Before elaborating on OPC-UA we discuss the history of OPC standards.

3.2.1 OPC-Classic

A common way of connecting an underlying process to an application is shown in Figure 6. These processes are usually running in a device at various levels of multilevel automation architecture shown in Figure 1. A device can also have multiple processes running in it.

A PC running the application which analyses data from devices and takes certain actions had to have as many custom drivers as the number of devices/processes that it used to connect to. There are two important aspects of this which led to the introduction of OPC.

- Devices could be overloaded with increased communication to different applications eventually consuming all CPU time for communication and no time to perform process functions such as sampling I/O data, protecting equipments connected to safety device etc.

- A PC containing application must have suitable drivers for process data communication from devices. Which could be a headache for end users because the devices are manufactured by different vendors and hence come with custom drivers.

The problems were highlighted in the early 90s when the number of embedded devices for control and protection in Industrial Automation increased drastically. OPC was introduced to address interoperability of these devices. Also the scalability problems with respect to number of devices connecting to multiple applications simultaneously. Figure 7 shows a new architecture for communicating process data from devices to applications.

The idea was very simple, introduce a "Middle Layer" which can communicate with underlying process data using proprietary protocols same way as before but relay data to applications.
in a standard way. This layer is called the "OPC-Server". Now the processes in devices have to communicate with only this OPC-Server and hence no performance burden due to communication. The OPC-Server can replicate the data and send it to different applications in a common format specified by OPC-specifications, which improves scalability. The applications must have a common client interface that can communicate with OPC-Server. This solves the second problem of custom drivers. OPC soon became very popular and gained acceptance in Industrial Automation. Several specifications of this version of OPC exist and a brief survey is presented in [5]. Since OPC server takes care of all the underlying communication with devices it is sufficient if applications have once "OPC client" that communicate with "OPC server", hence inter-operability of devices with respect to applications was achieved.

Disadvantages:

OPC was originally called OLE(Microsoft’s Object Linking and Embedding) for process Control and was developed on Microsoft’s COM/DCOM technology. Even though it was widely accepted it had problems with use on Linux systems making it platform dependent. The security was dependent on underlying operating system(Mostly Windows).OPC Classic lacks the ability to adequately represent the kinds of data, information and relationships between data items and systems that are important in today’s connected world. The increase in number of specifications such as Data Access, Alarms and Events, Historical Data Access needed different OPC-Servers hence recreating "custom drivers" problem. This classic-OPC could not cope with new industry trends such as Internet of Things and Industry 4.0. Hence in 2006 OPC Foundation took a new approach to OPC which resulted in OPC-UA

3.2.2 OPC - Unified Architecture

Even though OPC-Classic was popular the security, platform dependency and scalability issues led to development of alternate standards or change in the OPC paradigm. OPC-UA
is the collective effort of collaboration between open industry standards, manufacturers and OPC Foundation. OPC-UA specification is available in 14 parts and are available here [9] for free. OPC Classic was strictly client-server based connectivity solution while OPC-UA takes SOA (Service Oriented Approach). The OPC-UA standard introduces security, reliability, scalability and most importantly eliminated platform dependency. This new transformed version of OPC is open, allowing small embedded systems to be connected to internet securely and uniformly.

Features

- Platform Independent: We saw how OPC-Classic was bound by Microsoft’s COM/DCOM technology. OPC-UA can be seen as web-services for Industrial Automation but the services are standardised. OPC-UA has a low sized protocol stack written in ANSI C, hence can be ported on to small embedded systems. This freedom related to platform has made OPC-UA to be considered for ”Internet of Things”.

- Services: OPC-UA provides a suite of standardised services for data access, alarms, events, historizing etc.

- Address Space Flexibility: The OPC-UA is object oriented. Variables and methods are contained in an object. These objects are represented as nodes which could be referred and inherited by other nodes. Objects can be organised with complete freedom and standard does not define any mandatory objects. Objects can be type defined and referenced by other objects. Address space support methods which can be executed from remote clients.

- Common Protocol Suite: OPC-UA uses common protocol suites and encodings and hence makes it possible to be used with internet.

- Information Model: OPC-UA defines means to exchange useful information through its adaptable information model which can be integrated with industrial data models.

- Security: OPC-UA defines a robust security model to provide application to application security. The security model aims to provide user authentication, role based access rights, secure end to end communication through encryption. It uses same type of security as internet.

- Process Transparency: The Object Oriented approach of address space and different Information models help in representing the underlying process data transparently to the client-side users. Hence end users need not understand underlying technology or data representation. Example: A block of 1024 bytes contain all the data gathered from various IO devices, only an engineer with knowledge of proper offsets can decode the data. In case of OPC-UA each IO device can be represented as an object using variables, suitable data types and description of IO device.
OPC-UA Security

In OPC-UA security related decisions are not left to users but are defined in the specifications. Part 2 of the specifications describes security threats to OPC-UA based systems, sets Security Objectives and defines a security architecture [9]. Part 4 of the specifications defines a list of services to be provided by clients and servers to achieve the intended level of security. These services are used to discover servers, authenticate the application, create a secure communication channel and open a session where information can now be securely communicated. Part 6 provides the mapping of these services to suitable technology.

![OPC-UA Security Architecture](image)

Figure 8: OPC-UA Security Architecture

The security architecture displayed in Figure 8 shows the security aspects achieved in each layer. User authentication and authorization is handled in application layer. Application authentication is performed in communication layer on top of the transport layer. The Secure Channel has to be opened before client and server start exchanging routine information about the underlying process. Once the client and server are connected via a secure channel they possess a set of keys used to encrypt and/or sign future messages hence the term Secure Conversation. Clients can establish sessions using secure channel.

Profiles

Part 7 of the specifications contains profiles for clients and server products. It is mostly for vendors who manufacture OPC-UA products. The product may claim to adhere to certain profiles and a third party verification can be performed to certify the claim. The product may contain several profiles and each profile consists of ConformanceUnits which can be tested in a lab [9]. For example the Security is grouped into several ConformanceUnits such as security certificate validation, encryption type, signing, encryption etc.
OPC-UA Publish/Subscribe

OPC Foundation is now working on Publish/Subscribe (pub-sub in short) communication pattern for OPC-UA communication protocol. Which uses UDP multicast and AMQP broker to publish data. Subscribers can join the multicast group or register with the broker to retrieve data. The draft specification also defines a security architecture and is yet to be standardised. OPC-UA pub-sub with Time Sensitive Networking (TSN) could be used in time critical real-time control applications.

3.3 LWM2M for IIoT and Industry 4.0

OMA Lightweight M2M[13] is a protocol from the Open Mobile Alliance for M2M or IoT device management. Lightweight M2M enabler defines the application layer communication protocol between a LWM2M Server and a LWM2M Client which is frequently used with CoAP and located in a LWM2M Device. The OMA Lightweight M2M enabler includes device management and service enablement for LWM2M Devices. The devices targeted are mostly resource constrained. M2M applications can be built using this enabler. There are two frameworks providing the implementation of the protocol Leshan [14] and Wakaama [15]. OMA Lightweight M2M is designed to:

- Provide Device Management functionality over sensor or cellular networks
- Transfer service data from the network to devices
- Extend to meet the requirements of most any application

Initial release includes the following features:

- Simple Object based resource model
- Resource operations of creation/retrieval/update/deletion/configuration of attribute
- Resource observation/notification
- TLV/JSON/Plain Text/Opaque data format support
- UDP and SMS transport layer support
- DTLS based security
- Queue mode for NAT/Firewall environment
- Multiple LWM2M Server support
- Basic M2M functionalities: LWM2M Server, Access Control, Device, Connectivity, Firmware Update, Location, Connectivity Statistics

With OPC-UA and LWM2M/CoAP we can change the information exchange scenario in Industrial Automation shown in Figure 5 to much simpler scenarios as shown in Figure 9. The real-time control using OPC-UA is still a work in progress.
3.4 Real-Time Control and Information Exchange

So far we discussed various levels in the Industrial Automation, communication protocols and requirements. This section summarizes the scope of various protocols and their purpose in Industrial Automation.

3.4.1 Control

The control related tasks in Industrial Automation demand lower communication latency over Ethernet. Communication latency in different domains such as poultry, power, health, automotive, manufacturing etc. vary a lot. OPC-UA is the protocol for Industry 4.0 however current specifications do not meet the real-time control requirements over Ethernet. IoT protocols such as LWM2M/CoAP have not been designed for this purpose. Hence, fieldbus protocols will continue to exist due to their fast cycle times and timeliness till OPC-UA on TSN makes significant impact on real-time control applications.

3.4.2 Information Exchange

Information exchange or information reporting is a very important aspect especially in monitoring, traceability and analytical applications. OPC-UA can be used uniformly throughout all the levels of Industrial Automation. Figure 10 shows the scope of protocols in the Industrial Automation. IoT protocols such as LWM2M/CoAP which target cost effective applications can be extended or combined using the OPC-UA protocol or one of it’s variant. Omron is mainly interested in using the Sysmac-IPC for information exchange applications, hence the OPC-UA protocol is the best fit.
Figure 10: Summary: Communication Protocols
4 Information Modelling

This chapter explains the rationale behind of the information model as perceived by the OPC-UA standard and its specifications and presents basic concepts of the OPC-UA information model. Consider poultry processing as an example, process related information consists of status of machines, speed of motor, temperature of room, etc. and factory operations related information consists of number of birds being processed, yield per hour, etc. All such information must be represented in the OPC-UA server’s address space. A solution to our third research question from section 1.2, "how to model the information flowing in and out of Sysmac-IPC?" will be provided and finally the information models to be used with Sysmac-IPC will be discussed.

4.1 OPC-UA Information Models

An information model, as defined by the OPC-UA standard is ”an organizational framework that defines, characterizes and relates information resources of a given system or set of systems”. Figure 11 shows the OPC-UA hierarchical layered information model.

![Diagram of OPC-UA Hierarchical Information Model]

- The lowest layer in the information model hierarchy is the OPC-UA Data Model which defines Objects, Data Types, Variables, Classes etc. It is the base model on which all other information models are built.

- The layer above Data Model is the Address Space Model and has the information model of an empty server. Thus all OPC UA servers have it. The clients make use of OPC-UA services such as read/write to access the information.

- The next layer above the basic address space model contains some models for specific operations, such as Alarms and events, Historical data access. These are built on the
basic Address space model and these information models are specified by the OPC-UA standard. However, their basic form is inherited from classic OPC.

- The next layer above the standard defined information models contains different application domain specific models released by standard organisations. This layer helps in increasing the interoperability within the industries. This layer can also be built directly upon Address Space model.

- Finally, vendors can define their own information models to supplement the lower levels.

In general the information to be exchanged by a server can be modelled on top of basic OPC-UA Address Space Model without the need of OPC-UA’s standardised information models such as DA or Companion specifications such as PLCopen. However it is not recommendable in industrial automation where interoperability is desired. Considering the layered model, it is not necessarily important for the clients to have the knowledge of higher information models to access or discover data from them because the OPC-UA services are used to access/discover the information. The size and the complexity of the address spaces varies accordingly with the complexity of the servers. Multiple information models in a single server instance are supported by the use of NamespaceIndexes (Each model has a different NamespaceIndex). For example, OPC-UA server has all the information in its address space and the information can be grouped using NamespaceIndex. Further, the information can be organised into different views based users, for example: operator view, engineer view, maintenance view etc.

Though OPC-UA provides good potential for using information models however their use is not mandatory because the information can be easily presented by an OPC-UA server on top of empty server model such that the data can be accessed by clients. The resources can be represented in the OPC-UA server’s address space as per user’s convenience. For example, /Factory/Line1/Chicken/Weight, /Building/Floor1/Room1/Light1/Intensity etc., which is more transparent to the user in order to understand the poultry process instead of /PLC1/program1/lightingcontrol/input1.

**Advantages and principles of information modelling in OPC UA are:**

- Object-oriented techniques, such as type hierarchies and inheritance, allow clients to handle instances of a certain type while allowing them to ignore unnecessary information.

- Type information is exposed and accessed in the same way as instances.

- The network of nodes, being full-meshed, allows several hierarchies in the address space.

- Any information model can be exposed in OPC-UA, so that systems having an information model defined do not necessarily have to be mapped to another model. Also many information models can co-exist in a same OPC-UA server.

- All information modelling is on the server-side, so it is not necessary for clients to support it.

On the whole, the support for information models will enable new kinds of high level applications to be developed, for example: Traceability applications using OPC-UA Historical Data Access. An existing application’s information exchange model can be extended using the OPC-UA information modelling features, for example: LWM2M information model using
the OPC-UA’s information model. Chapter 6 will describe the OPC-UA information modelling process and apply it to obtain an OPC-UA implementation of the information model for LWM2M applications using suitable tools.

4.2 Representation of Information models

The address space of a OPC-UA server consists of nodes hierarchically organized into folders and objects. The data from a device in factory is represented as a node in OPC-UA server’s Address Space, for example: /Room1/temperatureSensor is one such node with attributes such as description, value, access rights, data type etc. If there are more such temperature sensors in each room, OPC-UA introduces a concept of type definition where the sensor can be type defined once and instantiated whenever required. For example, /Room1/temperatureSensor and /Room2/temperatureSensor can be the instances of a same type of temperatureSensor. The OPC-UA clients can also access all the type definitions in a OPC-UA server and learn about the contents of each instance. The notation for OPC-UA Information models is shown in Figure 12.

![Figure 12: The OPC-UA Information Model Notation](image)

Each of the Node Classes is represented by their unique notation, and also there are eight types of arrows to represent each of its references. These reference types are a small cross-section of the important predefined reference types.

**Symmetric References** are those References whose 'Symmetric' Attribute is set to true. It means that Reference is bidirectional when viewed either from the source Node or the target Node.

**Asymmetric References** are those References whose 'Symmetric' Attribute is set to false. This means that the Reference, when viewed from the target Node, is the inverse reference
when viewed from the source Node. The inverse relation may be named using the attribute ‘InverseName’.

Hierarchical References are used to define hierarchies in the Address Space, though it does not preclude loops. It is an abstract ReferenceType.

HasEventSource is used to build non-looping hierarchies that relay events. When a client listens to events from a Node in the hierarchy, it would also receive events from all Nodes beneath the listened Node in the hierarchy. It is a concrete ReferenceType, which means it may be used directly in the AddressSpace.

HasComponent is used for references which define the target Node of the reference to be a part of the source Node. It is a concrete ReferenceType and hence can be used directly in the AddressSpace.

HasProperty identifies Properties of a Node. It is a concrete ReferenceType.

HasTypeDefination binds Objects or Variables to their respective ObjectType or VariableTypes. It is a concrete ReferenceType.

HasSubType is used to express subtype relationships in the type hierarchy. For example, abstract ReferenceTypes can have concrete subtypes using this ReferenceType. It is a concrete ReferenceType.

4.3 Companion Specifications

There are many industrial organizations which have collaborated with OPC-UA to define information models apart from the base information model, these are called companion specifications. The OPC-UA can be extended to customize industrial information models which are to be represented in OPC UA server Address space. These companion specifications support the spirit of interoperability by unifying the representation of information in an application field.

4.4 Sysmac IPC: Information Modelling

Sysmac-IPC is a complex device with RTOS for real-time control and Windows OS for general purpose. Fieldbus devices are connected via Fieldbus networks such as EtherCat, EtherNet-IP, Profinet etc to PLC Engine in RTOS. Sysmac-IPC communicates to other PLCs (horizontal) and MES/ERP systems (Vertical) using OPC-UA protocol over Ethernet. The information model to be used with Sysmac IPC must include all the information, hence choice of standard or companion information model is a problem to be addressed. PLCopen, LWM2Mclient, ISA-95 and OPC-UA base information model share the same address space with different name spaces, views and user rights.
Figure 13: Sysmac IPC Information Model: Overview
Figure 13 shows the overview of the information modeling architecture proposed for the Sysmac-IPC. The OPC-UA server in Sysmac-IPC exchanges information horizontally with other devices in the plant or factory environment using the PLCopen information model for representing the data. Sysmac-IPC also exchanges information with MES/ERP/Cloud based systems vertically using ISA-95 information model. A single OPC-UA server instance or OPC-UA address space of the Sysmac-IPC contains basic OPC-UA information model, PLCopen information model, LWM2M client and ISA-95 information model as shown in Figure 14. Users at various levels can only access their share of information, hence user authentication and authorization over data is a necessity.

There are various software tools that are designed to build OPC-UA server address space using these information models, for example UAModeler [11], UA Model Designer [12], Status Enterprise [17] etc. B-SCADA’s Status Enterprise and Unified Automation’s UaModeler have been used to build Sysmac-IPC address space. Figure 14 shows the proposed information models to be contained in the Sysmac-IPC. OPC-UA base server model is defined by the standard, PLCopen and ISA-95 are the companion specifications to be used with PLC and MES systems respectively. A simple LWM2Mclient model is new contribution by this project, however OPC foundation and LWM2M group’s are working towards a standardized specification.

![Diagram of Sysmac-IPC Address Space](image)

Figure 14: Sysmac IPC Address Space: Example
As mentioned above the information from Sysmac-IPC is of interest to many users namely operators of Sysmac-IPC, other PLC users (horizontal) in the plant and ERP system users (vertical) outside the plant. So these information models are contained in a single server address space with different name space indices, views and workspace to achieve confidentiality and controlled information access.

For example, an ERP system user can only access ISA-95 information model from Sysmac-IPC and within the ISA-95 model there can be marketing view, maintenance view etc. Sysmac-IPC System administrator has access to all the information models and all the views. Other PLC system users within the plant can only access PLCopen information model and/or LWM2M information model. These are implemented using B-SCADA’s Status enterprise which has a concept of workspaces and user rights associated with workspaces.
5 Communication Architectures Using Sysmac IPC

This chapter discusses the solutions to our fourth research question from section 1.2 i.e. how to exchange data between the two operating systems in Sysmac-IPC securely and at the same time achieve high data throughput? What are the possible integration environments and communication architectures for the Sysmac-IPC? This chapter proposes architectural solutions for IIoT and Industry 4.0 scenarios using the Sysmac-IPC. The Sysmac-IPC on GPOS hosts a SCADA/MES/HMI application and RTOS has a PLC application. Hence it can accommodate two levels (Control and MES/SCADA) within a single device. The interaction of Sysmac-IPC with other PLC or SCADA devices in the plant is called Horizontal Communication and interaction with ERP or Fieldbus devices (Device Level) is called Vertical communication. Horizontal and Vertical communication architectures using different communication styles along with advantages and drawbacks are discussed in this chapter. Omron’s design principles behind Sysmac-IPC indicate that the existence of a RTOS for PLC functionality has to be hidden from users and they can only use GPOS for all purposes. Moreover, there is no physical connection between RTOS and other SCADA or PLCs for information exchange. Hence all the communication vertically upwards or horizontal must happen through GPOS and vertically downwards through RTOS. Inter-OS communication architectures between PLC (RTOS) and SCADA/HMI (GPOS) applications considering security, simplicity and throughput are also proposed in this chapter.

At first we discuss some important features of the Sysmac-IPC and then propose inter-OS communication architectures and a simple inter-OS communication protocol using shared memory and event system features provided by the hypervisor. Common scenarios in which the Sysmac-IPC can be used in an external environment are also discussed later in this chapter. The architectural solutions and protocols proposed consider only the information exchange aspect not the real-time control aspect of Industrial Automation. OPC-UA as a real-time control-communication protocol is yet to be accepted, however there are promising researches in progress about using OPC-UA over TSN which will be discussed in chapter 8.

5.1 Sysmac-IPC: Features

Sysmac-IPC contains two operating systems running on a hypervisor over Intel’s X86 Quad-core processor at 2.39 GHz. The hypervisor used in Sysmac-IPC for this project provides three services for inter-OS communication namely shared memory, events and virtual Ethernet as show in Figure 2.

- **Shared Memory**: Utilizing Shared Memory, application programs may pass data back and forth across otherwise impenetrable operating system boundaries.

  During run-time, applications under one or another of the supported operating systems will use functions from the Application Programming Interface (API) to read and write data in Shared Memory spaces. Functions are also provided to help protect and maintain data integrity.

- **Event System**: The hypervisor’s Event System makes it possible for operating systems running in the hypervisor environment to communicate with one another by sending
and receiving signals to and from user-created, named events. Using functions in the hypervisor’s Application Programming Interface, programs or tasks can create events, set a signal at an event, wait for a signal to arrive at an event and close events. Any event remains available until all application programs that opened it have finally closed it.

- **Virtual Ethernet**: When working with a traditional (external) data communications network, a common API (Application Programming Interface) provides a standardized means for application programs to communicate across network nodes. Such networks typically use Ethernet for this purpose. The hypervisor used in this project provides Virtual Network for the same functionality, but instead of transferring data across an external physical network, it moves the data through an area of main memory specifically designated for that purpose. Since data is transferred through memory, this is a correspondingly quick method for moving data from an application program under one operating system to another.

5.2 Inter-OS Communication

This section discusses the motivation for inter-OS communication and then possible solutions. An evaluation of each solution in terms of communication latency is presented in this section along with a note on security of each solution. The PLC on RTOS contains useful information about the factory functions, for example: speed of a motor, status of an equipment, temperature, pressure etc. High level applications on GPOS need this data from PLC either periodically or based on their value change or request based. Further, high level applications may also have to modify the PLC data. Hence, the need for inter-OS communication. The communication protocols designed for this purpose either use Virtual Ethernet Network or Shared Memory/Events or a combination of both in order to achieve a high throughput, secure communication without affecting the real-time control. Before discussing the solutions there are some important technical details to be noted. First, the data from device level is cyclically updated to the PLC on RTOS using EtherCAT communication interface every 500 µs, so the inter-OS communication latency has to be kept within 500 µs for the best possible performance. So if a monitoring application on GPOS requires a maximum of 2000 samples of an analog signal per second, the RTOS should be able to provide it, however current scenarios in monitoring applications use around 250 samples/second. Second, data changes made from GPOS to RTOS have to be authenticated to avoid controller malfunctioning. The information exchanged between the two applications should be confidential, meaning no other application (possible virus/malware) can access the information. Third, communication protocols used between the two operating systems need not be standardized because there are no interactions with external environment or other vendor products, however using standardized communication protocols and security measures over Virtual Ethernet saves development time but may not provide the best possible performance in terms of latency.
5.2.1 Secure communication over Virtual Ethernet

5.2.1.1 Using OPC-UA

OPC-UA server application is deployed on RTOS and the PLC data can be included in the Address Space of a OPC-UA server using PLCopen Information Model and a OPC-UA client can be access the PLC data using secure OPC-UA communication as explained in section 3.2. Figure 15 depicts this scenario. Data from the Device Level is updated periodically every 0.5 ms at the RTOS. The best case communication latency achieved using open62541’s OPC-UA communication stack \[18\] for an integer variable (4 Bytes) is 1.9 ms, which is good enough for some real-time applications but we target a latency of at least 0.5 ms. The target latency of 0.5 ms helps update data to GPOS before new data update arrives at the RTOS. Hence achieving a communication of latency of 1 ms between Device Level and SCADA/MES Level.

![Diagram](image)

Figure 15: Inter-OS: OPC-UA over Virtual Ethernet

5.2.1.2 Software Bus

This section briefly discusses Omron’s common communication infrastructure for communication between application programs running on RTOS and GPOS, called the Software Bus System.

Figure 16 shows components of the Software Bus System aka Software Bus. Detailed information about the Software Bus can not be shared in this document due to confidentiality concerns. However a brief explanation about the two important components is given below.

- **Software Bus RTOS Server**: It is a software application program that is responsible for authentication of applications from GPOS which try to communicate with PLC on RTOS and message encryption. In simple terms it is a server for PLC data for all applications on Windows.

- **Software Bus GPOS Client**: It is a software application program running on GPOS which is a client to Software Bus RTOS Server in order access PLC data. It provides an API to high level applications that are interested in PLC data.

The Software Bus System uses TCP/IP communication over Virtual Ethernet. Communication latency measurements performed on the first prototype implementation of the Software Bus System on Virtual Ethernet gives the best case latency of 9 ms.
It is evident from the above solutions that the best case communication latency achieved over Virtual Ethernet is in the range of 1 to 10 ms.
5.2.2 Publish-Subscribe Data Exchange

This section describes a simple means to exchange data between the two operating systems using shared memory and event system. The mechanism is explained below,

- GPOS has a Subscriber, which is an application program that can be a part of high level applications and subscribes to data from RTOS.

- RTOS has a Publisher, which is also an application program that can be a part of PLC program and publishes the subscribed data every 500µs.

- Subscriber can obtain the list of available data in terms of PLC variables from the Publisher.

- Subscriber can send a subscription message indicating the name/id of the variable along with subscription parameters such as update rate.

- Publisher, upon receiving a subscription request creates a publishing list based on update rates and periodically publishes the data.

- Publisher and Subscriber applications use shared memory hence synchronization becomes important. The hypervisor’s API provides lock-acquire and release functions where one application waits for a pre-determined period of time or throws a time-out error. Since awaiting release of a lock affects determinism, this mechanism is not recommended for hard deadlines.

- Hypervisor also has an event system, which can be used to signal an event. This protocol uses signaling mechanism to achieve synchronization with shared memory.

- If an application program opens a shared memory partition using Hypervisor’s API, no other application on the same operating system can access this partition till the first application program closes the shared memory partition.

- Pre-shared Signatures can be used with data variables, to authenticate Publisher & Subscriber.

Figure[17] shows an example scenario of the proposed mechanism.
Figure 17: Inter-OS: Publish-Subscribe over Shared Memory
Communication delay between message 6 and 12 in Figure 17 for each data update using different type of variables and data sizes were measured using cycle count method and findings are discussed below. Figures 18, 19 and 20 show the graphs of communication latency measured for 100 repetitions on integer, float and character type variables respectively with different data sizes.
Figure 19: Float Type Variable

Figure 20: Character Type Variable
Figures 21, 22 and 23 show the graphs of comparison of communication latency measured for 100 repetitions with integer, float and character type variables on 1KB, 2KB and 4KB data sizes respectively.
Figure 22: 2048 Bytes

Figure 23: 4096 Bytes
Table 1 summarizes the best case, the worst case and average latency. The proposed mechanism has overall best case latency of 31.619 µs, worst case latency of 502.550 µs and an average latency of 109.652 µs which achieves our target latency of 500 µs. The occurrence of the worst case latency is less than 1% considering all the repetitions across all the data types and sizes which is quite promising and is mainly due to unpredictability of the GPOS on Sysmac-IPC.

<table>
<thead>
<tr>
<th>Type</th>
<th>No. of Variables</th>
<th>Size in Bytes</th>
<th>Best Case (µs)</th>
<th>Worst Case (µs)</th>
<th>Average (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>1</td>
<td>4</td>
<td>31.619</td>
<td>78.503</td>
<td>35.464</td>
</tr>
<tr>
<td>Integer</td>
<td>256</td>
<td>1024</td>
<td>42.588</td>
<td>114.114</td>
<td>47.675</td>
</tr>
<tr>
<td>Integer</td>
<td>512</td>
<td>2048</td>
<td>81.891</td>
<td>322.555</td>
<td>112.444</td>
</tr>
<tr>
<td>Integer</td>
<td>1024</td>
<td>4096</td>
<td>99.327</td>
<td>502.550</td>
<td>146.153</td>
</tr>
<tr>
<td>Float</td>
<td>1</td>
<td>4</td>
<td>82.811</td>
<td>214.717</td>
<td>99.958</td>
</tr>
<tr>
<td>Float</td>
<td>256</td>
<td>1024</td>
<td>98.504</td>
<td>393.247</td>
<td>117.960</td>
</tr>
<tr>
<td>Float</td>
<td>512</td>
<td>2048</td>
<td>79.69426</td>
<td>471.434</td>
<td>153.514</td>
</tr>
<tr>
<td>Float</td>
<td>1024</td>
<td>4096</td>
<td>82.587</td>
<td>296.097</td>
<td>101.898</td>
</tr>
<tr>
<td>Char</td>
<td>1</td>
<td>1</td>
<td>97.224</td>
<td>497.778</td>
<td>123.002</td>
</tr>
<tr>
<td>Char</td>
<td>256</td>
<td>256</td>
<td>96.793</td>
<td>223.585</td>
<td>110.166</td>
</tr>
<tr>
<td>Char</td>
<td>512</td>
<td>512</td>
<td>76.391</td>
<td>335.431</td>
<td>305.903</td>
</tr>
<tr>
<td>Char</td>
<td>1024</td>
<td>1024</td>
<td>50.167</td>
<td>185.002</td>
<td>58.062</td>
</tr>
<tr>
<td>Char</td>
<td>2048</td>
<td>2048</td>
<td>39.4245</td>
<td>167.016</td>
<td>64.836</td>
</tr>
<tr>
<td>Char</td>
<td>4096</td>
<td>4096</td>
<td>47.5</td>
<td>170</td>
<td>58.1</td>
</tr>
</tbody>
</table>

Table 1: Best Case, Worst Case & Average Latency

All the results above are obtained using a single shared memory partition with event based synchronization. It is possible to achieve better performance in terms of communication latency using multiple shared memory partitions in parallel, where size of the partition and number of partitions are decided by throughput requirements. For instance publisher writes data updates consecutively to shared memory partitions while waiting for acknowledgement of the previous data update, eventually eliminating the need for synchronization since there is no possibility of memory overwriting. If the acknowledge is not received before a time-out for a memory partition, the memory partition can be re-used. This method is under prototyping stage. To conclude the discussion, the above mentioned publish subscribe communication mechanism using shared memory and event system achieves at least 10 times better performance compared to communication protocols such as OPC-UA over Virtual Ethernet.
Upon investigation, memory access synchronization overhead using the event system was found to be around $30 \, \mu s$. Another simple approach using semaphore to synchronize the access to shared memory gives the best performance in terms of update latency. The experiments were conducted for 1KB, 2KB, 4KB and 8KB of character, integer and float type data. Figure 24 shows the graph depicting results obtained for 8KB of data in order to prove that the results are better than previous methods even with higher data size.

![Latency Measurements of 8KB Data Using Semaphore for Synchronization](image)

Figure 24: Inter-OS: Publish-Subscribe over Shared Memory Using Semaphore for Synchronization

The graph shows an average latency of $35 \, \mu s$ for character type, $11 \, \mu s$ for integer type and $9 \, \mu s$ for float type. Character type has higher average because the number of memory accesses is 4 times that of integer or float. The above graph also shows the best case latency of $7 \, \mu s$ and the worst case latency of $126 \, \mu s$ which clearly indicated that this approach gives the best performance in terms of latency.

Based on the results obtained in this section, Omron is implementing the Software Bus system on shared memory and events. A patent was filed during the project for a communication mechanism which makes use of Virtual Ethernet, Shared Memory and Events to achieve high throughput communication for OPC-UA between RTOS and GPOS.
5.3 External Communication

Figure 25 shows an example of an external integration environment of the Sysmac-IPC. It connects to: fieldbus devices over Ethernet networks via RTOS using fieldbus communication protocols vertically downwards, other SCADA/MES/HMI devices in the plant/factory network using OPC-UA over Ethernet LAN horizontally via GPOS and ERP/Cloud based applications vertically upwards through the Internet using OPC-UA/HTTP via GPOS.

![Figure 25: Sysmac IPC : Horizontal and Vertical Integration Environment](image)

Using the OPC-UA communication protocol with the Sysmac-IPC has many advantages as discussed in chapter 3, the impact of using different communication styles such as Client/Server and Publish-Subscribe will be discussed in this section.

5.3.1 Client-Server

- **Features**: An external device can securely access information contained in the Sysmac-IPC by using OPC-UA client over Ethernet through a secure OPC-UA TCP session. The number of clients can be limited by server side based on its ability to service the clients simultaneously.

- **Performance**: Communication latency between a single client and a server in a LAN is in the range of 2 ms to 5 ms. Memory and CPU utilization are dependent on the size of the address space and number of simultaneous client sessions. Sysmac-IPC has 8GB RAM, 500 GB hard disk and 1Gbps LAN port, hence it can handle a large number
of clients easily. However, for low resource devices at Device Level, it is a matter of concern if they use OPC-UA communication.

- **Scalability**: OPC-UA doesn’t place any limitation on the size of its Address Space, it is only limited by memory availability. Number of client sessions supported mainly depends on available network bandwidth.

### 5.3.2 Publish-Subscribe (pub-sub)

OPC-UA pub-sub is not implemented on Sysmac-IPC, since standardization is still in progress and no commercial or open source communication stacks are available for experimentation. In OPC-UA client-server protocol specification, there is a concept of subscription and monitored items, which can be used to publish data, but OPC-UA pub-sub specification has mappings to UDP and AMQP middleware.
6 LWM2M/OPC-UA Protocol Translator

6.1 Introduction

This chapter discusses the solutions to the fifth research question from section 1.2 i.e. How to incorporate the advantages of IoT protocols such as low resource requirements, within the Industry 4.0 applications? Many organisations and international standardisation committees are constantly working on communication protocols for industrial control, monitoring and maintenance applications. OPC Foundation is one such organisation which has been working in this area for decades. The latest communication protocol by OPC-Foundation i.e OPC-UA has already been widely accepted in Industrial Automation. However industrial communication protocols such as OPC-UA sometimes seem a bit expensive for really small embedded devices. On the other hand OMA LWM2M standardisation committee is dedicated to design efficient communication and device management protocols for resource constrained embedded devices.

Consider a typical scenario where the temperature sensors or weighing devices have LWM2M clients and where the OPC-UA protocol is used to report the factory information to MES/ERP. Such a scenario requires an extension LWM2M services through OPC-UA services. Consider another example, in a poultry processing factory the weight of the birds are continuously measured and reported to MES/ERP systems for bird data analysis or yield improvement or local real-time monitoring as discussed in chapter 2 which can benefit from such protocol extension or translators. Poultry is a high volume – low margin business with short Return of Interest (about 12-18 months) and a fraction of cent more yield per chicken/bird means big profit gain, so any tiny improvement counts. In the current scenario the information about the weight of the birds measured travels across various levels of Industrial Automation Hierarchy to reach such end applications. Using LWM2M/CoAP for this purpose requires one to open UDP ports on firewalls at the plant boundary which is not a recommended cybersecurity practice and also using OPC-UA provides additional advantage as explained in the subsequent sections.

In this project, we combine LWM2M and OPC-UA protocols using an application level translator. OPC-UA and LWM2M are both secure communication protocols, especially OPC-UA scales from sensors to huge data servers. LWM2M is gaining popularity and widespread acceptance in resource constrained environments. Extending LWM2M-device’s information using the OPC-UA information models such as Historical Data Access, Alarms & events and Data access can serve interesting applications such as chicken weight monitoring on multiple lines in a poultry processing farm as discussed in chapter 2.

Section 6.2 contains a detailed comparison between the two protocols. Implementation details of the translator are discussed in section 6.3. The evaluation of the implementation is presented in section 6.4.
6.2 Comparison between OPC-UA and LWM2M

6.2.1 LWM2M

LWM2M is an enabler of M2M communication for resource constrained devices. The protocol is designed to be used for information & device-management of resource constrained devices. LWM2M is a client-server protocol, a LWM2M-Client hosts the actual resources and a LWM2M-Server can perform read/write/execute operations on clients (which contradicts the general meaning of client and server but that’s how the standard defines it).

LWM2M defines four interfaces with relevant operations between the client and the server, namely

- Client Registration: Register, Update, De-register
- Bootstrap: Bootstrap request, Write, Delete and Bootstrap Finish
- Device Management and Service Enablement: Read, Create, Delete, Write, Execute, Write Attributes and Discover
- Information Reporting: Observe, Cancel Observation and Notify

These operations mentioned above are classified into uplink(client to server) and downlink(Server to Client), which are shown in table 2.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Direction</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bootstrap</td>
<td>Uplink</td>
<td>Request Bootstrap</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>Downlink</td>
<td>Write, Delete</td>
</tr>
<tr>
<td>Client Registration</td>
<td>Uplink</td>
<td>Register, Update, De-register</td>
</tr>
<tr>
<td>Device Management and Service Enablement</td>
<td>Downlink</td>
<td>Create, Read, Write, Delete, Execute, Write Attributes, Discover</td>
</tr>
<tr>
<td>Information Reporting</td>
<td>Downlink</td>
<td>Observe, Cancel Observation</td>
</tr>
<tr>
<td>Information Reporting</td>
<td>Uplink</td>
<td>Notify</td>
</tr>
</tbody>
</table>

Table 2: Relationship of operations and interfaces

Figure 26 and 27 depict the use of the operations between LWM2M client and server.
Figure 26: LWM2M: Bootstrap and Client Registration

Figure 27: LWM2M: Device Management & Service Enablement and Information Reporting
LWM2M Object Model:

A LWM2M-system assumes that every device has a set of parameters. Each of these parameters is mapped to an aspect of its functionality. Some may be Read-write that needs configuration and control and some may only be Read/Only used for observing status information.

An example might be a simple dimmer that controls the brightness of the light. The dimmer may have a variable to switch it on or off and another to set the brightness level.

LWM2M assigns a unique resource ID for each of these parameters and groups them in to something called an Object identified by a unique Object ID. For each actual device present, an instance ID is allocated which contains the actual resource values. The instance ID management is done by the LWM2M Client and typically starts from 0. There may be any number of instances based on the actual number of devices.

The object model and resource ID definitions may be complex depending on the complexity of the device it represents.

LWM2M client hosts the objects that contain resources to LWM2M servers as shown in Figure 28. Resource represents the element of physical or digital significance for example a pressure sensor or a counter variable. There could be many such variables or sensors which can be represented by the same resource hence the resource can be instantiated or another resource can be created. In terms of object-instance/resource philosophy, it resembles the CIP standard from ODVA that is employed in DeviceNet & Ethernet/IP and other protocols being used in industrial automation.

Figure 28: LWM2M: Object/Resource Model
6.2.2 OPC-UA

In this section we present how ReSTful is OPC-UA as discussed by Sten Gruner et al. in [4] with respect to four important properties of ReSTful protocols namely Addressability, Uniform Interface, Connectedness and Statelessness.

- **Addressability**: As explained in concepts such as information modelling and address space of OPC-UA in chapter 4 [4], nodes can be found either via their unique ID or, since every node must have a hierarchical reference to a parent up to the root node, by walking a BrowsePath on the hierarchical references. Paths can be encoded in simple human-readable form, such as `/factory/slaughter/line1/`.

- **Uniform Interface**: The services defined in OPC UA are fixed and cannot be changed by the user. In that sense, there is a uniform interface that is common to all applications. But its capabilities go well beyond the standardized access to resource representations. For example, the Call service can be used for remote procedure calls with arbitrary arguments and return values that are defined in the corresponding MethodNode. But knowing the prototype of a function is no substitute for knowing its semantics and side-effects (also on the physical world). So the possibility to fully introspect an OPC UA information model does not by itself remove the need for out-of-band interface information transfer.

- **Connectedness**: It is a core concept of OPC UA to have nodes referring to each other with their unique identifier. NodeIdentifiers may also point to a remote server and are then called ExpandedNodeId.

- **Statelessness**: OPC-UA is a Service Oriented M2M protocol, with standardised fixed number of services (* indicates inherently stateless services). The services are grouped into service sets, namely

  - Discovery Service Set: FindServers*, GetEndpoints*, RegisterServer*
  - Secure Channel Service Set: OpenSecureChannel, CloseSecureChannel
  - Session Service Set: CreateSession, ActivateSession, CloseSession, Cancel
  - Node Managements Service Set: AddNodes*, AddReferences*, DeleteNodes*, DeleteReferences*
  - View Service Set: Browse* BrowseNext, TranslateBrowsePathsTo-NodeIds*, RegisterNodes, UnregisterNodes
  - Query Service Set: QueryFirst*, QueryNext
  - Attribute Service Set: Read*, HistoryRead, Write*, HistoryUpdate
  - Method Service Set: Call*
  - Monitored Item Service Set: CreateMonitoredItems, ModifyMonitoredItems, Set-MonitoringMode, SetTriggering, DeleteMonitoredItems
  - Subscription Service Set: CreateSubscription, ModifySubscription, SetPublishing-Mode, Publish, Republish, TransferSubscriptions, DeleteSubscriptions

52
In general, OPC UA cannot be considered stateless. First, the protocol defines compulsory *SecureChannels* and Sessions that expire without regular renewal. Second, many of the OPC UA services are inherently stateful. For example, the Browse service can be requested with a maximum number of returned items. The server then returns a *ContinuationPoint* token, used to request the remaining items. Third, push-notification by the server (*MonitoredItems* and *Subscriptions*) obviously cannot be implemented without storing state.

Table 3 gives the comparison of important properties between the two protocols.

<table>
<thead>
<tr>
<th>Property</th>
<th>OPC-UA</th>
<th>LWM2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol State</td>
<td>Stateful</td>
<td>Stateless</td>
</tr>
<tr>
<td>Carrier Protocol</td>
<td>TCP</td>
<td>UDP and SMS</td>
</tr>
<tr>
<td>ReST</td>
<td>Inherently not ReSTful but some services are stateless</td>
<td>ReSTful</td>
</tr>
<tr>
<td>Communication Style</td>
<td>Client-Server, Publish-Subscribe(specification in draft-stage)</td>
<td>Client-Server</td>
</tr>
<tr>
<td>Protocol</td>
<td>UA-TCP Binary and SOAP(Web Service)</td>
<td>CoAP</td>
</tr>
</tbody>
</table>

Table 3: Comparison Between OPC-UA and LWM2M

Hence, the OPC-UA protocol is inherently not ReSTful such as LWM2M, but some services are stateless, as discussed above. The set of downlink operations in LWM2M protocol can been seen as a subset of OPC-UA services. Hence, implementation of a translator from OPC-UA server to LWM2M server(LWM2M server can read/write etc. from LWM2M client, which is not the general notion of client-server) makes sense and can serve new range of applications and industry needs.

For, example: A temperature sensor hosting a LWM2M client connects to a LWM2M server located in a PC, LWM2M server can access information or get notified about data changes by the temperature sensor. The PC renders data to various application systems such as SCADA, Monitoring etc. using the OPC-UA protocol. Hosting LWM2M client devices in the OPC-UA server’s Address Space as shown in Figure 14 and translating the read/write/subscription service request from OPC-UA clients to LWM2M services is the idea behind this translator application. By doing so, the high level applications can benefit from OPC-UA information models such as Historical Data Access, Alarms & events etc. Figure 29 shows an example of historical trend view of a simulated random signal in the OPC-UA client, where the random signal is reported periodically by the OPC-UA server. Figure 33 shows real-time monitoring of temperature.
6.3 Implementation

To prototype the translation of OPC-UA services to LWM2M services as shown in Figure 32, a developmental set-up is made. This developmental set-up includes a RaspberryPi-2 Model B development board with Sense-HAT. Sense-HAT board contains a humidity and temperature sensor. LWM2M server, OPC-UA server and translator are deployed on Ubuntu Linux running on Windows host. OPC-UA client is deployed in another PC running Windows. A LWM2M client is deployed on RaspberryPi board which hosts the temperature sensor as an LWM2M object. The set-up is shown in the Figure 30.

The important step in proving the concept of this translator is translating read/write/subscription/method services. To begin with the OPC-UA server is developed with LWM2M client or its resource such as temperature as an object in the OPC-UA server's Address Space. As explained in chapter 4, the information modelling in OPC-UA is very flexible, a temperature sensor can be represented as a resource of some OPC-UA object or complete LWM2M client's resources can be included as shown in Figure 31.

Explaining OPC-UA server development process is out of scope of this project. An information model based on the OPC-UA protocol is built in this project, a simple tutorial which explains the server development process by Unified Automation can be found here [7]. Important considerations for the translator are listed below.

- While defining OPC-UA Object Types and Data Types corresponding to LWM2M
Object and Resources it is important to know that OPC-UA supports a large set of Data Types and in fact the LWM2M Data Types are a subset of OPC-UA. For example: A LWM2M Object can be treated as OPC-UA object and a LWM2M objet’s resource can be treated as a OPC-UA variable with attributes such as description, value etc.

- Method Call: LWM2M’s Exec resource can be included in OPC-UA Address Space as OPC-UA Method. An important difference between the Method or Exec services is that OPC-UA Methods return output parameters while LWM2M Exec service only returns success or failure of the operation. Another important aspect to be noticed is that, the LWM2M exec call’s arguments are passed as ”plain text” and data types are not associated with the parameters while OPC-UA is much more advanced. Since we are trying to extend LWM2M using OPC-UA protocol this is not an issue.

Steps involved in the development:

- Creating LWM2MClientType object using OPC-UA base server model and object types using UaModeler as shown in Figure 31.
- Extending OPC-UA’s read, write, subscriptions, method call and node management services using LWM2M’s read, write, observer/notify, execute and create service calls along with error/success codes.
- Security and end point configuration.
- Including other models such as PLCopen, ISA-95 if necessary.
Figure 31: OPC-UA Information Model for LWM2M Devices
Use Historical Data Access, Alarms and Events on LWM2M model based on application needs.

### 6.3.1 Translating OPC-UA services to LWM2M operations

This section explains the implementation of LWM2M read service call from a OPC-UA client to a OPC-UA server translated into a LWM2M read service call to a remote LWM2M client, through a LWM2M server. An example of interactions between the OPC-UA client and the LWM2M Client via translator are shown in Figure 32. We focus mainly on translating the operations of the Device Management and Service Enablement and Information Reporting interfaces of the LWM2M protocol.

![Figure 32: Example Interactions between OPC-UA client and LWM2M Client](image)

Wakaama, an open source C language implementation of LWM2M is used for both LWM2M client and server. Evaluation version of ANSI C SDK from Unified Automation which can be found here[10] is used to develop the OPC-UA server.

OPC-UA read and write services are translated to LWM2M operations using message passing (named pipes), and shared memory is used for information reporting operations (observe, notify). Extending LWM2M ”Observe/Notify” operations using OPC-UA ”Subscriptions” is shown in the code snippet below. When a OPC-UA client creates subscription, the OPC-UA server translates it to ”Observe” operation using named pipes as shown below.
// opc-ua create subscription callback function
if (pUserData != OpcUa_Null)
{
    switch (pUserData->Type)
    {
    case UserDataTemperature:
        
            int fd;
            char * myfifo = "\tmp/funwrite"; //named pipe
            fd = open (myfifo, O_WRONLY);
            // LWM2M client ID : 0,
            // LWM2M object ID : 1024,
            // Instance ID : 10,
            // Resource ID : 3,
            write (fd, "observe 0 /1024/10/3", sizeof("observe 0
            /1024/10/3"));
            close (fd);
            break;
        
    case UserDataMachineSwitch:
        
            break;
    default:
    
            break;
    
    }
}

On LWM2M server, the observe request from OPC-UA is forwarded to LWM2M client as shown below.

char * myfifo1 = "/tmp/funwrite"; //named pipe
char buf1[MAX_BUF];
mkfifo (myfifo1, 0666);
while (0 == g_quit)
{
    int fd=-1;
    fd = open (myfifo, O_RDWR,O_NONBLOCK);

    int fd1=-1;
    fd1 = open (myfifo1, O_RDWR,O_NONBLOCK);

    FD_ZERO(&readfds);
    FD_SET (sock, &readfds);
    FD_SET (STDIN_FILENO, &readfds);
    FD_SET (fd, &readfds);
    FD_SET (fd1, &readfds);

    tv.tv_sec = 60;
tv.tv_usec = 0;
result = lwm2m_step(lwm2mH, &(tv.tv_sec));
result = select(FD_SETSIZE, &readfds, 0, 0, &tv);

if ( result < 0 )
{
    // error
}
else if ( result > 0 )
{
    uint8_t buffer[MAX_PACKET_SIZE];
    int numBytes;
    .
    .
    // when observe command is written to the named pipe
    if (FD_ISSET(fd1, &readfds)) // OBSERVE OPERATION FROM OPC_UA
    {
        numBytes = read(fd1, buf1, MAX_BUF - 1);
        close(fd1);

        if (numBytes > 1)
        {
            buf1[numBytes] = 0;

            handle_command(commands, (char*)buf1); // FORWARDS OBSERVE TO LWM2M CLIENT
        }
    }
    else
    {
        // close pipes
    }
}

Client notifications are in "Plain Text", a shared memory variable is created between OPC-UA and LWM2M servers and the code snippet below shows conversion of data in "Plain Text" notifications from clients to float data using standard C library functions.

void output_new_notify(FILE * stream,
    uint8_t * buffer,
    int length,
    int indent)
{
    int i;
    int j;

    char tmpbuf[256]=" ";
    char buf[32]=" ";

    int shmid;
    key_t key;
    float *shm; // shared memory variable
    key = 5678;
    if ((shmid = shmget(key, sizeof(shm[0]), IPC_CREAT | 0666)) < 0) {
        perror("shmget");
        exit(1);
    }
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}

if (length == 0) printf(stream, "\n");

i = 0;
while (i < length) // process notification
{
    // extract value from data buffer
    strncpy(buf, tmpbuf, i+j);
    // converting plain text to float or double and assign it to shared variable
    shm = atof(buf);
}

The temperature of the sensor transmitted using LWM2M Client to LWM2M Server and extended using the OPC-UA’s Historical Data Access information model and monitored using a OPC-UA Client is as shown in Figure 33.

Figure 33: Temperature Monitoring Using the LWM2M-OPC-UA Translator
6.4 Evaluation

As mentioned before the main focus of the project is to prove the concept rather than producing a commercial software. Measurement of communication latency with the set-up shown in Figure 30 have been performed over LAN and the best case communication latency of value change notifications from LWM2M client in Raspberry Pi to LWM2M server in Ubuntu PC obtained is 5 ms. The worst case latency mainly depends on the number of simultaneous clients and transmission distance for a given physical medium such as Ethernet or Wi-Fi. End-to-end communication latency of value change notification depends on OPC-UA MonitoredItems settings such as sampling period or event system settings. The best case end to end latency is 7.5 ms from LWM2M client to OPC-UA client over 100 Mbps LAN. The translator can be extended to include multiple LWM2M clients, where the communication latency depends on the number of simultaneous value change notifications from different clients. The OPC-UA Historical Data Server logs the value change notifications in a persistent storage with time stamps and high level applications can request these samples of data till a period of time, which is decided by the application needs and after which the data samples in the server will be erased to make space for new samples.
7 Results and Recommendations

This chapter highlights the contribution of the project in order to prepare the Sysmac-IPC for IIoT and Industry 4.0 environments and future recommendations.

7.1 Results

- **Information Modelling:** The project makes use of the OPC-UA communication protocol for real-time information reporting for applications which have latency requirements up to a few milliseconds over Ethernet. The project delivered an information model based on OPC-UA communication protocol for the Sysmac-IPC considering its Horizontal and Vertical integration environment, which is accepted by Omron. The solution is discussed in section 4. The information model includes PLC data, MES data and other user-defined data along with it. The information is segregated in workspaces to manage controlled access to intended users or user groups, which achieves accountability of individual users of the system. The project made use of BSCADA’s status enterprise as GUI based information modelling tool and also has a web gateway which makes it possible to access the information from a smartphone.

- **Inter-OS Communication Architecture:** Measurement of communication latency of standard protocols such as OPC-UA and LWM2M was performed between RTOS and GPOS over Virtual Ethernet. A simple publish-subscribe mechanism is implemented to update PLC data to high level applications running on GPOS within communication latency of 500 μs. In industrial automation, the communication delay between devices in Control Level and MES/SCADA level is in the order of 10 ms to 1 s. This is because PLCs are implemented on hardware or dedicated embedded software hence the information has to be exchanged using a standardized communication protocol over Ethernet. With Sysmac-IPC the Control and MES Levels can exist on a single device which makes it possible to achieve high performance with respect to communication latency.

- **OPC-UA/LWM2M Translator/Extender:** Two protocols from different domains have been combined in order to achieve cost-effective high level applications. A temperature monitoring application with a best case end-to-end latency of 5 ms using Raspberry-pi SenseHat is implemented.

- **Concerns Addressed:** The project addresses important concerns and requirements discussed in the case study such as ease of integration, secure communication, monitoring applications, control etc. One Sysmac-IPC device can be used for control, SCADA, Traceability, Monitoring and other high level applications.

7.2 Recommendations

- **Real Time Control on OPC-UA:** At present the Sysmac-IPC has EtherCAT interface on the controller side (RTOS), research OPC-UA Pub-Sub protocol on TSN sounds promising and has to be investigated.
• **Inter-OS Communication**: Inter-OS communication mechanism explained in this project can be extended using multiple memory partitions which can improve the performance in terms of communication latency. Further, OPC-UA services can be mapped on shared memory and events instead of TCP/IP over Ethernet which achieves better performance and uniformity of protocols across all the levels.

• **Translator**: As mentioned earlier extending LWM2M using OPC-UA is implemented for proof of concept, further advancements and experiments can be conducted in order to commercialize it as a software product. Experiments such as Scalability (No. of Clients) vs Latency, Scalability (No. of Clients) vs Bandwidth utilization etc. have to be performed.
8 Conclusions

Figure 34: Sysmac-IPC Communication in IIoT & Industry 4.0 Environments

The project has prepared Omron’s Sysmac-IPC for data exchange in Industry 4.0 and IIoT environments. The OPC-UA server on GPOS acts as a single access point for communication of Control Level Data and provides authentication, authorization and confidentiality of information. Control Level is hidden to users in the form of RTOS in Symac-IPC. Users can benefit from the real-time control (RTOS) and user friendliness of GPOS (Windows) in one single device. The possibility of having controller and a higher level application from MES/SCADA Level on one single device improves communication latency at least by 10 times. A standardized information modelling is followed using OPC-UA communication protocol for interoperability. Finally, we combined protocols from IoT (LWM2M) and Industry 4.0 (OPC-UA) domains to explore new possibilities and useful applications which are cost effective. The Sysmac-IPC now has a OPC-UA Client interface on RTOS to access data from all lower level devices, publish subscribe mechanism internally between the two OS and a OPC-UA server on GPOS for all external communications as shown in figure 34. This project answered all the questions in 1.2 and satisfied Omron’s requirements.
References


http://www.commsvr.com/Products/OPCUA/UAModelDesigner.aspx

http://openmobilealliance.org/about-oma/work-program/m2m-enablers/

https://github.com/eclipse/leshan

https://github.com/eclipse/wakaama


http://www.scada.com/Software/StatusEnterpriseEdition

http://open62541.org/