exeley INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS

Article | DOI: 10.21307/ijssis-2021-022 Issue 1 | Vol. 15 (2022)

Optimising data visualisation in the process control and IIoT environments

Robin Maharaj, Vipin Balyan* and Mohammed Tariq Khan

Department of Electrical, Electronics and Computer Engineering, Cape Peninsula University of Technology, Cape Town, South Africa.

*E-mail: vipin.balyan@rediffmail.com

This paper was edited by Ashutosh Sharma.

Received for publication August 8, 2021.

Abstract

In the present context of the 4th Industrial revolution, there is a growing need to get data from different data sources in a standard data format. This paper presents a solution to achieve this convergence in using IoT technology, extracting available data, and making it available to high-level systems in a standard, low latency framework. This paper presents different protocols like OPC UA, Message Queuing Telemetry Transport (MQTT), and Constrained Application Protocol(COAP) to achieve this data transportation/acquisition. It also presents an emerging Low Power Wide Area Network (LPWAN) technology, LoRa WAN, to augment the data of the process control system, explicitly extending the range of sensors to wireless data points.

Keywords

LoRa WAN, MQTT, OPC UA, COAP, Device gateway.

Introduction

Digital transformation creates a need for protocols convergence to ensure cross-system data exchange (Bassi, 2017) and efficient machine-to-machine communication. OPC UA, MQTT, and COAP are some protocols that facilitate cross-system data exchange in the fast-changing data environment. These protocols are functionally achieving common goals in terms of data interoperability but have significant differences in their make in both overheads and data latency. In the IIoT and the 4IR space, there is also a need to increase the data points without implementing expensive process control systems but rather adding to the existing process control system with additional sensors functioning in the IoT domain (Muller et al., 2017). If there were a need for an ideal protocol/architecture, it would have been a protocol like OPCUA had it not been for its heavy-weight nature. Protocols like MQQT. COAP and AOQP offer a lightweight solution to facilitate this digital transformation but do not offer the frills of a protocol/architecture like OPC UA (Schleipen et al., 2016).

Process data are now seeing itself integrate into high-level systems. These data are also being made available to web interfaces where even the end-user/customer can view process/manufacturing information in real-time or, for the very least minimal latency. There is, of course, the concern of the security of data that is where architectures like OPC UA come to the rescue (Durkop et al., 2015). But in the IoT scheme of things seems to be too heavy-weight. There is also a need to provide these data at a minimal cost.

There are a few debilitating factors in the South African context when researching/testing the options of transferring data across communication networks, especially cellular networks that block traffic on non-standard ports. UDP is restricted and barred from going across cellular networks as a single user, even for research or academic purposes. However, we were not limited to UDP protocols when from a Lan perspective.

Overview of protocols

From an IIoT perspective, there is often the question of TCP or UDP. TCP/IP specifies that the telegram must be acknowledged automatically after transmission. If the transmission is negative or there is no acknowledgement, TCP/IP repeats the message several times automatically. However, if the acknowledgement is impossible for technical reasons, TCP/IP consumes energy unnecessarily (Silveira Rocha et al., 2018). So what this spells out is that if the network is unreliable and power is a key driver, then TCP/IP is not a viable option as this could be too energy-intensive for a low-power WAN solution. TCP in its pure implementation would not be suited for IoT or lower power applications in unreliable networks. In IIoT frameworks closely associated with TCP/IP, MQTT addresses some of the drawbacks of a pure TCP connection.

There are many emerging LPWAN technologies, and quite a relatively new radio-based IoT technology is NB-IoT. However, data were already transmitted via radio before NB-IoT. Another example is 802.15.4 with IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN). 6LoWPAN is a transmission protocol on IP Point-to-Point Protocol (PPP) and was specified for radio and wire. The layer above 6LoWPAN is UDP, if a radio channel is disturbed, then it generally makes no sense to start a new transmission of the telegram. UDP does not send an acknowledgement and expects an acknowledgement, making it most suitable for constraint low power WAN networks. If there is a need for acknowledgement and elect to use UDP, then the acknowledgement must be done in the protocol layer above it. The protocol layer above UDP can be a protocol like COAP in the case of 6LoWPAN. CoAP regulates that a telegram is sent (with or without acknowledgement). This means that the programmer himself can decide whether he needs an acknowledgement or not. Most modern programming libraries have COAP libraries to facilitate the seamless use of this UDP-based framework.

One of the other objectives of this research was to demonstrate how to make low-level or field level sensor data available to High level/application layer MES/ERP systems with non-propriety and interoperable protocols.

The challenge in the modern context is to marry IOT technology with higher-level systems to give customers or end-users the comfort of information, allowing maintenance and operation managers to optimise systems and processes with this added intrinsic data.

In the IIOT world, one of the most significant challenges is seamlessly and effortlessly getting sensor data to a WEB API or MES. More and more processes are being automated in the modern industrial environment, thus providing new possible data sources to high-level systems. Still, not all of the data at these process control systems data needs to be exposed to the MES/high-level system, and there needs to be some aggregation and packaging of the data in the format that these high-level systems require.

At the same time, these present and existing process systems do not always provide all the data that the MES system requires, and sometimes there is a need to get additional data from the same environment/system at minimal costs. In some cases, the data points are geographically dispersed, and devices need to provide for these data source. This research has delved into all the different data sources to transmit the data from these different data sources to a standard "device gateway". This paper looked at the various technologies and frameworks to get data to the device gateway. The overview of this paper, as seen in Figure 1, covers the following data sources:

A. LoRa WAN Gateway.
B. Data logger using GSM/LTE.
C. SQL database with JSON engine using System on Chip LTE modem.
D. Packet Data from Radio/Tetra Network.

Lorawan

LoRa WAN is the LPWAN technology that was used to get the additional data required by the MES that the existing Process Control System did not provide.

LPWAN, LoRa WAN are gaining ground in the IIoT space and are finding themselves being implemented in a wide range of applications. But to get the full benefit of these technologies, it is apparent that there needs to be some low-level implementation of the technology and some system integration into MES systems.

Many developers and engineers chose the off-the-self and rapid development route, but those routes fail to deliver the full potential of the technology. One of the problems/debilitating factors encountered during the implementation of the LoRa WAN solution was that all the off-the-shelf products encountered for a LoRa end node had only one analogue sensor input coupled to the LoRa radio node. It would be beneficial to have two or more analogue inputs interfaced with one LoRa radio module in an industrial plant. For this reason, this research implemented two analogue-type sensors with the capacity to interface more if required.

The true benefit of these IoT implementations lies in the integrative approach and the appreciation and performance of the emerging protocols and applications to get actual value for money. This paper

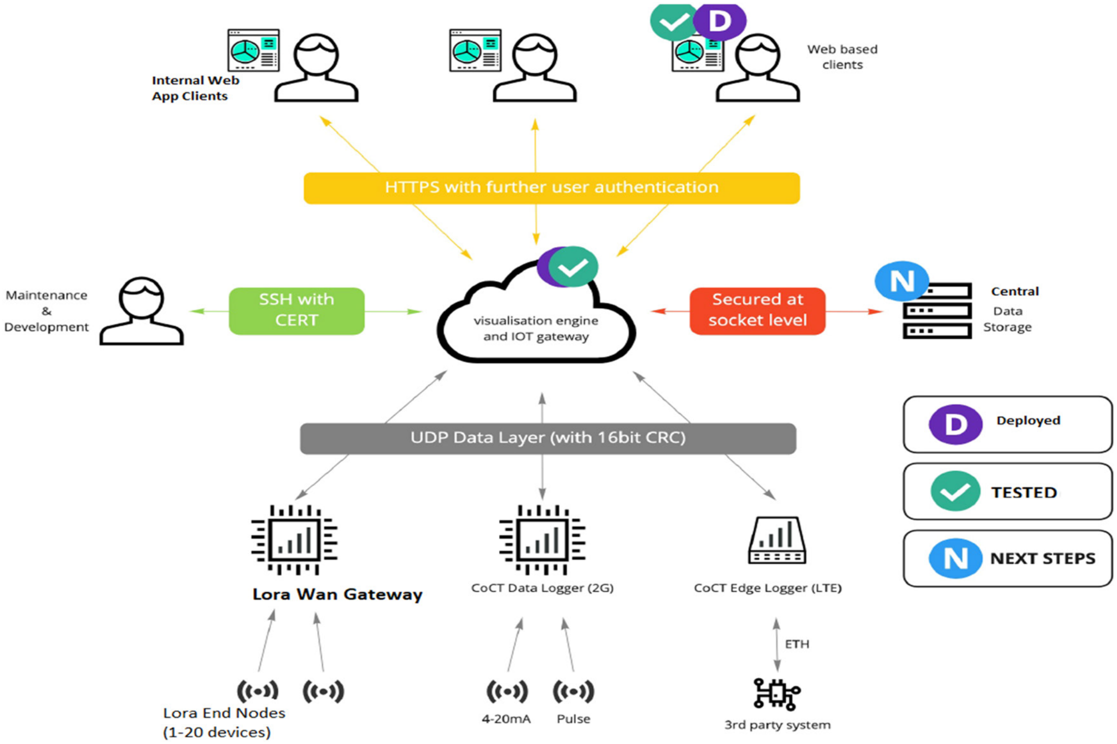


Figure 1: Architecture of new framework.

will present a solution encompassing this ideology: appreciating the hardware architectures, writing some low-level code, and adapting existing libraries to optimise.

Some of the present debilitating factors with off-the-shelf solutions are:

i. Most LoRa end nodes only allow one sensor signal per end device.
ii. Managing transmission intervals are limited with the shelf devices.
iii. On the LoRa gateway: Limitations on which UDP packet Forwarder can be used.
iv. Hardware Configurability limitations if no embedded development is done.

Present and future architectures

LoRa is one of the fastest-growing LPWAN technologies that warranted some investigation into a technology option for getting industrial sensor data to a high-level system. There are numerous research papers published with architectures presented for LoRa public and private networks. Still, few address the system integration of LoRa to MES systems purely because many LoRa implementations have their propriety application server coupled to it.

There are presently various Process automation plants controlling different wastewater water treatment plant processes. These processes are controlled by both PLCs and local SCADA systems. All this plant information is present locally in SCADA, PLC, and sometimes local SQL databases. All these data are used by the process systems as part of the control system and the visualisation of the control system when it comes to the SCADA system (Calderón Godoy and Pérez, 2018) Leveraging this local data at a remote site or system can be valuable if this data could be aggregated and parsed as data to high-level systems or a web application or even for data analytics. The local SQL data can also be augmented by IoT devices/systems that can be added to the same central API/MES. For example, in the Wastewater plant environment, IoT devices could relay Air quality, Inflow, and Outflow data of the plant, which can be added /aggregated to existing process plant data. The aggregated data would provide intrinsic value to the customer/end-user in the form of performance dashboards. The IIoT component of this is where LoRa WAN seems a viable technology option.

This research implements a wastewater treatment plant dashboard using various IoT devices and the local SQL plant database to visualise Important KPIs in a web-based application.

Another challenge in accomplishing the above is to keep cost and labour at a minimum while providing additional data to the MES/web application.

In this research, data sources were investigated to present/parse data to the MES/web application, namely:

- i. Existing data from process control systems and SCADA are stored in an SQL database. Here only relevant data needed by the MES will be extracted from this process control database and made available to be parsed to a High-level system like an MES or Web API.
- ii. Additional IoT data using LPWAN technologies, specifically LoRa WAN, to enhance the data of these plants to provide Overall Equipment Effectiveness (OEE) data and Key Performance Indicators (KPI) data.
- iii. Additional data sources using 2G/3G dataloggers parsing data in a prescribed format.

Plant data

PLC/PAC systems have variables and tags already defined locally in these process-controlled systems. Plants that have local SCADA systems also have these tags mapped to the SCADA systems.

Some Modern process plants may also have a SQL database storing these data, especially in Water and Waste Water Treatment Plants. But this data does not get processed further than the plant, and there is an opportunity to make this data available to MES /high-level API. This data that is local to the plant can be aggregated and sent to the remote application server. In this paper, we demonstrate how some of the data from these plants can be a data source for MES/API to be translated to end-user/customer information to improve production KPI's and Customer perspective.

Lora wan iot device architecture

To provide additional data to the MES systems, it was necessary to implement a solution to provide the low-cost answer that is suitable to provide data to the MES without the need for complex hardware infrastructure to achieve this additional data. In Water or Wastewater treatment plants, these data points are sometimes more than 1200m from the nearest communication gateway or process control plant. Finding low-cost wireless technology was the clear plan but this technology needed to be adapted for the wastewater treatment environment. The wastewater environment is industrial, and as mentioned earlier, many LoRa-off-the-shelf devices offer only one analogue signal per end device. Added to this, the device does not conform to any rugged specifications like IP67 rating. The end device required to capture the data needed for the wastewater environment needs to conform to these specifications to endure extraordinary environmental challenges experienced in WWTP's.

Cost-benefit of lora wan option

Also, with as many as 20 additional standalone data points, GSM/LTE loggers can be a costly option. In South Africa, just the GSM data logger without sensors could retail around R15000 to R20000 each. To get this additional data to the MES would cost R400000 per plant for this option with additional monthly data cost for each of the twenty devices. With LoRa, on the other hand, it would require 20 LoRa nodes at R700 each plus one LoRa WAN Gateway of R5000, bringing the total cost of LoRa solution under R20000. Regarding the monthly data cost, the LoRa solution will have one device gateway logging data to a remote site putting the operational cost of GSM 20 times the cost of using the LoRa WAN option.

Lora technology briefly

LoRa is a spread spectrum modulation technology (Orange, 2016). One of the most significant challenges in acquiring data in the environment like a wastewater treatment plant or a water treatment plant is the distance of the plants from the central control room, typically where all the SCADA is housed. There are several distributed process control plants, all relaying data to the central control centre.

Adding data points to these distributed plants utilising existing legacy process control systems can be very expensive, especially when there is no more capacity to adding these additional data points. Also, getting sensor data even when there is capacity can still be very expensive and labour-intensive because of the need to use fibre etc. In the modern era, wireless options seem to be the preferred choice. Fortunately, this is where options like LoRa can deliver on the needs of the system.

This paper presents a variation of the standard implementation of LoRa WAN. To get data to the Device gateway, it was necessary to implement a private LoRa WAN network with an MQQT bridge implemented on the LoRa WAN Gateway. The Typical LoRa WAN Implementation consists of multiple end nodes, LoRa gateway, LoRa Network server, LoRa application server. The LoRa gateway either has the legacy Semtech packet forwarder, or another popular UDP packet server is the LorIoT packet forwarder. Most of these packet forwarders have an Application server coupled to them for Visualisation purposes. The application server could be local to the Gateway or a remote location.

This research implements the LoRa end node with multiple sensors using the B-L072Z-LRWAN1 development board. The 4–20 ma was interfaced via SPI by mikroe Click board, and the 0–5V sensor was interface using the onboard ADC peripheral of the development board. All coding for these interfaces was done using the Mbed platform. There were fortunately many rapid development platform libraries that made this interfacing relatively seamless. The development process concerning interfacing of the LoRa radio module was done in a two-phase approach.

The first phase was to interface an STM32F4 discovery board to the two sensor variations mentioned above and the interface sx1276 LoRa radio module. During this 1st phase, the sensors and the LoRa radio module were connected with wire jumpers for proof of concept but were too cumbersome for actual testing.

In the 2nd phase, because we worked with radio technology, i.e. LoRa chirp spread spectrum, we wanted to limit the number of jumper connections and any factors that could negatively influence the results.

The above diagram Figure 2 illustrates the first phase of the hardware development for the LoRa End Node. The proof of concept and testing was carried out on an STM34F4 discovery board. Yongde et al. (2014) The coupling to the 4–20 mA interface was implemented using the microcontrollers SPI to read analogue values. The SX1276 LoRa radio module was also interfaced to the discovery board using SPI, whereas the 0–3.3V IOT sensor was interfaced directly to the microcontroller using its 12-bit resolution ADC interface.

All the coding and deploying in the 1st phase was done using the Keil µvision IDE and STM CUBEmax. This proved the concept before rolling out to the more expensive development board used in phase 2.

In the second phase, we used the B-L072Z-LRWAN1 development. This board has a built-in sx1276 LoRa radio module. Instead of interfacing the two sensor types directly to each product, as in phase one, dummy sensor data was added to the variables defined in phase one to test the multiple end nodes more robust and less robust cumbersome. The focus of the testing was to test the capability of the communication interface as well as data rates and latency. The B-L072Z-LORWAN1 board is mbed-enabled so all coding and deploying was done using the mbed rapid development platform.

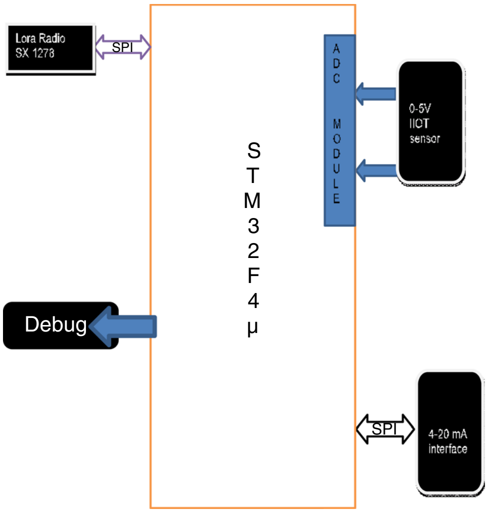


Figure 2: Hardware layout LoRa end node device.

Implementing the different data sources

To register the respective end nodes on the LoRaWAN Gateway, each end node needs to be programmed with their unique dev EUI, an app EUI. Gateway (Robustel R3000) would parse data to the Gateways GSM modem in the same format. The difference between this option and the GSM data logger is that this option would allow for 20 different end nodes transmitting sensor data to one Gateway where the Gateway would then be transmitted to the central device gateway. The benefit of this option is the reduced hardware cost as well as there is no need for the external permanent power source to get these analogue values.

Lora wan gateway as a data source

There are many implementations of this LPWAN technology which also include public or private network options. Private LoRa WAN network is essentially a LoRa WAN Architecture where the LoRa WAN server and application server either is resident on the Gateway itself or the Local network. In other words, the visualisation and authentication are done local to the network and is not done via a WAN. The other differentiator separating these two variations is the LoRa WAN Server and LoRa application server implemented. The LoRa WAN server can further be made "more private" by deploying a LoRa UDP packet forwarder that is propriety and excludes the legacy option of the LoRa WAN server commonly referred to as the SEMTECH legacy packet forwarder.

The LoRa WAN gateway used for testing and proof of concept was the Robustel R3000 LoRa WAN Gateway. UDP packet forwarder installed on this Gateway was LoRa Server. LoRa packet forwarder is a program running on a LoRa gateway and interfaces to the LoRa concentrator to pull and push while interacting with the LoRa Network server, which could have a remote address or be running locally on the Gateway or the machine on the local network (Koon, 2020). The packet forwarder and LoRa WAN network server communicates with each other by a defined UDP packet forwarder. The legacy UDP packet forwarder has numerous shortcomings. It is not recommended for deployment on a large-scale network or an unreliable network, for that matter. Still, this does not apply because this implementation is local to a plant and does not function in the public domain.

Additional to this application, an MQTT bridge was installed to the Robustel gateway to parse the data from the LoRa Gateway to the COCT Device Gateway.

In this research, LoRa WAN implementation is simply 5 LoRa end node devices all 600m to 1200m away from the Robustel gateway. The LoRa WAN Implementation can be seen in Figure 3. Each of the devices for testing purposes had two analogue 4–20 mA dummy sensors interface to it (Koon, 2020).

Typically, it is possible to connect up to 2000 LoRa end nodes with multiple sensors transmitting to a LoRa WAN gateway. The LoRa WAN gateway is configured to have a LoRa network server and application server local to the plant and using MQTT to parse these data to the device gateway.

To receive the data acquired by the LoRa devices via the LoRa gateway, the device EUI was used as the device ID for the central device gateway for the web service. In line with parsing data to a standard gateway, we subscribed to the same format as shown in Table 2.

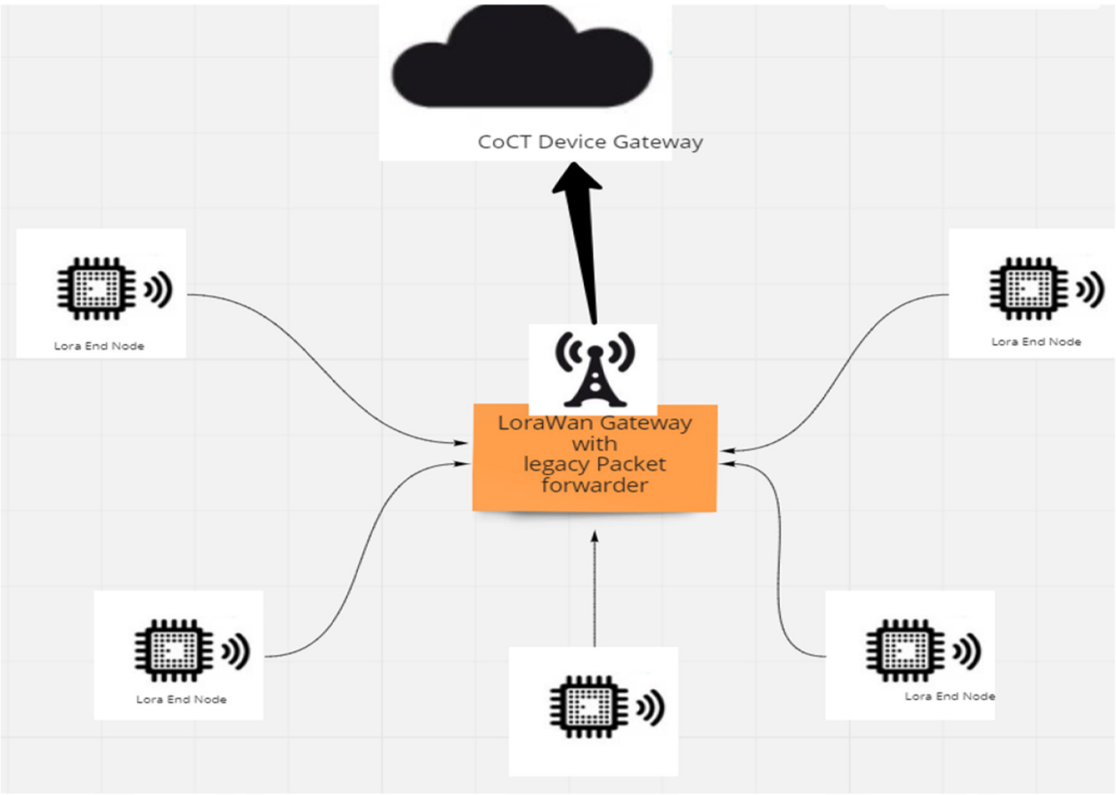


Figure 3: LoRa WAN as a data source to device gateway.

LTE/GSM logger as a data source

One of the most common types of data loggers out there is GSM/LTE loggers. They are prevalent in remote locations. These loggers are a microcontroller interfaced to a couple of digital and analogue inputs and coupled to a GSM modem via Uart or another Serial communication interface (SCI). These are usually some propriety type device parsing data via TCP or UDP using an IoT framework. So for this research and to add more context to this research, one of the data sources was the common GSM/LTE datalogger (Durkop et al., 2015). The code/firmware was also written in a similar LoRa end node, except the LTE logger transmitted data directly to the device Gateway. In contrast, the LoRa end node transmitted data First to LoRa WAN Gateway with a built-in UDP packet forwarder with other translations before transmitting to the device gateway (Artuso and Palladino, 2019).

The GSM/LTE data logger could also be implemented in very remote sites with minimal I/O requirement; this is also the most cost-effective way of adding these data points from remote locations.

The GSM logger (Figure 4) could also interface directly into the existing process control system to expose some data “tags” that the remote MES system might require. The GSM logger used in this research that was developed for this research had two hardware variations, namely:

i. 2 x 4 to 20 mA interfaced to GMS modem transmitting data to the remote site in a power-optimised way Azhari and Kaabi (2001).
ii. It has a built-in Modbus wrapper to exposed Modbus registers from a plc in this Schneider m340 plc that the MES system requires.

The GSM logger forms two of the explicit device types that the device gateways receive data from. The data format from this data source also transmits in the prescribed COCT Gateway packet format, as seen in Table 2.

The firmware of this logger was written to parse data in the format as per CoCT packet format. This research also implemented two hardware variations of this data logger; one variation was where there was power available at the location and the other where the device was battery powered. The one that was battery operated, it was necessary to implement a circuit to boost the battery voltage to a voltage of greater than 12V to power the current loop for the 4–20 mA sensors (Azhari and Kaabi, 2001). The code written for the battery version of the hardware was also sensitive to optimise power consumption to allow the device to log data for long periods without changing batteries.

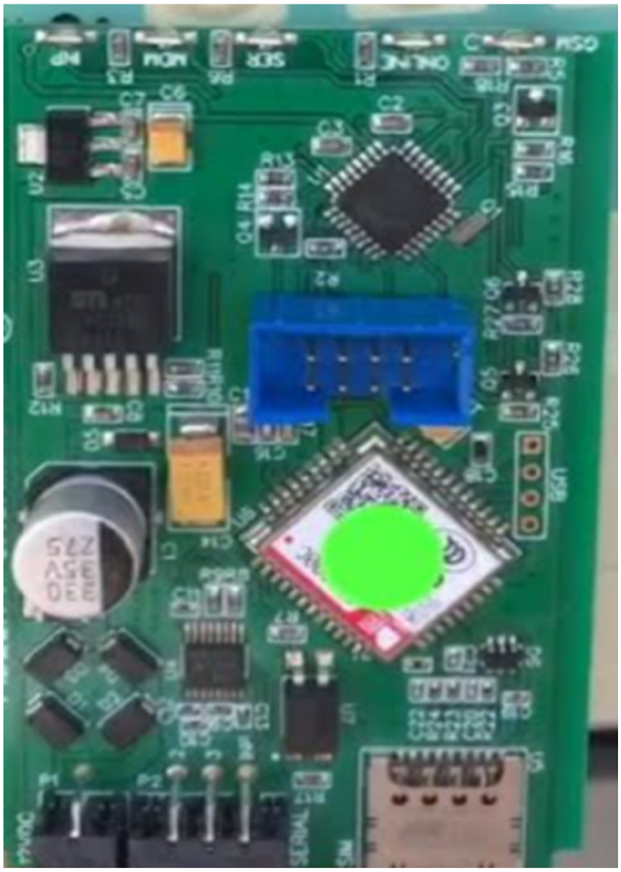


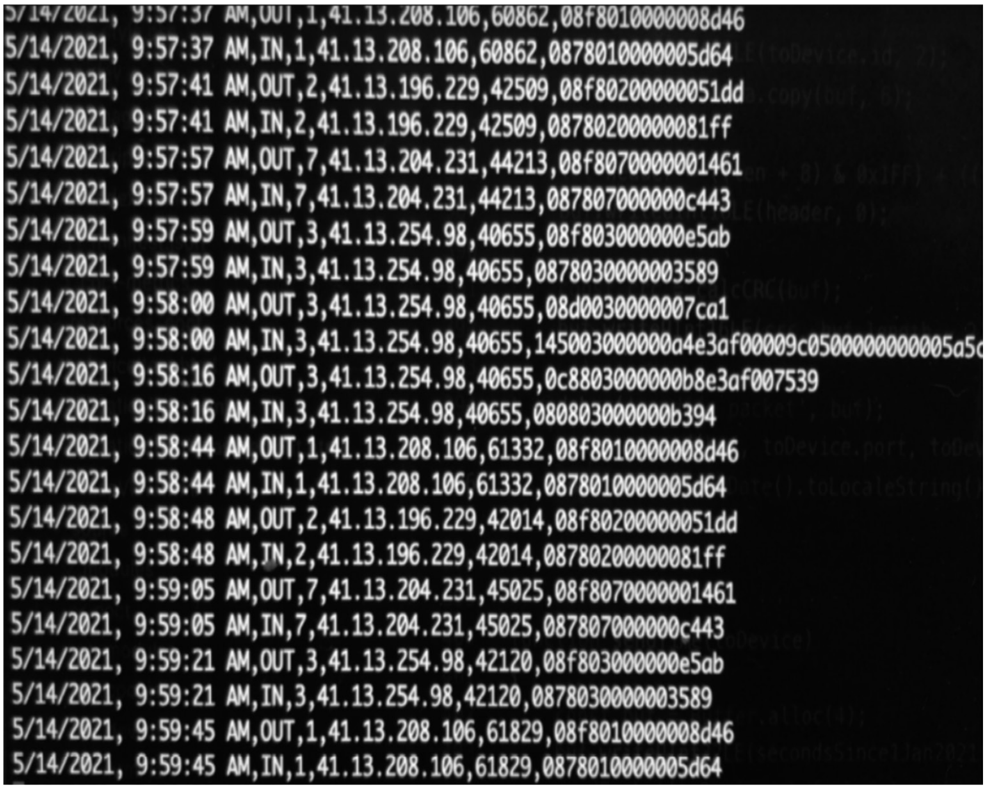
Figure 4: Footprint of GSM logger.

Seeing that this device was reading analogue sensor values, dead-banding was also incorporated in the code. The design made it possible to update the transmission intervals of data logged, but the default was set to 15 minutes. Data was also transmitted if the change was greater than a prescribed variation of the value last read.

Figure 5 captures the real-time raw data received by the device gateway as per CoCT packet format before it can be processed by Web API.

Local plant SQL database as data source

In a typical process plant environment, the architecture consists of many process control systems of PLCs, SCADA, and HMI’s with a local SQL database that stores all the relevant data points of all the process control systems. These data points/tags are an essential source of data for the MES systems (Katti et al., 2018).



```
5/14/2021, 9:57:37 AM,OUT,1,41.13.208.106,60862,08f8010000008d46
5/14/2021, 9:57:37 AM,IN,1,41.13.208.106,60862,0878010000005d64
5/14/2021, 9:57:41 AM,OUT,2,41.13.196.229,42509,08f80200000051dd
5/14/2021, 9:57:41 AM,IN,2,41.13.196.229,42509,08780200000081ff
5/14/2021, 9:57:57 AM,OUT,7,41.13.204.231,44213,08f8070000001461
5/14/2021, 9:57:57 AM,IN,7,41.13.204.231,44213,087807000000c443
5/14/2021, 9:57:59 AM,OUT,3,41.13.254.98,40655,08f803000000e5ab
5/14/2021, 9:57:59 AM,IN,3,41.13.254.98,40655,0878030000003589
5/14/2021, 9:58:00 AM,OUT,3,41.13.254.98,40655,08d0030000007ca1
5/14/2021, 9:58:00 AM,IN,3,41.13.254.98,40655,145003000000a4e3af00009c0500000000005a5
5/14/2021, 9:58:16 AM,OUT,3,41.13.254.98,40655,0c8803000000b8e3af007539
5/14/2021, 9:58:16 AM,IN,3,41.13.254.98,40655,080803000000b394
5/14/2021, 9:58:44 AM,OUT,1,41.13.208.106,61332,08f8010000008d46
5/14/2021, 9:58:44 AM,IN,1,41.13.208.106,61332,0878010000005d64
5/14/2021, 9:58:48 AM,OUT,2,41.13.196.229,42014,08f80200000051dd
5/14/2021, 9:58:48 AM,IN,2,41.13.196.229,42014,08780200000081ff
5/14/2021, 9:59:05 AM,OUT,7,41.13.204.231,45025,08f8070000001461
5/14/2021, 9:59:05 AM,IN,7,41.13.204.231,45025,087807000000c443
5/14/2021, 9:59:21 AM,OUT,3,41.13.254.98,42120,08f803000000e5ab
5/14/2021, 9:59:21 AM,IN,3,41.13.254.98,42120,0878030000003589
5/14/2021, 9:59:45 AM,OUT,1,41.13.208.106,61829,08f8010000008d46
5/14/2021, 9:59:45 AM,IN,1,41.13.208.106,61829,0878010000005d64
```

Figure 5: Real-time capture of packets arriving at device gateway.

This is also another data source that the device gateway can receive data from. These data are aggregated data that the C# application makes available to an LTE modem from the transmission to the device gateway in the prescribed CoCT Gateway packet format. This aggregation will take place locally to the database server and will do all the heavy lifting in terms of data aggregations and queries. This aggregation will be achieved by an engine that essentially will perform all the database queries and aggregation.

The JSON Engine is merely a C# application that resides on the plant database server that is responsible for aggregating data and presenting it in a prescribed JSON format so that the device gateway can understand for use by the Web server system. The C# application will connect to the Microsoft SQL server and extract data using SQL queries. The data will appear as a virtualised hardware device, much like the GSM loggers.

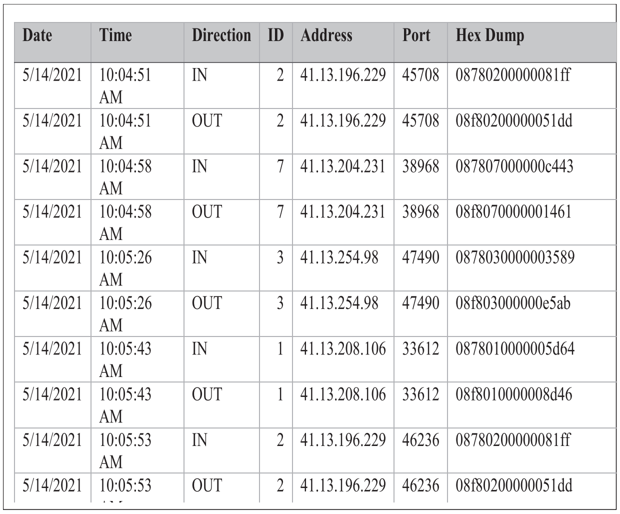
Attached to this plant server is a physical hardware device accepting input data via Ethernet to submit/parse to the existing device gateway. Any updates and new data found will be timestamped and buffered for transmission immediately to the Device Gateway. Should connectivity be disrupted, the application will cache the timestamped data for transmission later.

The C# application will package the data according to the gateway packet format seen in Table 1. One of the fields of the header of the gateway packet format is a type that represents the data source type; the other field data the C# application will serialise the data is device ID will mean as specific aggregated KPI value from 0 to 255. So all data coming from the SQL data source will have a device ID of 0 to 255, which is reserved by the device gateway.

Getting data using OPC UA & MQTT From embedded OPC UA Data source

There are numerous ways of getting data from an OPC UA data source to an MES system. Still, for this research to demonstrate how OPC UA data is parsed to the Device Gateway for Visualisation To our WEB API, we will use a WAGO PLC with an EWON communication gateway. The data accessed from the WAGO PLC, which has an embedded OPC UA server, make the Tags available to the EWON communication interface,

Table 1. Packet capture of device gateway.



Date	Time	Direction	ID	Address	Port	Hex Dump
5/14/2021	10:04:51 AM	IN	2	41.13.196.229	45708	08780200000081ff
5/14/2021	10:04:51 AM	OUT	2	41.13.196.229	45708	08f80200000051dd
5/14/2021	10:04:58 AM	IN	7	41.13.204.231	38968	087807000000c443
5/14/2021	10:04:58 AM	OUT	7	41.13.204.231	38968	08f8070000001461
5/14/2021	10:05:26 AM	IN	3	41.13.254.98	47490	0878030000003589
5/14/2021	10:05:26 AM	OUT	3	41.13.254.98	47490	08f803000000e5ab
5/14/2021	10:05:43 AM	IN	1	41.13.208.106	33612	0878010000005d64
5/14/2021	10:05:43 AM	OUT	1	41.13.208.106	33612	08f8010000008d46
5/14/2021	10:05:53 AM	IN	2	41.13.196.229	46236	08780200000081ff
5/14/2021	10:05:53	OUT	2	41.13.196.229	46236	08f80200000051dd

which exposes these tags using MQTT (Profanter et al., 2019). Not many PLCs presently have an embedded OPC UA server built into the process control system (Power and Kominek, 2013). Usually, any framework can be a client to this OPC UA data. Still, for this design, the data were parsed to the Gateway using EWON communication gateway where the EWON became the client, and the data were parsed to the CoCT gateway using MQTT in JSON format.

The device gateway

The purpose of the device gateway is to have a gateway could be a software communication interface to different data.

sources and data formats. The function of the device gateway is to process data from various data sources and make it available to the webserver/MES. This is achieved by having a data structure tied to the format of the UDP data frame. See Table 1. Figure 6 shows the database schema of the device gateway.

At the Gateway, there are two stages, and the first stage is to de-serialise the data, i.e. function readPacketDetails(buf). The second phase is to interpret the payload.

As seen from the Gateway Packet format demonstrated in Table 2 we can differentiate from 16 different data sources based on the 4-bit type field of

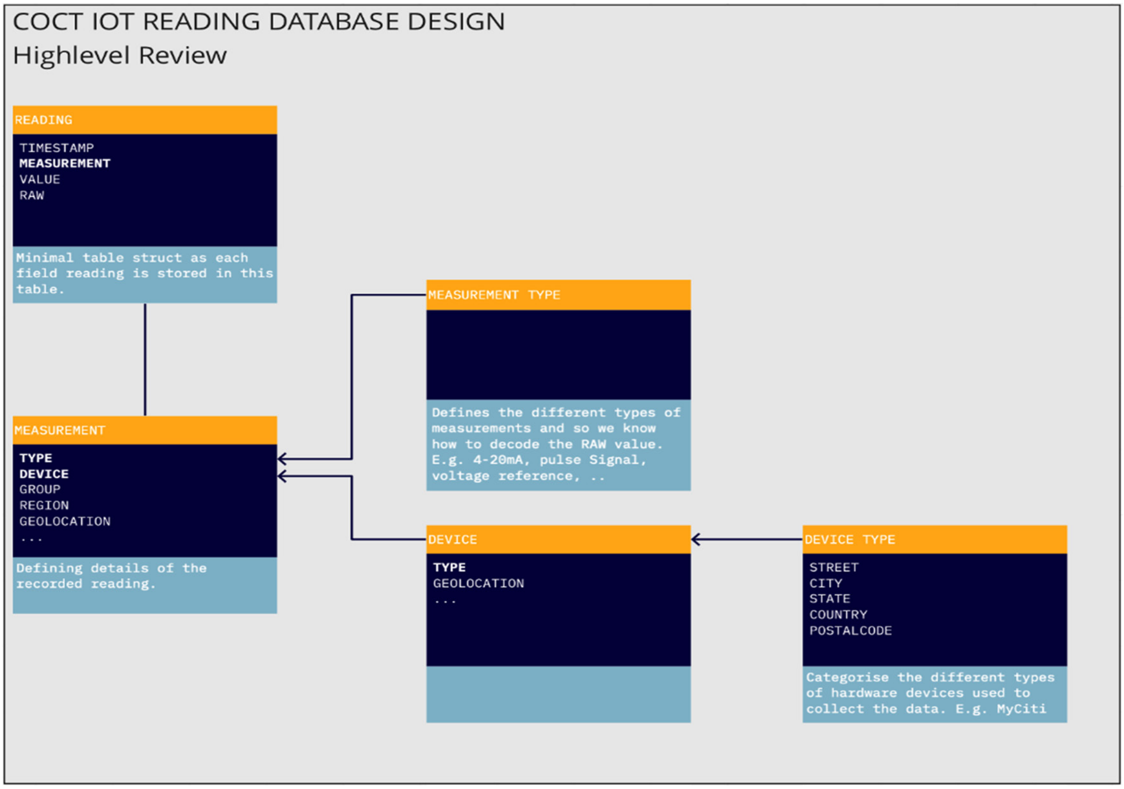
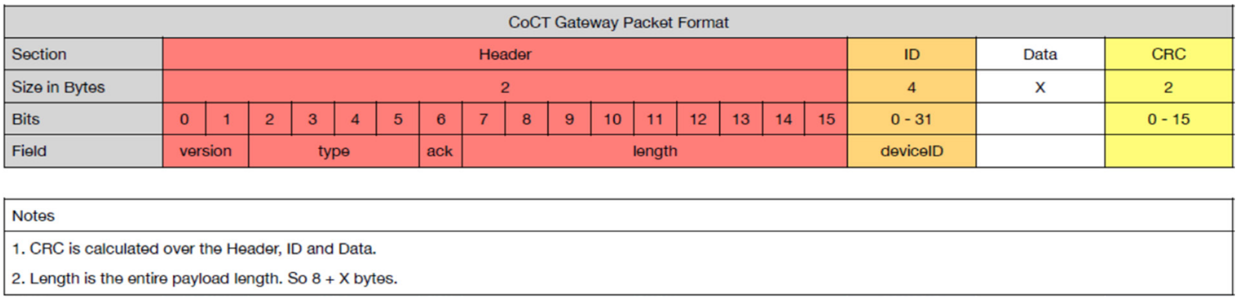

COCT IOT READING DATABASE DESIGN
Highlevel Review
READING
TIMESTAMP
MEASUREMENT
VALUE
RAW
Minimal table struct as each field reading is stored in this table.
MEASUREMENT TYPE
Defines the different types of measurements and so we know how to decode the RAW value. E.g. 4-20mA, pulse Signal, voltage reference, ..
MEASUREMENT
TYPE
DEVICE
GROUP
REGION
GEOLOCATION
...
Defining details of the recorded reading.
DEVICE
TYPE
GEOLOCATION
...
DEVICE TYPE
STREET
CITY
STATE
COUNTRY
POSTALCODE
Categorise the different types of hardware devices used to collect the data. E.g. MyCiti

Figure 6: Database mapping of data received from device gateway.

Table 2. Gateway packet format.



CoCT Gateway Packet Format																			
Section	Header																ID	Data	CRC
Size in Bytes	2																4	X	2
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0 - 31		0 - 15
Field	version		type				ack	length									deviceID		

Notes
1. CRC is calculated over the Header, ID and Data.
2. Length is the entire payload length. So 8 + X bytes.

the data packet received. The device gateway will de-serialise according to the type field received in the UDP packet. So each type will have its own de-serialising routine/function.

The device ID is another differentiator which the gateway server can use to process data. Apart from using this field to assign a unique address to the device transmitting the data in the case of hardware type (gsm_logger_io module) we also used it to parse different KPI values from the SQL Data source to the device gateway.

For data where the data source is SQL data source where the data is aggregated by data engine What was essentially done was that we assigned a unique device ID to specific KPI together with type field, where the device Gateway knows that it needs to treat device ID as KPI data and populate the database accordingly based on type field.

The device gateway will handle the device ID as KPI' populating the relevant table in the webserver's database. Bit no 2 to 5 represents type (data source), So if the type bits represent SQL data source, then specifically, the device gateway knows that device ID will represent a specific KPI. For this proof of concept device ID, 0 to 255 were reserved for SQL data source.

For example: if Belt press availability is a KPI required. These data are sitting in the local plant database as belt press running status. The engine will aggregate these data by counting the number of belt presses running. The result is made available as key/value pair, where the key is the enumerated data type from 0 to 255 and value is the actual aggregated value representing that specific KPI. The key is then the device ID that the Device Gateway is expecting. The value is the payload(data that the Gateway is expecting).

On the other side, these enums are associated with specific KPIs on the device gateway.

Looking at Table 2 it can be seen that the field named type is what the device gateway uses to differentiate the different data sources and thus will de-serialise the payload data according to this field. The device gateway is a software engine that receives UDP packets, performs a few checks and balances(packet manager and de-serialises the data and populates the necessary fields in the database.

Understanding, comparing implementing the protocols

The three protocols are discussed briefly because parsing data to the remote system is essential to understanding the complete data value chain. Solutions, especially in the Industrial Internet of Things(IIoT) spaces, require a mix of protocols but using a suitable protocol at the right layer/tier of the data value chain. Some of the critical decision-makers in choosing the most appropriate protocol often come down to how lightweight the protocol is, the network's reliability, the layer's purpose, the layer of security required, and the controllability expected layer (Wollschlaeger et al., 2017). We discuss these very relevant protocols relevant in the IIoT space as well the OT environment.

OPC UA

OPC Unified architecture is a connectivity framework that introduces a platform independence framework/architecture. OPC UA completely objects to orientation. OPC UA = IEC 62541. OPC UA is based on a client/server architecture that uses TCP/IP and HTTP/SOAP as the underlying frameworks (Cavalieri et al., 2017). It is important to understand that OPC UA is an architecture and not a Protocol on its own. It uses different protocols to complete its framework/

architecture. In layman's terms, data are packed in a certain way, so the other side, when it sees this data-packed in this "sequence", recognises this as OPC UA and can de-serialise this packed data according to the framework/architecture rules.

One of the key objectives is device interoperability independent of propriety protocols and API of device manufacturers. OPC UA was initially intended or had its roots in the factory automation of Large manufacturing plants within a LAN environment (Drahos et al., 2018). This protocol is quite heavy-weight in its make-up, and this is quite understandable due to its origins. The protocol has been implemented in various environments and is gaining considerable.

ground in its implementation in different environments. End devices will publish or render data to API's standard architecture that API can unpack without any datacasting. The addition of a" browseable" built-in information model includes executable services like read, write, method-call, subscribe, etc.

OPC UA essentially converts the hardware protocol of the device into a standardised device model. One of the modern evolutions of the OPC UA protocols is embedded OPC UA, which essentially means the OPC UA server is embedded in the SOC, and its tags can be exposed to any high-level client. PLC's like WAGO are presently using embedded OPC UA servers to Expose tags to high-level systems (Tel, 2012).

Message queuing telemetry transport (MQTT)

In the IoT space, lightweight and straightforward protocols are gaining ground, so if the requirement is to bypass complexity and move to a small footprint solution that guarantees a secure and reliable data exchange in industrial automation, you are bound to come across MQTT.

MQTT is messaging protocol, not a data communications protocol; it does not specify a particular format for the payload data. The data are determined by each client connecting. In the case of MQTT, the publisher and subscriber need to agree in the format in advance, or they will not have any clue what the payload means. MQTT is explicitly not interoperable but is used in industrial applications for lightweight data transfer.

MQTT is a lightweight protocol based on IP used by mainly IoT platforms. MQTT has PUB/SUB architecture (O. P. C. U., 2018).

Connection is always initiated by the client to the broker using port 1883 or 8883 for secure/encrypted,

The broker is usually publically accessible via the internet and acts as a communication bridge or central node/point between the different clients. There are, of course, options to install the broker locally without public access. Regarding the content of the message, MQTT does not care (Tao et al., 2014).

MQTT allows an unlimited number of clients/subscribers to listen to a published topic. It is up to the clients and broker to agree on the message format. The basic understanding of the MQTT architecture translates to the publisher publishes a topic, and whoever listens to that topic can see the message content.

Table 3 provides a snapshot comparison of the three protocols related to their applicability in different environments regarding data transfer.

Table 3. Snapshot summary of comparison of protocols.

Description	OPC UA	MQTT	COAP
Protocol binding	TCP	TCP	UDP
	Server/ client	Pub/sub	Server/ client
Security	Excellent	Good	Fair
Reliability over networks	Only good over very stable networks with excellent bandwidths	Suitable for transferring data/ commands over unstable connections	Ideal for client/server connections Preferred for stable networks
Architecture	Client-server model	Broker is the centre of the network	Node is the centre of the network
Controllability of nodes	Offers secure controllability.	No controllability built into the protocol	Offers controllability of nodes within the protocol

Constrained application protocol (COAP)

Constrained Application protocol as defined by the RFC7252 standard is an open IoT standard. The protocol is completely asynchronous, which is essentially not connection orientated, making it very efficient for web applications.

COAP was designed to support the RESTFUL protocol synonymous with GET, PUT, POST, and DELETE verbs (Durkop et al., 2015).

COAP usually is associated with ports 5683 and 5684. COAP implements UDP binding and supports Uniform Resource Identifier(URI). COAP also 4byte binary header.

Results

So in principle, the device gateway was reading data from four different data sources, namely:

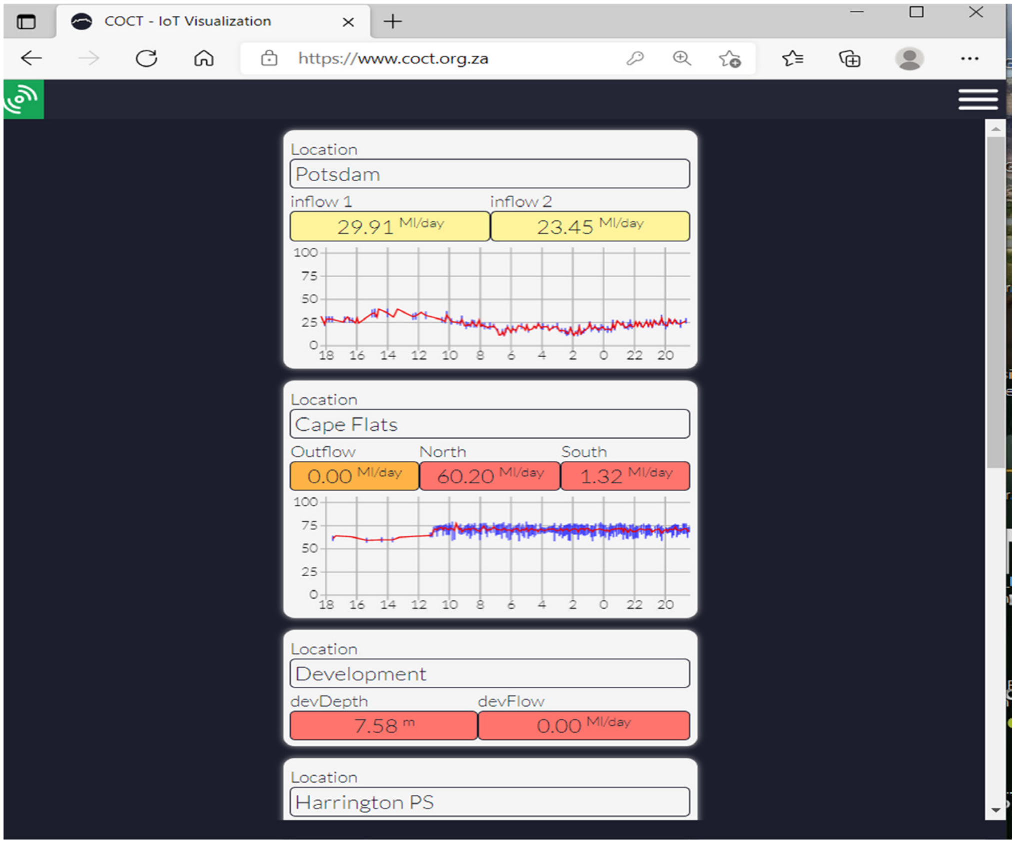
i. Standalone LTE data logger.
ii. LoRa WAN data logger.
iii. Local Plant SQL data engine.
iv. WAGO PLC using EWON Communication Gateway.

Figure 7 is a snapshot of a web page rendered showing visualisation of devices from 2 wastewater treatment plants, namely Potsdam and Cape Flats waste Water treatment plants.

Figure 6 shows how the type is one of the differentiators in the database schema. The LTE logger, LoRa WAN (Gupta and Zyl, 2021a, b; Balyan, 2020), and SQL data engines represent the device gateway's different types as a differentiator. The MES or web server has no idea what the specific data source is. All it queries is get method () based on message type that gets rendered as de-serialised payload. All the data being parsed to the device gateway is UDP.

Conclusions

We successfully parsed all these data sources to a standard device gateway, negating the need for expensive visualisation software like a Modern Propriety SCADA system. It also negates the need



Location: Potsdam
inflow 1: 29.91 Ml/day
inflow 2: 23.45 Ml/day

Location: Cape Flats
Outflow: 0.00 Ml/day
North: 60.20 Ml/day
South: 1.32 Ml/day

Location: Development
devDepth: 7.58 m
devFlow: 0.00 Ml/day

Location: Harrington PS

Figure 7: Web API visualisation.

for expensive propriety end devices in the form of off-the-shelf data loggers.

In the IoT space, where microcontrollers are cheaper by the day, and libraries for implementing these decoders are also freely available from different open-source platforms, solutions of this nature are readily implemented by engineering personnel with minimum coding/software development experience.

The other realisation in this research is how easily we can integrate technologies to Web APIs and MES systems for providing data for end-users and data analytics using JSON and other frameworks.

One of the other pertinent realisations of this paper is how close the OT and IT worlds have converged in data exchange. With protocols like OPC UA, MQTT, and COAP being used extensively and interchangeably, who knows ? There might be an open-source protocol combining these protocols to let us have one with all the best traits of each of these protocols.

Acknowledgements

Chad Venables of RIOTware for developing the Frontend Web API to visualise the data received from the device gateway.

Literature Cited

Artuso, C. and Palladino, P. 2019. Long-term memory effects on working memory updating development. PLOS ONE 14(5): e0217697. Available at: https://doi.org/10.1371/journal.pone.0217697.

Azhari, S. J. and Kaabi, H. 2001. A novel data compression technique for 420 Ma current loop transmitters. *International Journal of Engineering* 14(1): 35–40.

Balyan, V. 2020. Outage probability of cognitive radio network utilizing non orthogonal multiple access. *7th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, pp. 751–755.

Bassi, L. 2017. Industry 4.0: Hope, hype or revolution? *RTSI 2017—IEEE 3rd International Forum on Research and Technologies for Society and Industry, Conference Proceedings*, doi: 10.1109/RTSI.2017.8065927.

Calderón Godoy, A. J. and Pérez, I. G. 2018. Integration of sensor and actuator networks and the SCADA system to promote the migration of the legacy flexible manufacturing system towards the industry 4.0 concept. *Journal of Sensor and Actuator Networks* 7 (2). doi: 10.3390/jsan7020023.

Cavalieri, S., Di Stefano, D., Salafia, M. G. and Scroppo, M. S. 2017. *OPC UA integration into the web. Proceedings IECON 2017—43rd Annual Conference of the IEEE Industrial Electronics Society*, doi: 10.1109/IECON.2017.8216590.

Durkop, L., Czybik, B. and Jasperneite., J. 2015. Performance evaluation of M2M protocols over cellular networks in a lab environment. *18th International Conference on Intelligence in Next Generation Networks, ICIN 2015*, pp. 70–75, doi: 10.1109/ICIN.2015.7073809.

Drahos, P., Kucera, E., Haffner, O. and Klimo, I. 2018. Trends in industrial communication and OPC UA. *Proceedings of the 29th International Conference on Cybernetics and Informatics, K and I 2018*, doi: 10.1109/CYBERI.2018.8337560.

Gupta, G. and Zyl, R. V. 2021a. Energy harvested end nodes and performance improvement of LoRa networks. *International Journal on Smart Sensing and Intelligent Systems* 14(1): 1–15.

Gupta, G. and Zyl, R. V. 2021b. NOMA-based LPWA networks. *Expert Clouds And Applications. Lecture Notes in Networks and Systems* Vol. 209. Springer, Singapore, pp. 523–530. Available at: https://doi.org/10.1007/978-981-16-2126-0_42.

Katti, B., Plociennik, C., Ruskowski, M. and Schweitzer, M. 2018. SA-OPC-UA: Introducing semantics to OPC-UA application methods. *IEEE International Conference on Automation Science and Engineering*, doi: 10.1109/COASE.2018.8560467.

Koon, J. 2020. LoRaWAN empowers very low-power, wireless applications. Tech Idea Res., vol. 1.0.

Muller, M., Wings, E. and Bergmann, L. 2017. Developing open source cyber-physical systems for service-oriented architectures using OPC UA. *Proceedings—2017 IEEE 15th International Conference on Industrial Informatics*, INDIN 2017, doi: 10.1109/INDIN.2017.8104751.

O. P. C. U. 2018. Architecture. *"IoT,"* June, pp. 1–48. Available at: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf.

Orange 2016. LoRa Device Developer Guide. p. 42 [Online]. Available at: https://partner.orange.com/wp-content/uploads/2016/04/LoRa-Device-Developer-Guide-Orange.pdf.

Power, L. and Kominek, D. 2013. Keys to developing an embedded UA server. Matrikon Opc, pp. 1–7. Available at: https://opcfoundation.org/wp-content/uploads/2015/03/Keys-To-Developing-an-Embedded-UA-Server_Whitepaper_EN.pdf.

Profanter, S., Tekat, A., Dorofeev, K., Rickert, M. and Knoll, A. 2019. OPC UA versus ROS, DDS, and MQTT: Performance evaluation of industry 4.0 protocols. *Proceedings of the IEEE International Conference on Industrial Technology* vol. 2019-February, doi: 10.1109/ICIT.2019.8755050.

Schleipen, M., Gilani, S. S., Bischoff, T. and Pfrommer, J. 2016. OPC UA & Industrie 4.—enabling technology with high diversity and variability. *Procedia CIRP* vol. 57, doi: 10.1016/j.procir.2016.11.055.

Silveira Rocha, M., Serpa Sestito, G., Luis Dias, A., Celso Turcato, A. and Brandao, D. 2018. Performance comparison between OPC UA and MQTT for data exchange. *2018 Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2018—Proceedings*, doi: 10.1109/METROI4.2018.8428342.

Tao, F., Zuo, Y., Da Xu, L. and Zhang, L. 2014. IoT-Based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Transactions on Industrial Informatics*, doi: 10.1109/TII.2014.2306397.

Tel, G. 2012. Communication protocols. *Introduction to Distributed Algorithms*, Cambridge, pp. 74–102. Available at: https://ki.pwr.edu.pl/lemiesz/info/Tel.pdf.

Yongde, Z., et al. 2014. The controller development of multi-layer parking equipment based on STM32. *International Journal of Smart Home* 8(1): 303–310.

Wollschlaeger, M., Sauter, T. and Jasperneite, J. 2017. The future of industrial communication: automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, doi: 10.1109/MIE.2017.2649104.