Application Note Part 2

Simplifying Connectivity for Predictive Maintenance

To achieve optimized results for predictive maintenance, it is critical to acquire data from diverse data sources to increase the accuracy and precision of data

Predictive maintenance enables operators to predict when maintenance should be performed by determining the condition of their in-service equipment. Basically, predictive maintenance comes down to keeping equipment in good working order to prevent unexpected downtime, thus ensuring reliability. This practice brings huge cost savings in comparison with routine or scheduled preventive maintenance, because tasks are performed only when needed. **ARC Advisory Group** estimates that predictive maintenance can reduce maintenance costs by 50% and unexpected failures by 55%.

To further increase the efficiency of predictive maintenance, it is critical to leverage the ability of edge computers to preprocess increasing volumes of data acquired from sensors, meters, and other network devices, as well as to autonomously react before machine failures occur.

However, with new opportunities come new challenges, and the same holds true for enabling predictive maintenance. Two main challenges that managers have to deal with are performing diverse data acquisition and deploying edge intelligence.

Case in Brief:

Connecting Data From Sensors to AIoT Systems With Ease

KPMG, a distinguished global firm providing audit, tax, and advisory services, leveraged AIoT (AI and IoT) technology to help an automotive engine parts manufacturer to increase yield and build predictive maintenance. More sensors were added to existing IoT devices to collect additional data on vibration, temperature, rotating speed, and electric current. Moxa's easy-to-use connectivity solutions were implemented to help send the data to a backend AI platform where, through analysis, control standards were established, making predictive maintenance possible as any deviation, which could result in the production of defective products, was immediately detected. The OEE was increased from 70% to 85%. **Learn More**

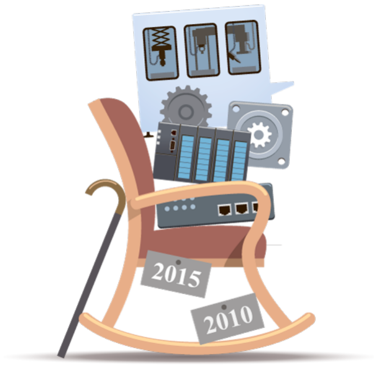
Connectivity Challenge #2.1

Diverse Data Acquisition Causes Complex Connectivity

In order for predictive maintenance to deliver the required results, manufacturers need more and diverse data to unlock insights. The obvious route is to acquire more big data by adding more sensors and devices to legacy machines for analysis and intelligence software development in the cloud or IT systems. However, we know the interfaces and protocols of legacy machines are varied. For example, barcode scanners and displays use RS-232, RS-422, or RS-485 serial interfaces while tower lights and signal lights use analog and digital I/O interfaces. Furthermore, industrial standard protocols, such as Modbus, Ethernet/IP, and PROFIBUS, can also be employed at the same time in a network. So in order to collect data, our customer tended to just use familiar PLCs or communication modules to connect devices of different proprietary protocols, as well as different analog and digital I/O interfaces, installing edge gateways to covert industrial standard protocols to online SCADA, cloud, or IT systems in the absence of cloud communication capabilities.

However, we know that PLCs are assigned for control jobs such as storing procedures, sequential or position control, time counting, and input or output control. The additional data acquisition and edge-to-cloud protocol conversion jobs maybe minor in the case of just one or two points, but will bring major challenges to the fore in managing diverse connectivity on a large scale.

Cost Issue

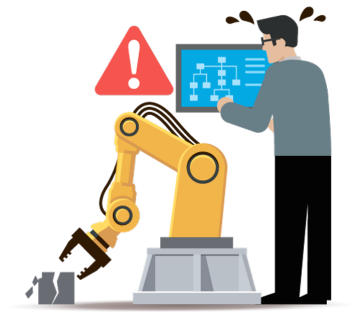


First, it is a cost issue as the need for additional PLC modules for protocol converters and I/Os can be prohibitively expensive. In addition, legacy PLCs may not have the capability to communicate in cloud protocols such as MQTT or AMQP.

Time Issue

Second, there is the time issue as extra configuration and programming efforts are required to realize edge-to-cloud connectivity from scratch, and they have often proven to be time-consuming in large deployments.

Troubleshooting Issue



Then, adding to engineers' frustration is the issue of troubleshooting, as it is also very difficult and time-consuming to pinpoint all the communication issues caused by incorrect software parameters, such as slave IDs and register addresses, or incorrect command configurations in large-scale networks.

WHAT IF

We Can Simplify Your Diverse Data Acquisition?

Moxa offers ready-to-run connectivity solutions that easily convert protocols of multiple field devices, including serial, I/O, Modbus, and EtherNet/IP, to MQTT, the most widely used protocol for cloud connectivity, in order to communicate with intelligent software and systems in private and public cloud platforms, such as Azure and Alibaba Cloud. It only takes a few steps to complete cloud selection, connection and message tags settings, and data mapping between the field and cloud with our intuitive UIs.

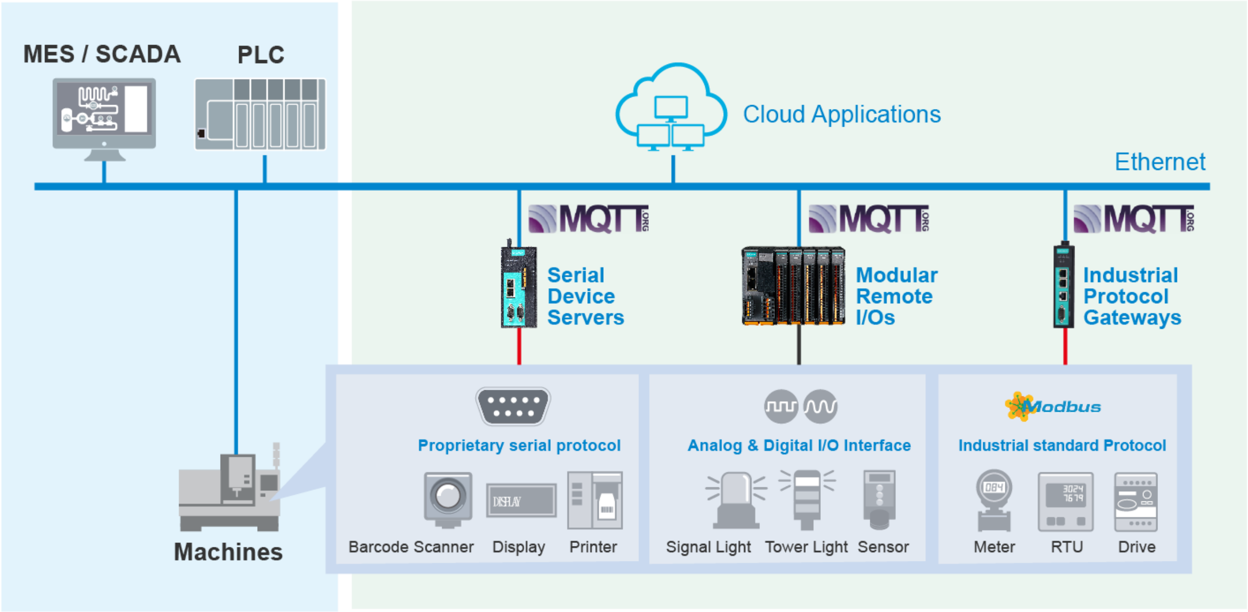


Figure 1: Ready-to-run connectivity solution from serial, I/O, and standard protocols to the cloud

- The MQTT-ready NPort serial device servers easily connect serial to MQTT/Azure/Alibaba Cloud. Its diagnostics tool ensures the accuracy of settings while connection loss buffer function can avoid packet loss when the cloud connection is disconnected.
- The ioThinx Series, Moxa's MQTT-ready advanced modular Remote I/O, supports I/O-to-IT/OT protocol conversion, such as Modbus TCP for OT engineers, as well as SNMP and RESTful API for IT engineers. Safe mode function allows you to react instantly to anomalies with preprogrammed safety protocols.
- Moxa's MQTT-ready MGate industrial protocol gateways simply convert protocols such as Modbus and EtherNet/IP to MQTT/Azure/Alibaba Cloud. Its embedded traffic monitoring and diagnostic information functions and convenient web-console make troubleshooting easy.

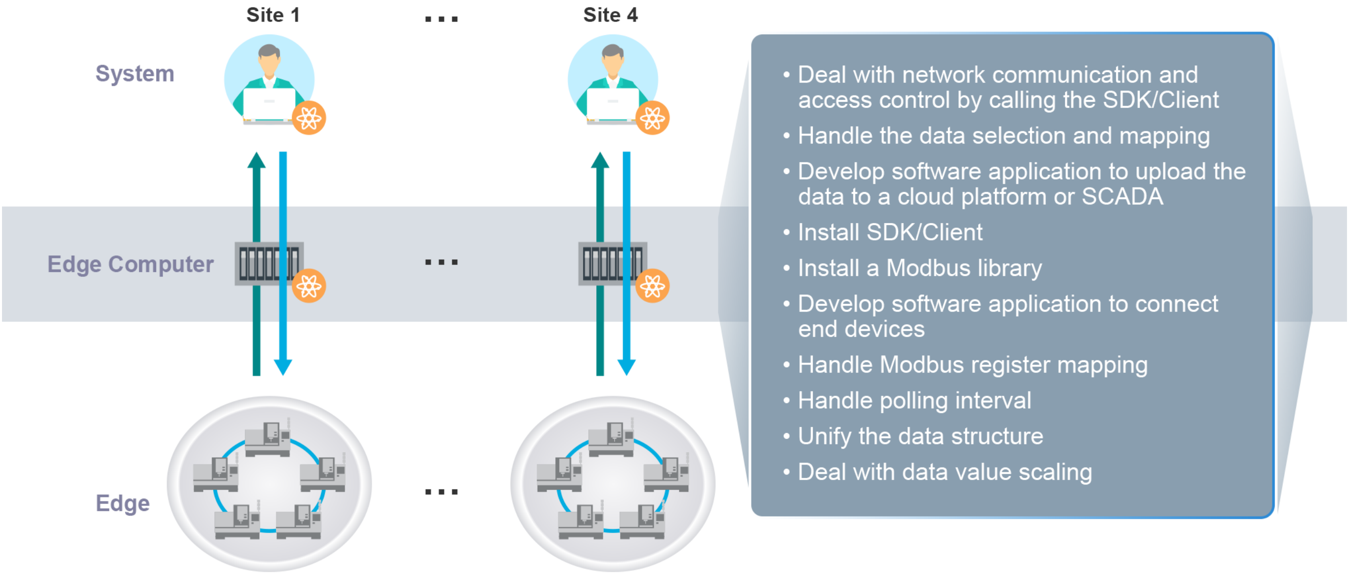
Connectivity Challenge #2.2

Deploying Edge Intelligence Is Hard to Start With

Sending all of your device data to a system for processing and analytics can take anywhere between a few minutes to several hours at a stretch. For example, in large-scale deployment scenarios, your IoT devices could generate more than one terabyte (TB) of data per day, and that could take you a while to transfer this data to the system, process it, and generate actionable items. However, in industrial applications, critical actions, such as alert, emergency shutdown, human safety protection, need immediate action at a site. In addition, transferring large volumes of data from the edge of the network to a public cloud server can be very expensive.

Deploying intelligent software in edge computers can help preprocess the data from industrial sensors or devices by analyzing and filtering the data to save network and computing resources. In addition, it can decrease the latency of local responses to ensure the reliability and agility of interconnected applications, as well as the dependency to MES or Manufacturing Intelligent (MI) software. Thus, the MES and MI software could better coordinate the process and better handle the big data from various sources. As the trend to a more connected future moves forward, the demand for edge or fog computing will increase.

Usually, deploying edge computing in multiple sites for big data preprocessing and intelligence deployment is an uphill task. For example, if you want to deploy an edge computer in one site to collect Modbus data from meters and send it to the cloud, you have to go through the following programming efforts to convert OT and IT protocols in order to connect OT data to the cloud for analysis, and then deploy and redeploy intelligence and rule engines to the edge to optimize operations.



- Deal with network communication and access control by calling the SDK/Client
- Handle the data selection and mapping
- Develop software application to upload the data to a cloud platform or SCADA
- Install SDK/Client
- Install a Modbus library
- Develop software application to connect end devices
- Handle Modbus register mapping
- Handle polling interval
- Unify the data structure
- Deal with data value scaling

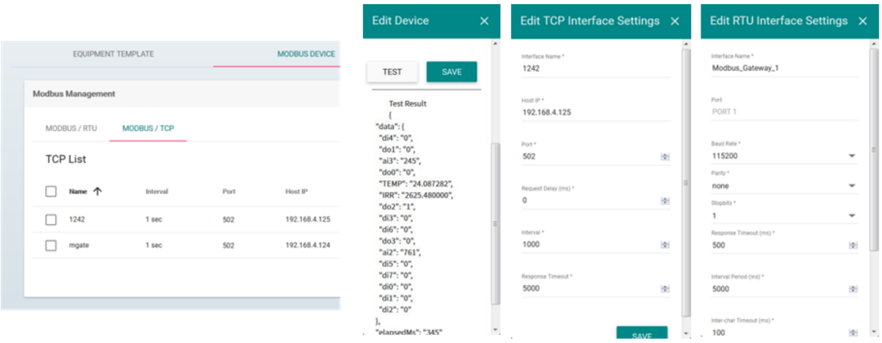
Figure 2: A complex process to deploy edge computing

In large-scale networks, deploying edge connectivity is very time-consuming and requires a considerable task force. It is a daunting task for OT engineers to finish hundreds or even thousands of IIoT gateway settings.

WHAT IF

We Can Simplify Your Large-Scale Edge Computing Deployment?

When a gateway-involved IIoT project goes through POC and moves to the implementation stage, it often involves hundreds of gateways being deployed at multiple field sites across a large area. Thus, choosing a cloud-ready IIoT gateway that can simplify mass device deployment definitely has numerous advantages. Moxa's cloud-ready IIoT Edge Gateways can simplify your data acquisition configuration with their intuitive web GUI, which allows an IT or OT expert to manage IT or OT protocols without the need for any additional programming. Our gateways also enable mass configuration that makes large-scale deployment significantly faster than manually inputting data. What's more, the risk of human error, which most certainly will occur when data is input manually on hundreds of IIoT gateways, is eliminated. Furthermore, the IIoT Edge gateway allows network operators to install intelligence from the cloud to the edge. This enables them to react promptly before production gets interrupted.



TCP List

□	Name ↑	Interval	Port	Host IP
□	1242	1 sec	502	192.168.4.125
□	mgate	1 sec	502	192.168.4.124

```
Test Result
{
"data": {
 "di4": "0",
 "do1": "0",
 "ai3": "245",
 "do0": "0",
 "TEMP": "24.087282",
 "IRR": "2625.480000",
 "do2": "1",
 "di3": "0",
 "di6": "0",
 "do3": "0",
 "ai2": "761",
 "di5": "0",
 "di7": "0",
 "di0": "0",
 "di1": "0",
 "di2": "0"
},
"elapsedMs": "345"
```

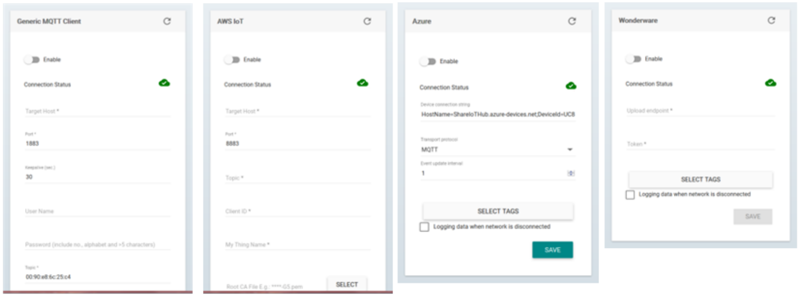
Edit TCP Interface Settings

Interface Name * 1242
Host IP * 192.168.4.125
Port * 502
Request Delay (ms) * 0
Interval * 1000
Response Timeout * 5000

Edit RTU Interface Settings

Interface Name * Modbus_Gateway_1
Port PORT 1
Baud Rate * 115200
Parity * none
Stopbits * 1
Response Timeout (ms) * 500
Interval Period (ms) * 5000
Inter-char Timeout (ms) * 100

Figure 3: Easy configuration for Modbus and Modbus/TCP edge devices with web GUI


Generic MQTT Client

Target Host *
Port * 1883
Keepalive (sec.) 30
User Name
Password (include no., alphabet and >5 characters)
Topic * 00:90:e8:6c:25:c4

AWS IoT

Target Host *
Port * 8883
Topic *
Client ID *
My Thing Name *
Root CA File E.g.: ****-G5.pem

Azure

Device connection string HostName=ShareIoTHub.azure-devices.net;DeviceId=UC8
Transport protocol MQTT
Event update interval 1

□ Logging data when network is disconnected

Wonderware

Upload endpoint *
Token *

□ Logging data when network is disconnected

Figure 4: Ready-to-run cloud connectivity to AWS, Azure, etc.