工業 4.0 通訊技術於運動控制系統之應用與趨勢

Trends and Application of Motion Control System with Industry 4.0 Communications

陸品丞、李宜靜、李宜玲

工研院機械所 控制核心技術組 機電控制整合部　副研究員

摘要：工業技術已發展至第四次工業革命，近年來德國也提出工業 4.0 的概念，透過將數位化與虛擬化導入生產系統中，以打造智慧工廠為訴求，結合製造業與資訊技術，提出網路實體系統 (Cyber-Physical System, CPS) 的解決方案。國內外大廠也紛紛投入設備自動化、系統虛實化及工廠智慧化的發展。而互聯網在工業自動化領域中也逐漸占有重要的地位，但因為缺乏一個統一的通訊標準，使得資訊整合成為工業 4.0 的一大瓶頸。因此本文將介紹 OPC UA (Open Platform Communications Unified Architecture) 及 MQTT(Message Queuing Telemetry Transport) 兩種目前最新的通訊機制，並與機械手臂之運動控制系統進行整合。

Abstract：The fourth industrial revolution has been taking place in recent years. Germany has proposed the concept of Industry 4.0 by importing digitization and visualization into manufacturing systems with information technology, and thus the solution of Cyber-Physical System (CPS) has been promoted to create a smart factory. Domestic and foreign enterprises have invested in equipment automation, CPS system and the development of factory intelligence. Although the Internet becomes more important in industrial automation, the lack of unified communication standard makes the information integrating an obstacle for Industry 4.0. Therefore, the newest communication protocol OPC UA and MQTT will be introduced in this article and these protocols will be combined with an robotic arm and motion control system.

關鍵詞：開放平台通訊統一架構、訊息佇列遙測傳輸、工業物聯網　、運動控制

Keywords：OPC UA, MQTT, IIoT, Motion control

前言

順應工業 4.0 的潮流，國內各大廠皆積極推動工廠環境與作業模式的轉型，而虛實整合也成為了核心發展技術，透過整合工廠內各種設備的資訊，經由網路實體系統協助分析、處理與決策，提升產線設備的工作效率、故障排除及數據管理的能力。然而為了實現廠區內的工業自動化，未來勢必將導入更多物聯網設備來協助機台進行串聯與資訊蒐集，因此如何針對不同的應用情境選擇適合的通訊協定，對於系統的執行效率與擴展性也成為了關鍵因素。目前在物聯網的應用中，輕量級的通訊協定如 MQTT、CoAP(Constrained Application Protocol)、XMPP(Extensible Messaging and Presence Protoco)、DDS(Data Distribution Service)、AMQP(Advanced Message Queuing Protocol) 等，皆有利於使用在硬體受限的設備或是不穩定的網路環境中，這些輕量級的通訊協定也讓物聯網的應用可以觸及更多的應用層面。

另一方面，透過結合物聯網與工業自動化技術，產線設備除了擁有基本網路連線和資料傳輸的功能之外，在具備高運算能力的設備中，還可

以進一步整合來自其它設備 (如感測器) 的資訊，透過生產參數進行自動分析及計算，提供後續維護策略和預防性維護等功能。由於目前工業控制器的種類繁多，配合 HMI 或 SCADA 等工業軟體的應用，供應商需要各針對不同機台類型進行軟體及資料整合，而導致效率低與擴展不易的問題。此外在製程自動化的領域中，當製程數據從分散式控制系統 (Distributed Control System, DCS) 傳送到上層的製造執行系統與企業資源管理系統時，必須按照各個供應商的機台種類開發特定的通訊介面，使得企業在進行資訊垂直整合時遇到瓶頸，因此需要透過一套標準化的通訊機制來實現不同供應商和設備之間的資料通訊。

本文主要分為三個章節，第二章節與第三章節介紹 OPC UA 與 MQTT 的通訊技術與目前產業的應用情況，第四章節為整合 OPC UA 與 MQTT 的應用實例，透過結合工研院機械所開發之運動控制函式庫 (Motion Control Command Library, MCCL）和運動控制軸卡，應用在機械手臂控制系統中。

OPC UA

1. OPC UA 背景與介紹

OPC UA 的發展可以追溯於 1987 年，Microsoft 推出了動態資料交換 (Dynamic Data Exchange, DDE) 通訊機制，DDE 藉由共享記憶體的方式，讓不同的應用程式間可以動態進行資料交換。1990 年，Microsoft 將 DDE 改良後發佈了物件連結與嵌入技術 (Object Linking and Embedding, OLE)，透過複合檔的概念，使單一檔案能夠儲存如文字、圖形、視訊與聲音等多重資料格式，但由於 OLE 的架構過於複雜，以至於鮮少廠商使用此技術進行內部資訊的整合，所以之後便發展出 COM(Component Object Model)/DCOM(Distributed COM) 技術。由於當時工業自動化領域還未有一個統一的通訊標準，軟體廠商需要針對特定機台與專屬的通訊協定製作特殊的驅動程式，使得機台資料的擁有權成為工業市場主要的爭奪點，因此在 1995 年時，四家自動化公司組成一個專案小組，依據 OLE 與 COM/DCOM 技術制定一個用於工業自動化資料存取 (Data access, DA) 的通訊規範，這也成為了早期 OPC(OLE for process control) 的基礎，專案小組於隔年發佈了 OPC 1.0 版並成立 OPC 基金會。

OPC 通訊規範的核心為解決互通性 (Interoperability) 和標準化 (Standardization) 的問題，透過 Client/Server 的方式來達成資料傳輸，所以在整合應用時只要設備支援 OPC 規範即可輕易地交換資訊，大幅減少硬體設計者、軟體合作廠商、SCADA 及 HMI 廠商在建立與整合設備管理介面上所需要的時間成本。由於 OPC 是第一個能將自動化數據與資訊系統進行整合的通訊規範，並支援不同類型的設備及控制器，因此 OPC 在設備製造商和自動化軟體供應商中受到廣泛支持，經過多年的發展後，OPC 額外添加許多機制以增強 DA 規範，並將 OPC 重新定義為開放平臺通訊 (Open Platform Communication)。

2006 年，OPC 基金會將 OPC 的功能整合成一個可擴展的服務導向 (Service-Oriented) 架構，即為 OPC UA，OPC UA 涵蓋了 OPC 資料存取規範 (OPC DA)、歷史資料規範 (OPC HA)、報警與事件規範 (OPC A&E) 和安全協議 (OPC Security) 等功能，並在其基礎上進行功能的擴展，OPC UA 的基本架構如**圖 1** 所示。有別於傳統的 OPC，OPC UA 有以下 5 項特點：

(1) 資訊瀏覽統一性：OPC UA 有效地將現有的 OPC 規範（DA、A&E、HA、命令、複雜資料型態和物件類型）進行整合，並提供了一致、完整的定址空間 (Address Space) 和服務模型，解決過去相同系統的資訊不能以統一方式讀取的問題。

(2) 擴充通訊性能：OPC UA 規範可以通過任何通訊埠 (Port) 進行資料的傳輸，解決傳統 OPC 因資訊系統防火牆導致資料被阻擋的問題。訊息的編碼格式可以是 XML 或二進位格式，也可使用多種通訊協定進行傳輸，例如 TCP、

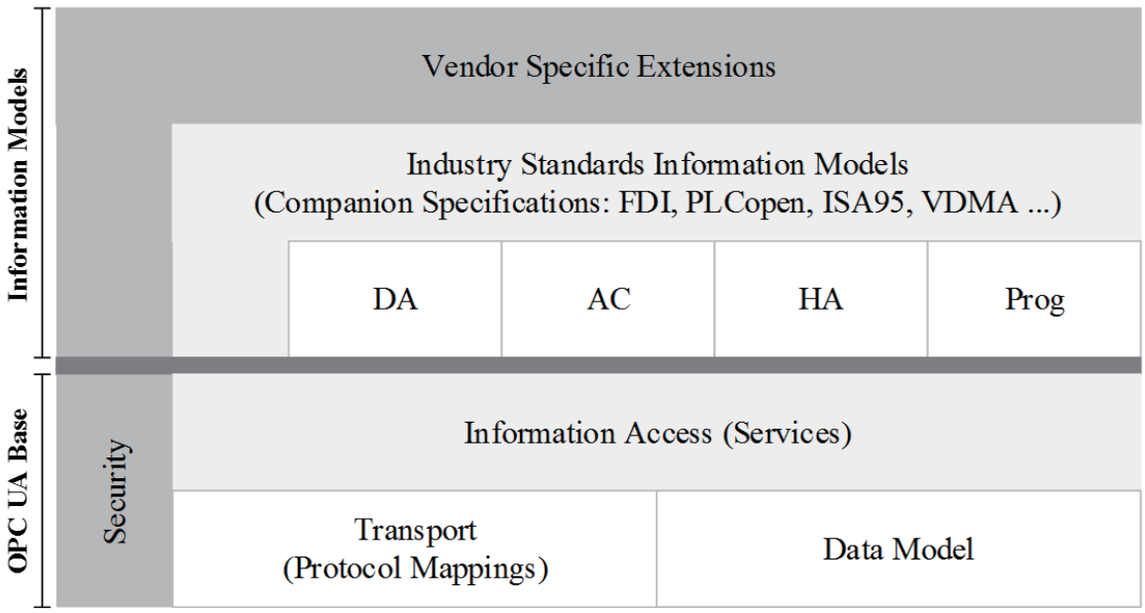

圖 1 OPC UA 基本架構

HTTPS、AMQP 和 MQTT 等。

(3) 可靠性：OPC UA 的設計高度提升了系統可靠性，新增了錯誤警報和自動校正等功能，使得符合 OPC UA 規範的軟體產品可以自如地處理通訊錯誤。OPC UA 的標準模型也使得來自不同廠商的軟體應用可以相容，提升資訊整合的便利性。

(4) 標準安全模型：OPC UA 規範明確定義了標準安全模型，每個 OPC UA 的應用都必須執行 OPC UA 安全協議。OPC-UA 採用 X.509 憑證、Kerberos 協定和使用者密碼進行身份驗證，也提供數位簽章和加密傳輸以及級別驗證等認證機制。

(5) 跨平臺：OPC UA 的應用程式不再依靠和局限於特定的作業系統。由於傳統的 OPC 是基於 Microsoft 的技術進行開發，使得 OPC 的應用程式只能在 Windows 系統上運行，而 OPC UA 將支援性拓展到 Linux、Unix、Mac 等平臺。

2. OPC UA 通訊機制

OPC UA 提供兩種方式進行資料的傳輸，第一種為透過用戶端 (Clients) 與伺服器端 (Server) 的架構，用戶端向伺服器發送一個 Request，伺服器接收到 Request 後產生相對應的 Response 回傳至用戶端；第二種為透過發佈 (Publish) 與訂閱 (Subscribe) 的模式，經由仲介層進行接收與訊息發送。透過這兩種資料傳輸模式，加上 OPC UA 能夠支援不同類型的通訊協定，使得機台間數據傳送的可攜性和效率能夠得到良好的提升。

在 Client/Server 模式中，OPC UA 會定義伺服器可提供的服務模型，並可依據不同的用戶端來支援特定的服務。另一方面，傳輸的資訊結構可由各廠商自行定義，或是使用 OPC UA 內建的定義。與傳統不同的是，OPC UA 提供自動探索功能 (Discovery)，用戶端不必事前知道伺服器端定義的資訊結構與服務，當用戶端與伺服器端連接時即可自動進行偵測，取得伺服器上已定義好的服務模型與資訊模型，這也讓 OPC UA 的用戶端能夠快速地在不同的設備間進行資訊整合。另一方面，OPC UA 也針對 Publish/ Subscribe 的資訊交換提供相對應的機制。

除了能相容不同的通訊協定外，OPC UA 另一個特色是能夠確保已發佈資料的可靠性。所有 OPC 伺服器的主要功能皆包含資訊發佈和事件通

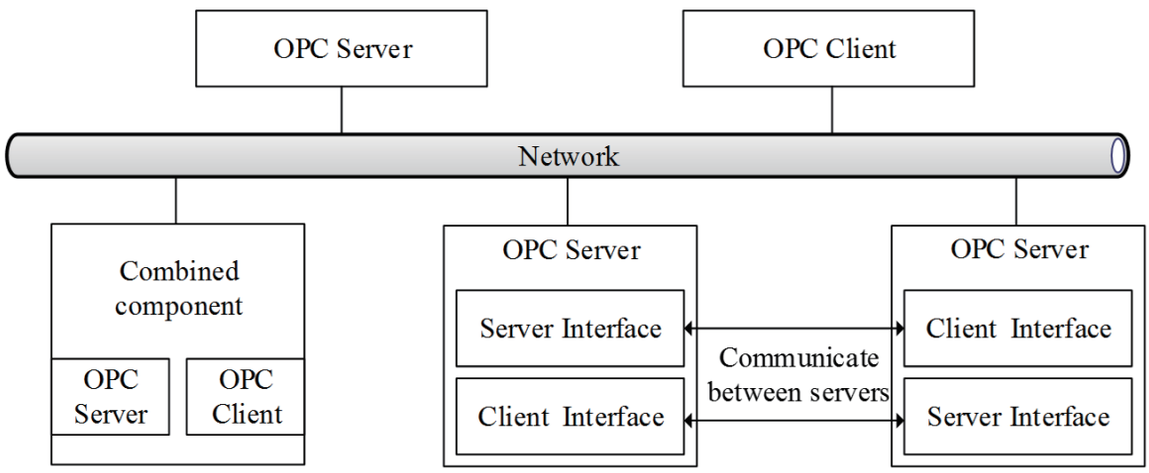

圖 2 OPC UA Client/Server 連線機制

知，讓伺服器發生錯誤時，能將錯誤資訊主動推播至用戶端；此外 OPC UA 也提供用戶端資料傳輸的快速檢測和恢復機制，可大幅的減少資料在傳輸時因封包遺失導致底層通訊協定所造成長時間的逾時。

為增加對未來通訊協定的擴充性，OPC UA 將核心設計與底層運算技術和網路傳輸層區隔開來，讓之後在整合新的技術時不會更改原本的設計架構。目前 OPC UA 有支援 XML/Text、UA Binary 和 JSON 三個資料編碼，及 OPC UA TCP、HTTPS、AMQP 和 Web Sockets 四種通訊協定。

在 OPC UA 連線架構中，每一個 OPC UA 系統內可包含多個用戶端和伺服器，用戶端與伺服器可同時多對多連線；除此之外，OPC UA 的應用可以將用戶端和伺服器的部分模組進行整合，再與其他的用戶端或伺服器互動，這也讓 Client/Server 的配置更具有彈性。OPC UA 的系統連線如**圖 2** 所示，OPC UA 也提供伺服器與伺服器的連線方式，在此連線方式中，其中一個伺服器會扮演用戶端的角色，主要用途為伺服器之間彼此透過 Peer-to-Peer 的方式交換資訊，讓資訊系統可以更有效地進行垂直整合。

3. OPC UA Pub/Sub Model

在 Publish/Subscribe 機制中，用戶端與伺服器不會以 Request 和 Response 的方式直接進行資訊的傳送，而是必須經由訊息導向仲介軟體 (Message-Oriented Middleware, MOM)。訊息導向仲介軟體可以當作是一個訊息的交換場所，提供訊息傳送的正確性及負載平衡等功能，所以發佈者 (Publisher) 與訂閱者 (Subscriber) 不需要知道彼此的存在，發佈者將訊息傳送至訊息導向仲介軟體，再依照訂閱者所訂閱的項目發送訊息，為此發佈者與訂閱者之間為鬆散耦合，且彼此不需要建立通訊連結，訊息傳遞也支援同步及非同步模式。為了能夠支援多元的應用，OPC UA 將 Publish/Subscribe 模式中的訊息導向仲介軟體分為下兩種形式：

(1) Broker-less：利用網路的基礎架構作為訊息導向仲介軟體的角色，讓發佈者的訊息以封包的形式經由路由傳送給訂閱者，例如訂閱者與發佈者可使用 UDP 協定傳送封包來達成多點傳送 (Multicast)。

(2) Broker-based：訊息導向仲介軟體即為中間人 (Broker)，訂閱者與發佈者透過標準的訊息協定如 AMQP 或是 MQTT 與 Broker 進行資訊交換，所有的訊息會先發送至 Broker 上特定的佇列，並以主題 (Topic) 區分訊息的類別，而 Broker 則提供訂閱者進行監聽，當有新的訊息時再依照訂閱者訂閱的主題傳送資訊。

4.OPC UA Client/Server 架構

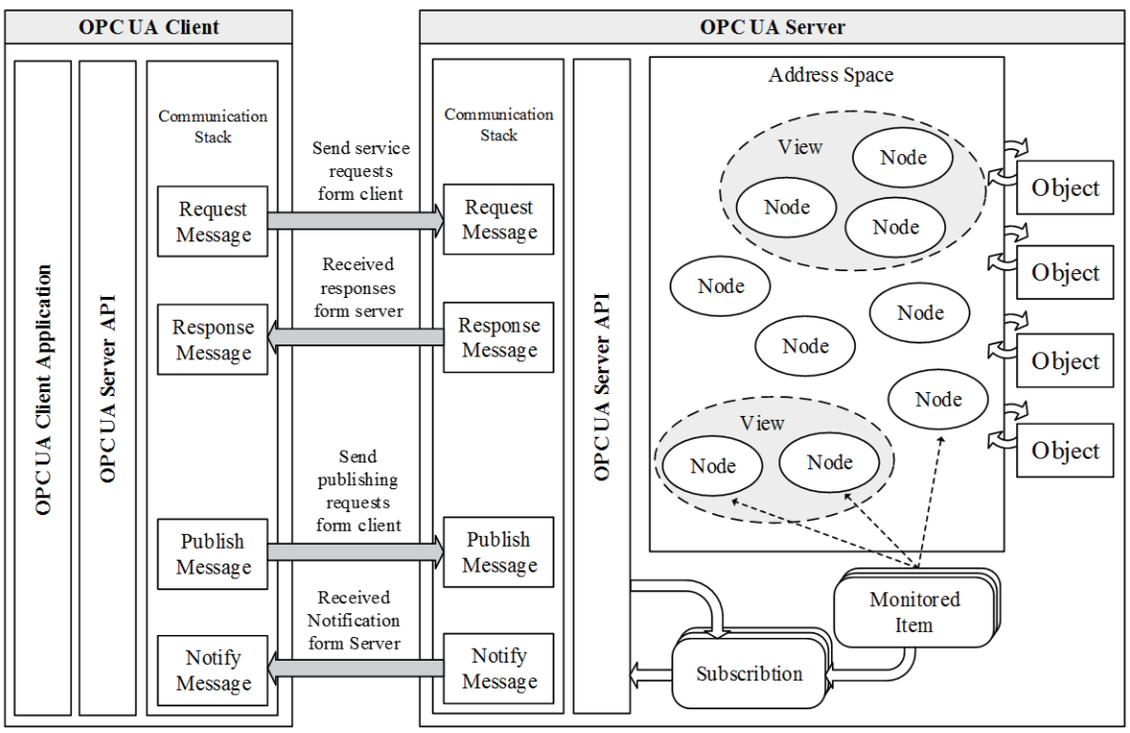

圖 3 OPC UA Server/Client 架構

OPC UA Server/Client 的架構如**圖 3** 所示，其中 Client 架構包含 Client Application、OPC UA Communication Stack、OPC UA Client API。Client-Application 主要定義 Client 中的功能，除了使用者客制化的功能外，還包括新增 / 刪除伺服器中定址空間 (AddressSpace) 裡的節點和節點參數的修改等；OPC UA Client API 內實做 OPC UA 的通訊機制，傳送和接收來自 OPC UA Server 的服務請求與回覆。OPC UA 伺服器的架構中包含下列元素：

(1) 實體物件 (Real Objects)：代表一個真實的物件，且可透過 OPC UA 伺服器的應用進行資料的讀寫。例如以 Objects 定義機臺上的溫度感測器的功能及參數，OPC UA 伺服器即可經由此 Objects 讀取目前的溫度數值，或是控制其他感測器的功能等。

(2) 定址空間 (Address Space)：定址空間包含多個節點 (Node)，節點主要是用來表示實體物件，讓用戶端可以透過伺服器的服務模組瀏覽定址空間內的節點，並取得相關的定義和參數，此外用戶端不需事前定義伺服器內節點的資料結構，可以在與伺服器連線時即時取得資料結構。

(3) 可視範圍 (View)：代表定址空間中的子集合，可以包含一個或多個節點，其主要功能為限制節點在定址空間中的存取權限，讓用戶端只能存取特定的節點資訊，在初始設定中可視範圍包含整個定址空間的節點，定址空間內可定義多個可視範圍。

(4) 監視元件 (Monitored Item)：對應到定址空間的節點，用來監控節點的狀態資訊，當節點的參數值發生變化或是發生警示訊息時，便會透過訂閱服務將訊息即時推播至連線的用戶端。

(5) 訂閱服務 (Subscriptions)：用來發佈伺服器的訊息至連線的用戶端，伺服器發送訊息的頻率可經由用戶端發送 Publish Messages 進行更改。

5.OPC UA 定址空間

定址空間為 OPC UA 中最重要的概念，功能為提供一個標準的方法讓伺服器呈現物件給用戶端，伺服器的功能及物件的資料結構都需要在

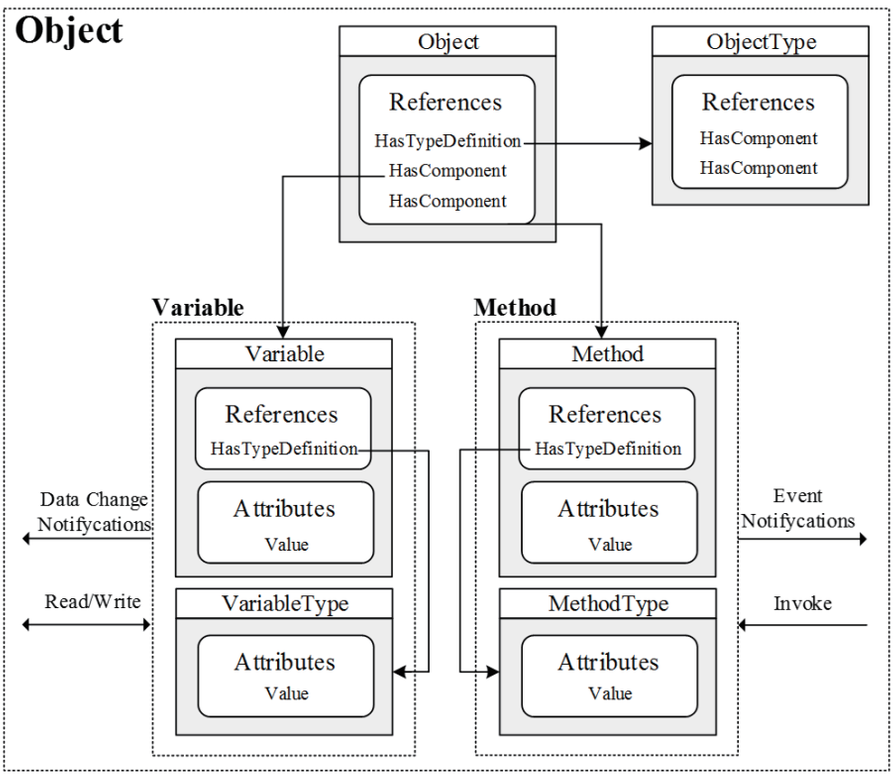

圖 4 OPC UA 物件架構圖

定址空間的基礎上實現，其中包含數據、預警資訊、事件和歷史資訊等。在定址空間內，則是透過物件來描述系統上的元件，可以是控制器、馬達或是感測器等。物件包含變數 (Variables) 和方法 (Methods)，變數用來表示物件所擁有的參數；方法則是定義物件的功能。以溫度感測器為例子，變數中可包含溫度、名稱和建立日期等，而方法中可定義溫度感測器 Reset 和 Open/Close 的功能。

節點 (Node) 為定址空間中最基本的單位，在定址空間內所有的元素都要以節點的形式呈現，包含上述物件中的變數和方法，因此一個物件會由多個節點構成。節點的型態則是定義在 Information Model 中，像是節點的基本種類有 Object、ObjectType、Variable、Method 等等，此外也可以自行定義節點的型態，比如溫度感測器可表示為 TemperatureSensorType。**圖 4** 為節點模組的基本架構，節點的架構中包含以下兩個要素：

(1) 屬性 (Attributes)：用來描述節點，用戶端可以透過 Read/Write、Query、訂閱服務和監視元件等方式取得屬性的數值。屬性中包含了 Id、名稱、功能敘述和資料型態。

(2) 參考 (References)：表示節點與節點之間的關聯。ReferenceType 用來表示兩節點間參考的關係，如果節點 A 參考了節點 B，我們稱節點 A 為 SourceNode，節點 B 則為 TargetNode，而節點 A 與節點 B 之間只能存在一種 ReferenceType。

6.VDMA–OPC UA Information Model for Robotic

雖然 OPC UA 以標準化的通訊規範解決了不同供應商和設備之間資訊互通性的問題，但由於每個廠商可以自訂 Information Model 中的資料型態，導致同一類產品可能會有不一樣的 Information Model 結構，因此還是需要針對不同的應用領域，設計一套標準來定義 Information Model。所以德國機械設備製造業聯合會 (VDMA) 與 OPC 基金會進行合作，對於運動控制系統制定一套 Information Model 規範，其中運動控制系統

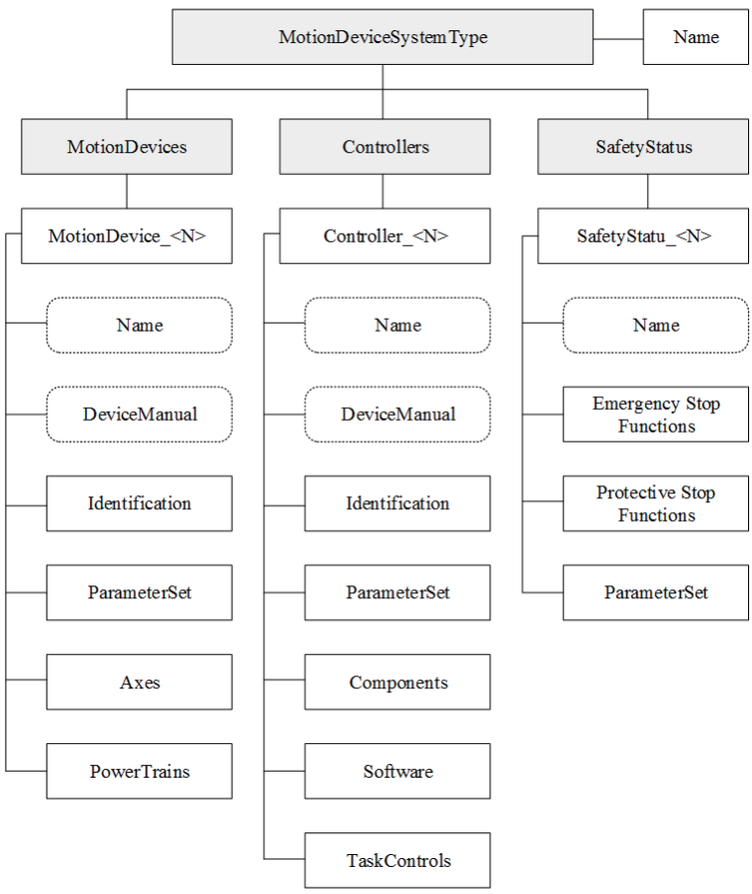

圖 5 VDMA Robotics Information model

泛指機械人 (例如工業機械人或是遠端遙控機器人等)、運動學、機械手臂、控制器以及周邊的感測元件等。**圖 5** 為 VDMA Robotic Information Model 架構，在 MotionDeviceSystemType 以下有 MotionDevices、Controllers 和 SafetyStatus 三個主要的子節點，定義了各軸的座標位置、轉速、運動狀態和系統功能等資訊。

MQTT

1.MQTT 發展背景

MQTT 通訊協定最早的全名為 MQ Telemetry Transport， 由 IBM 的 Andy Stanford-Clark 和 Arcom (現為 Eurotech) 的 Arlen Nipper 於 1999 年發明，MQTT 是專門為 IBM 的 MQ 系列產品所設計的輕量級通訊協定，2011 年 IBM 和 Eurotech 公司宣佈加入 Eclipse M2M 工作組織，並將 MQTT 原始碼捐贈給 Eclipse 的 Paho 計畫，2013 年 IBM 將 MQTT 3.1 版本交給結構化資訊標準促進組織 (OASIS) 進行標準化，隔年發布了 MQTT 3.1.1 版本正式成為 OASIS 標準，並於 2016 年成為 ISO 標準 (ISO/IEC PRF 20922) 下的開放式通訊協定，定義為 Message Queuing Telemetry Transport。

2.MQTT 技術特色

MQTT 是基於發佈 (Publish)/ 訂閱 (Subscribe) 架構的通訊協定，專門設計給硬體資源有限的設備或低頻寬、高延遲以及不可靠的網路所使用，即使在如此嚴雋的環境下，仍然能確保訊息傳遞的可靠度與品質，此外 MQTT 和 HTTP(HyperText Transfer Protocol) 都屬於網路架構中的應用層，底層皆依靠 TCP/IP 進行傳輸，但 MQTT 的標頭檔 (Header) 相較 HTTP 更加的小，採用數字編碼，

Fixed header								
bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet Type				DUP flag	Qos level		RETAIN
byte 2	Remaining Length							

Control Packet Type		
Name	Value	Description
Rservred	0	Rservred
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Rservred	15	Rservred

圖 6 MQTT 訊息固定標頭檔格式

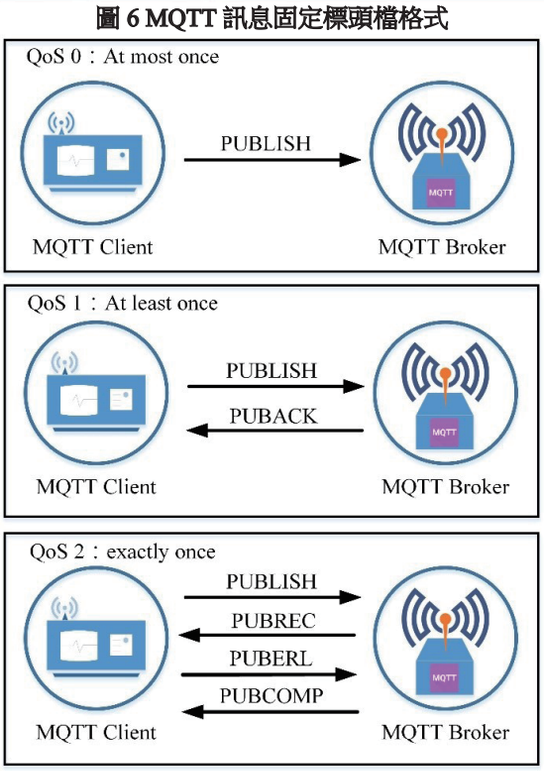

圖 7 MQTT QoS 運作模式

僅占 2 bytes，協議中規定了每個控制項的封包標頭檔一定有固定標頭 (Fixed header)，格式如**圖 6** 所示，有些訊息格式還會含有可變標頭 (Variable header) 與消息內容 (Payload)，依照訊息種類 (Control Packet Type) 不同有特定規則。以上這些特性使得 MQTT 非常適合作為 M2M 以及 IoT 設備間的溝通協定。統整 MQTT 的特色功能如下：

（1）輕量化的資料傳輸協定、設計簡單易於應用。

（2）適用於頻寬低、延遲性高的不穩定網路環境，或資源、電量受限之設備。

（3）基於發佈 (Pub)/ 訂閱 (Sub) 架構，傳輸訊息依靠三個角色：代理人 (Broker)、訂閱者 (Subscriber) 和發佈者 (Publisher)。

（4）使用主題 (Topic) 來過濾訊息，主題與訊息的格式為 UTF-8 編碼的字串。

（5）提供三種服務品質 (Qualities of service, QoS) 機制，QoS 0：最多一次 (At most once)、QoS 1：至少一次 (At least once)、QoS 2：只有一次 (Exactly once)。如**圖 7** 所示，選擇 QoS 0 時用戶端只傳遞一次訊息，並且不驗證，而 QoS 1 會確認訊息至少被傳送到一次，使用 QoS 3 會進行四步驟交握機制，確保訊息一定會被傳送與接收到一次，是最為可靠也相對耗資源的服務品質。用戶端即使訂閱了層級較高的 QoS，當 Publisher 向該 Subscriber 發佈訊息的 QoS 未達時，Subscriber 僅會以較低的 QoS 接收到該訊息。

（6）使用訊息保留功能時，需要將 MQTT 封包標頭的 RETAIN Flag 設置為 True，此時 Broker 會保留 Publisher 發送的最後一筆訊息以及對應的 QoS 值，當有 Subscriber 訂閱存有保留訊息的 Topic 時，Broker 便會向該 Publisher 發佈該筆訊息。

（7）具有「最後留言 (Last Will & Testament)」特性，用於伺服器的連線異常中斷時，通知用戶端有非正常斷線的用戶，並傳送最後一筆的 last-will 訊息至該主題。

（8）心跳時間 (keepAlive) 為客戶端和 Broker 建立連線的時間間隔，以秒為單位，用來設定兩者間沒有傳遞訊息時，最長可維持連線的時間。

（9）預設的連接埠為 1883 和 8883。

3.MQTT 技術說明

有關使用 MQTT 協定的運作模式，如**圖 8** 所示，其中 Broker 作為 Server 端，負責處理作為 Subscriber 的 sub01 和 sub02 的訂閱要求，例如訂

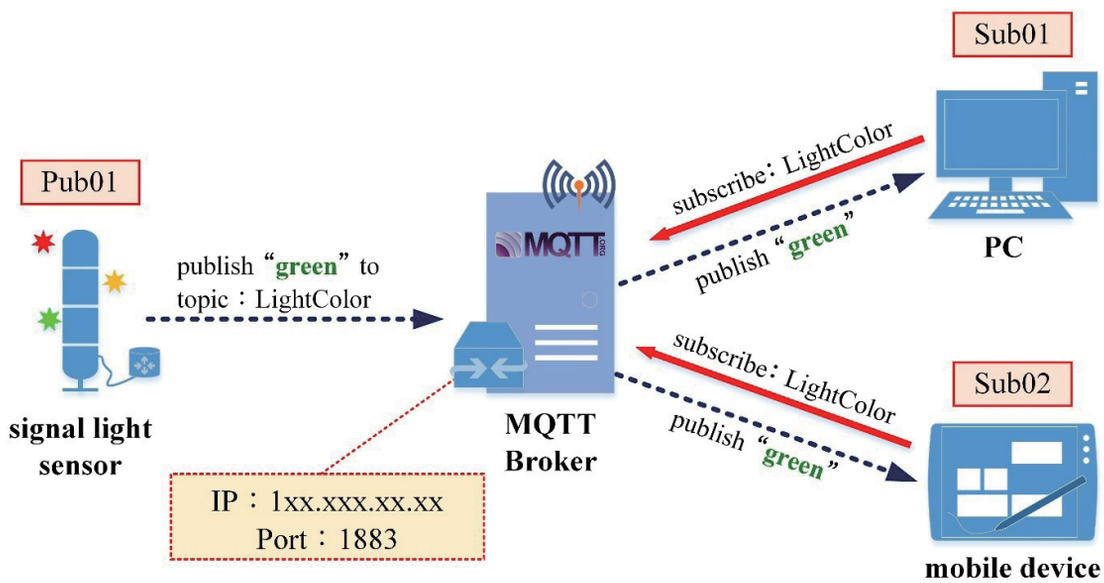

圖 8 MQTT Publish/Subscribe 架構

表 1 主題 (Topic) 命名範例

範例 1	ITRI_MMSL/EPCIO/6000/ID_001/Position/x
範例 2	ITRI_MMSL /EPCIO/6000/+/Position/x
	OK：ITRI_MMSL /EPCIO/6000/ID_001/Position/x
	OK：ITRI_MMSL /EPCIO/6000/ID_002/Position/x
	OK：ITRI_MMSL /EPCIO/6000/ID_003/Position/x
	NOK：ITRI_MMSL /EPCIO/4000/ID_002/Position/x
範例 3	ITRI_MMSL /EPCIO/6000/ID_001/#
	OK：ITRI_MMSL /EPCIO/6000/ID_001/Position/y
	OK：ITRI_MMSL /EPCIO/6000/ID_001/ErrorCode
	OK：ITRI_MMSL /EPCIO/6000/ID_001/EncoderCount/x
	NOK：ITRI_MMSL/IMP2/ID_001/ Position/z

閱 Topic 為 “LightColor”，當 Broker 接收到身為 Publisher 的 pub01 對 ”LightColor” 所發佈的資訊時，並向 sub01 和 sub02 發布 “green” 的訊息，因此同樣身為 Client 端的 Subscriber 和 Publisher 並不需要知道對方是誰，僅需要知道 Broker 的主機位址和連接埠號即可。此外 Topic 的命名規則也非常重要，關乎到訊息是否可正確被訂閱與傳遞，**表 1** 示範了主題的命名規則，說明如下：

範例 1：Topic 是採用正斜線 “/” 進行階層式的命名，其組成可使用空白字元並區分大小寫，並可搭配萬用字源的使用。

範例 2：使用 “+”，可以訂閱某不同單一階層其下一階層的共同主題。

範例 3：訂閱某階層後的所有主題，則需使用 “#”，且只能放置在主題結尾，並以正斜線區隔。

此外 “$SYS” 開頭的主題是保留給 MQTT Broker 的內部統計資料，有關主題的規劃，以下統整 HIVE MQ 檔上的幾點建議：

（1）主題開頭不要直接使用正斜線，因為等於佔用了一個字元也缺乏意義，例如 / ITRI/MMSL/0E/100。

（2）各階層內盡量不要使用空格，以免編寫程式時容易造成誤判。

（3）主題精簡扼要，以減少佔用的字元數量；具備彈性，以利於未來新產品或新功能的擴充；盡量獨特與具體化，增加辨讀與分類的詳細程度。

（4）原則上使用 ASCII 字元命名，避免特殊字元的使用。

（5）使用獨特的編碼方式或以 Client 端 ID 命名，增加主題的辨別度。

（6）避免單獨以 “#” 作為訂閱主題 (等同於訂閱全部的訊息)，會造成 Subscriber 無法瞬間負荷如此大量的訊息。

表 2 Public Broker 列表

HIVE MQ	address	broker.hivemq.com
	port	1883, 8000 (WebSockets)
Eclipse IoT	address	iot.eclipse.org
	port	1883, 80 (WebSockets), 443(WebSockets+SSL)
Moquitto	address	test.mosquitto.org
	port	1883, 8883 (SSL), 8884 (SSL), 80 (WebSockets)

4.MQTT 開源軟體與資源

由於 MQTT 是開放式的通訊協定，故在 Broker 架設與 Client 端開發上，有許多開源軟體可供選用，例如由 Eclipse 負責維護的 Mosquitt 實現了 MQTT 協議版本 3.1 和 3.1.1，可以支援 Windows、MAC OS、Debain、Ubuntu 以及樹莓派等平臺；而 HIVE MQ 提供了公用和付費版的 Broker，其官網上整理了 MQTT 基礎知識檔和相關函式庫使用教學，對初次接觸的開發者相當有幫助。**表 2** 為常被使用的公用 Broker，欲參考更多的公用版本，或其他廠商開發的 Broker，可於 GitHub 的 MQTT wiki 上找到。

有關 Client 端功能的開發工具，可選擇 Eclipse 團隊所維護的開源專案 - Eclipse Paho，

表 3 Eclipse Paho 支援功能比較表

Client	C	C++	Embedded C/C++	.Net (C#)	Java	Python	Android Service
MQTT 3.1	O	O	O	O	O	O	O
MQTT 3.1.1	O	O	O	O	O	O	O
LWT	O	O	O	O	O	O	O
SSL / TLS	O	O	O	O	O	O	O
Automatic Reconnect	O	O	X	X	O	O	O
Offline Buffering	O	O	X	X	O	O	O
Message Persistence	O	O	X	X	O	X	O
WebSocket Support	O	X	X	X	O	O	O
Blocking API	O	O	O	X	O	O	X
Non-Blocking API	O	O	O	O	O	O	O

使用者可根據不同平臺選擇對應的函式庫，建構 Client 端訪問 MQTT 伺服器之功能，各程式語言函式庫的支援功能比較如**表 3** 所示，不僅限於 Paho 專案，在 GitHub 的 MQTT wiki 上亦列出了其他語言支援的函式庫，例如支援 C 的 libmosquitto、MQTT-C；支援 .NET 的 MQTTnet、xamarin mqtt 等等。

5.MQTT 於市面工業產品上的應用

目前許多國內外廠商也看準了 MQTT 可輕易的將資訊整合至各式雲端服務或資訊系統的優點，開始將此協定導入工業控制器的資訊應用方面，並搭配 OPC UA 協定進行上層功能的整合。以下針對相關的應用產品進行介紹：

（1）Beckhoff 的 TwinCAT 3 產品具備了物聯網通訊的擴充功能模組 [14]，它使用 Tc3_IotBase 函式庫將 PLC 上的資訊透過 MQTT Broker 進行傳輸，搭配 Amazon AWS IoT、IBM Watson IoT、MathWorks ThingSpeak 平臺或 Microsoft Azure IoT Hub 以上四種提供 IoT 設備連接的雲端服務，可將 PLC 的工作資訊上傳至雲端進行應用。

（2）威倫所開發的 cMT 系列閘道器 [15] 內建 MQTT 與 OPC UA 功能，可直接將現有控制設備與閘道器對接，將系統連上 IIoT，搭配遠端監控軟體 EasyAccess 2.0 ，可於第一時間由行動化裝置上瞭解工廠系統之狀態。

（3）泓格的 UA 系列 IIoT 通訊服務器 [16]，亦採用了 OPC UA 與 MQTT 協定，可將控制設備資訊整合至 IT、OT、Cloud 與 Web APP 等各式系統，開發雲端物聯網應用

（4）研華提供的嵌入式物聯網解決方案 [17]，整合了各式標準通訊協定如 Modbus、OPC、MQTT、BACnet，藉由工業物聯網通訊協定整合雲端服務，可進行資料管理與分析，企業可藉由這些產品的輔助，加速開發智慧工廠或其他領域的智慧化應用。

OPC UA 及 MQTT 於運動控制系統之整合應用

本文結合 OPC UA 和 MQTT 兩個通訊協定實作一個資訊擷取系統，配合工研院機械所開發的運動控制軸卡與 MCCL 運動控制函式庫，蒐集機械手臂控制系統的資訊並傳送至上層的應用如 MES、SCADA 與 ERP 等。資訊擷取系統的應用情境如**圖 9** 所示，在機械手臂的控制器中，下層為運動控制軸卡，搭配上層的 MCCL 運動控制函示庫進行運動軌跡規劃以控制機械手臂；另外在機械手臂上架設 MQTT Client，將感測器的資訊發佈至 MQTT Broker。本系統開發以開源軟體 mosquitt 架設 MQTT Broker，可安裝於 Windows 系統的控制器或工作 PC 上，MQTT Client 採用支援 .Net C# 的 M2Mqtt 函式庫，MQTT Client 的應用程式介面如**圖 10** 所示。

在資訊擷取系統中，主要包含 MQTT Client

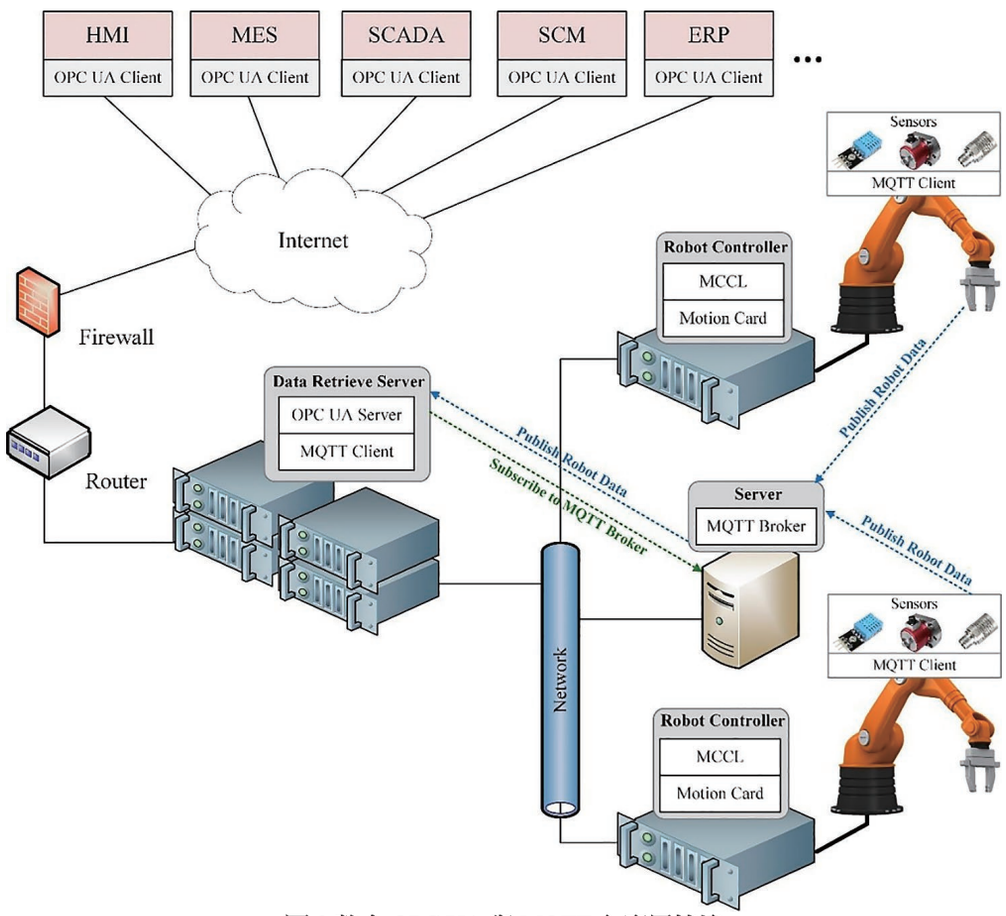

圖 9 整合 OPC UA 與 MQTT 之應用情境

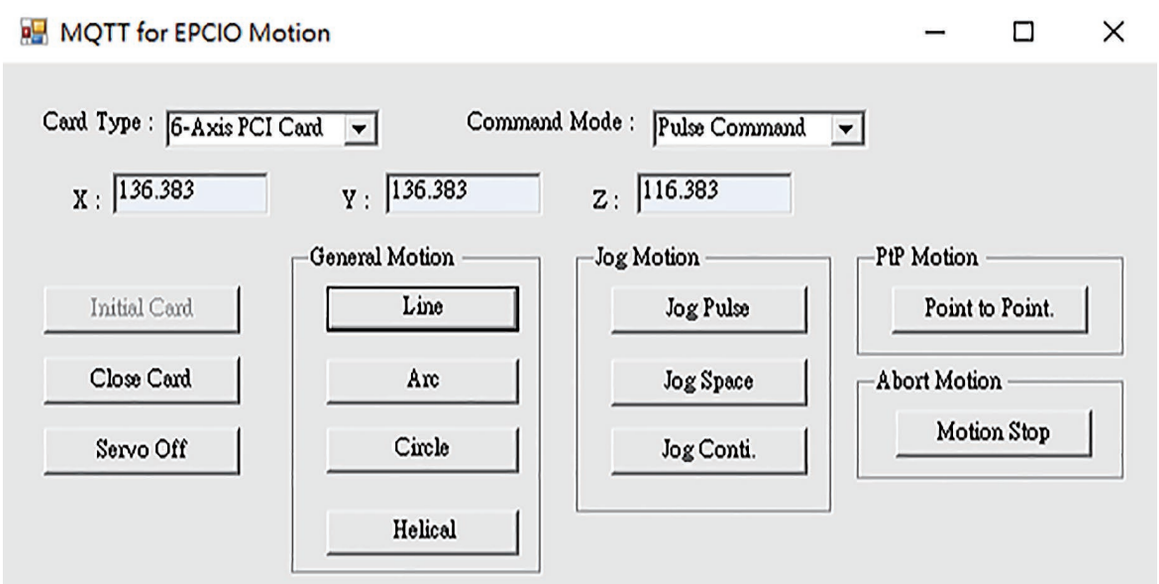

圖 10 MQTT Client 執行畫面

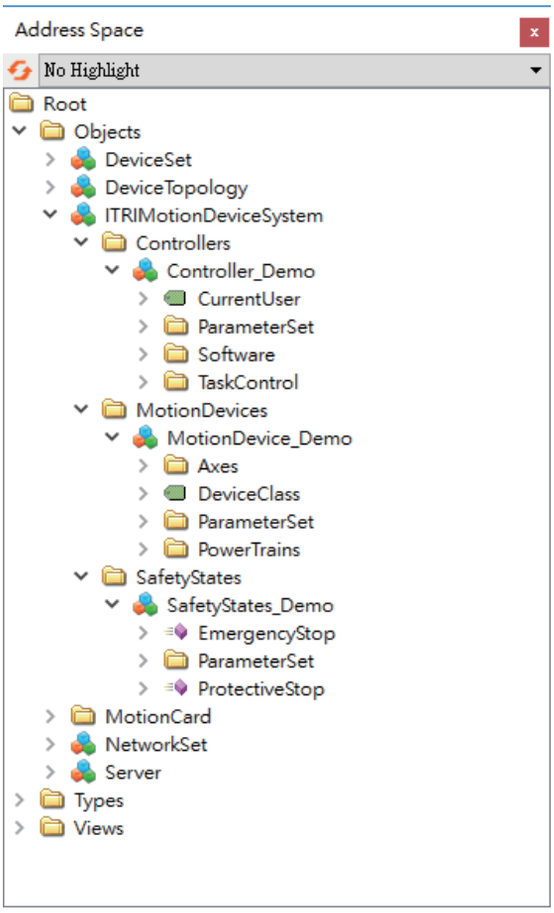

圖 11 OPCUA Server Address Space 節點架構

與 OPC UA Server 兩個部分，以 MQTT Client 訂閱 MQTT Broker，週期性取得控制器所蒐集的資訊參數，然後經由 OPC UA Server 傳送至上層應用中的 OPC UA Client，OPC UA Server 的 Inforamtion Model 使用 VDMA 規範來定義，Address Space 中的節點架構如**圖 11** 所示。透過整合 OPC UA 與 MQTT 兩個通訊協定，期望能藉由軟體功能的改良，協助工具機或控制器之功能升級與擴充。

結論

工業自動化是一個非常重要且極具吸引力的市場，目標是將資訊技術、生產系統和產線設備之間進行整合。為此本文介紹了工業 4.0 中兩種重要的通訊協定，其中 OPC UA 以統一的通訊規範解決設備間互通性的問題，並結合了兩種通訊機制以支援不同的應用情境，除了可以透過 Client/Server 架構以 TCP 與 HTTPS 進行資訊傳輸之外，當工廠中的小型感測器或微控制器需要傳遞資料時，這些設備便可透過 MQTT 或其他輕量級的通訊協定，以 OPC UA 相容的方式傳送數據。因此工廠中各式設備如感測器、致動器或控制器等，皆可藉由 OPC UA 和 MQTT 的搭配，輕易地實現工廠設備間的水平整合，並與 ERP、MES 或雲端服務等上層系統傳輸與交換資料，協助工廠進行更有效的進行數據分析及應用，創造新的商業模式。目前包括威倫科技、研華科技、新代科技與泓格科技等大廠，都有相關的產品導入 MQTT 及 OPC UA 的應用。因此通訊協定間的相互搭配依然是不可或缺，如此才能找出工業 4.0 中最適合的解決方案。

誌謝

感謝工業技術研究院控制核心技術組機電控制整合部 (計畫編號 G453REE160) 的支持，使本計畫得以順利進行，特此致上感謝之意。

參考文獻

[1] D. Bruckner, M. P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, “An Introduction to OPC UA TSN for Industrial Communication Systems,” *Proceedings of the IEEE*, 2019.

[2] P. Ferrari, A. Flammini, S. Rinaldi, E. Sisinni, D. Maffei, and M. Malaram, “Impact of Quality of Service on Cloud Based Industrial IoT Applications with OPC UA,” *Electronics*, Vol. 7, no. 7, Jul. 2018.

[3] M. Schleipen, S. S. Gilani, T. Bischoff, and J. Pfrommer, “OPC UA & Industrie 4.0 - enabling technology with high diversity and variability,” *Procedia Cirp*, Vol. 57, pp. 315-320, 2016.

[4] T. Hannelius, M. Salmenpera, and S. Kuikka,

"Roadmap to adopting OPC UA," *IEEE. 2008 6th IEEE International Conference on Industrial Informatics*, pp.756-761, 2008.

[5] P. Drahoš, E. Kučera, O. Haffner, and I. Klimo, "Trends in industrial communication and OPC UA," *IEEE*. 2018 Cybernetics & Informatics (K&I)., pp. 1-5, Jun. 2018.

[6] OPC UA Robotics Companion Specification, 1 ed., *Verband Deutscher Maschinen-und Anlagenbau*, Germany, 2018.

[7] OPC Unified Architecture Specification, 1.04 ed., OPC Foundation, 2017.

[8] There is no Industrie 4.0 without OPC UA, OPC Foundation, 2017.

[9] U. Hunkeler and H. L. Truong, A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks," *IEEE. 3rd International Conference on Communication Systems Software and Middleware and Workshops* (COMSWARE'08), pp. 791-798, 2008.

[10]D. Locke. (2010). Mq telemetry transport (mqtt) v3.1 protocol specification(v3.1). IBM developerWorks Technical Library, 15.

[11]A. Banks, R. Gupta. (2014). MQTT Version 3.1. 1. OASIS standard, 29, 89.

[12]R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *The Journal of Open Source Software*, Vol. 2, no. 13, May. 2017.

[13]D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*, pp. 1-6, Apr. 2014.

[14]Beckhoff TwinCAT3, Available: https://www.beckhoff.com/TwinCAT3/

[15]Weintek cMT, Available: https://www.weintek.com/globalw/Product/Product_cMT.aspx

[16]ICPDAS, Available: http://www.icpdas.com.tw/index_tc.php

[17]Advantech Co., Available: https://www.advantech.tw/

[18]ISO/IEC 20922:2016, Available: https://www.iso.org/standard/69466.html

[19]MQTT Paho Client, Available: https://www.eclipse.org/paho/downloads.php