White paper
Cisco public

Translating Non-native Industrial Protocols on the Cisco Catalyst Industrial Routers

Contents

Use edge compute and Cisco IOx on the industrial routers to translate non-native protocols to IP, with Modbus as a use case.

Deploy a Modbus RTU sensor or PLC with under 10 lines of code

Written for industrial control automation and Programmable Logic Controller (PLC) integration with Cisco's industrial routing product lines, this white paper outlines how to approach Modbus agent implementation on the Cisco Catalyst™ industrial IoT routers and send data over IP networks to northbound services and the cloud. It offers a template for other non-native protocols that cannot be converted to IP using the standard IOS-XE and command-Line Interface (CLI) configuration.

Introduction

The Modbus protocol is used across multiple industries. Devices such as Remote Terminal Units (RTUs), PLCs, and sensors have been known to use the Modbus client/server architecture, either over serial (Modbus over serial) interfaces (RS-232, RS-485) or over TCP/IP (Modbus TCP). While Cisco IOS® XE does not have a Modbus implementation, it is quite easy to use edge compute and Cisco® IOx (application hosting) on Cisco Catalyst™ industrial routers to meet your needs.

How to get started

There are two ways to send Modbus over serial packets over an IP network. You can either use raw socket encapsulation to transport Modbus RTU serial data over an IP network OR use the Cisco IOx application to convert Modbus RTU over serial packets to an IP-based protocol, such as Modbus TCP, MQTT, etc.

This white paper discusses how to connect Modbus serial devices to Cisco Internet of Things (IoT) routers and leverage IOx and edge compute capabilities to make Modbus serial RTU data friendly to IP networks, such as a Grafana dashboard, an MQTT broker, or another application.

Note: We will not be looking at the Modbus TCP variant, as that does not need any configurations on the routers other than making sure the IP interfaces are configured correctly to send data from the Modbus TCP client to a server over a TCP/IP network.

What is Modbus and why is it popular?

Modbus is a communications protocol[1] used in industrial and building automation and is the most commonly available means of connecting automated electronic devices. It is also used to connect sensors over the bus, using serial transmission protocols for efficiency.

The creators of Modbus did not keep the standard proprietary, rightly assuming that it would be best for everyone. Because of this, Modbus became the first widely accepted fieldbus standard. In a brief time, hundreds of vendors implemented the Modbus messaging system in their devices.

The transport layer for Modbus RTU commands is also simple to understand. Although Modbus packets may be sent over RS-232 serial links, most RTU devices use RS-485, a differential communications standard that supports up to 32 nodes in a multi-dropped bus configuration. RS-485 supplies noise immunity superior to that of the RS-232 electrical standard. Modbus implements a remarkably simple data representation. Its primary purpose is simply to move data between an RTU server device and an RTU client device. There are only two kinds of data to move, registers and coils. Registers are 16-bit unsigned integers. Coils are single bits.

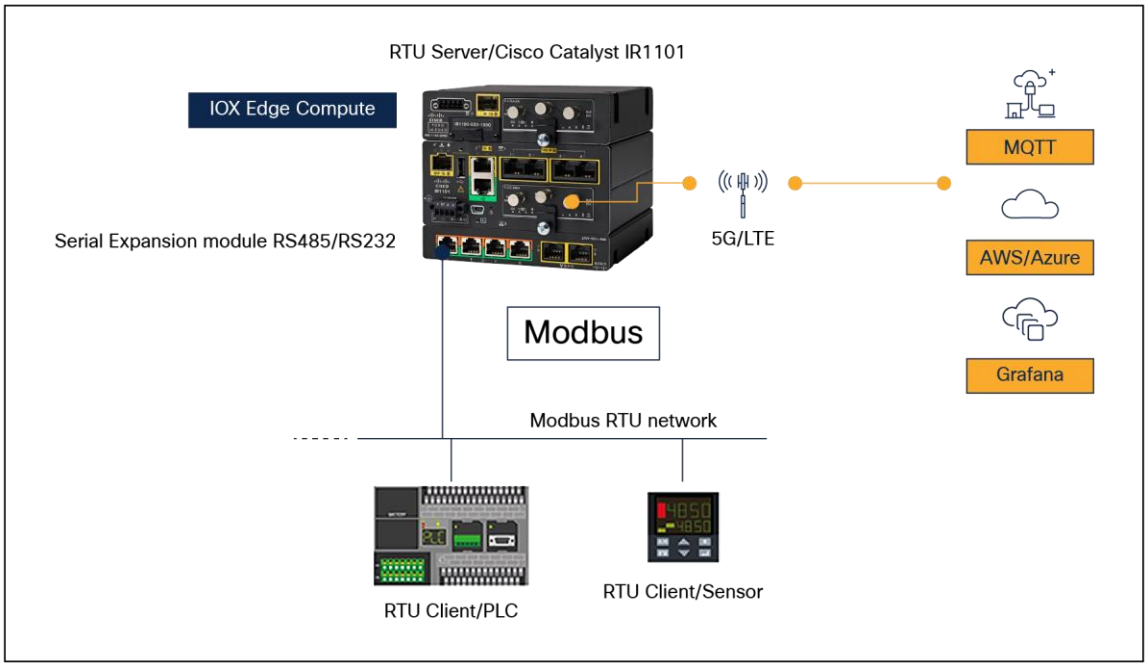


Figure 1.
A Modbus RTU network with the Catalyst IR1101 using an RS-485 serial port and IOx

Modbus implements an uncomplicated request–response command structure. A Modbus server requests or sends data to a client, and the client responds. There are simple commands to read a register, read a coil, write a register, and write a coil.

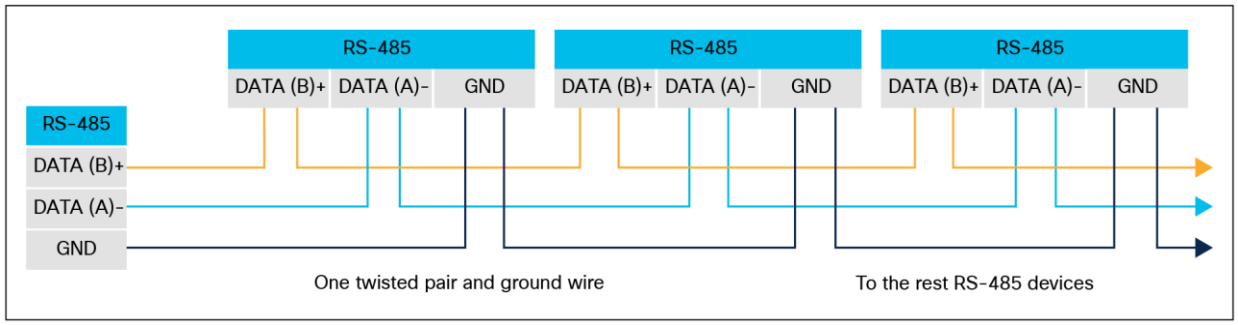
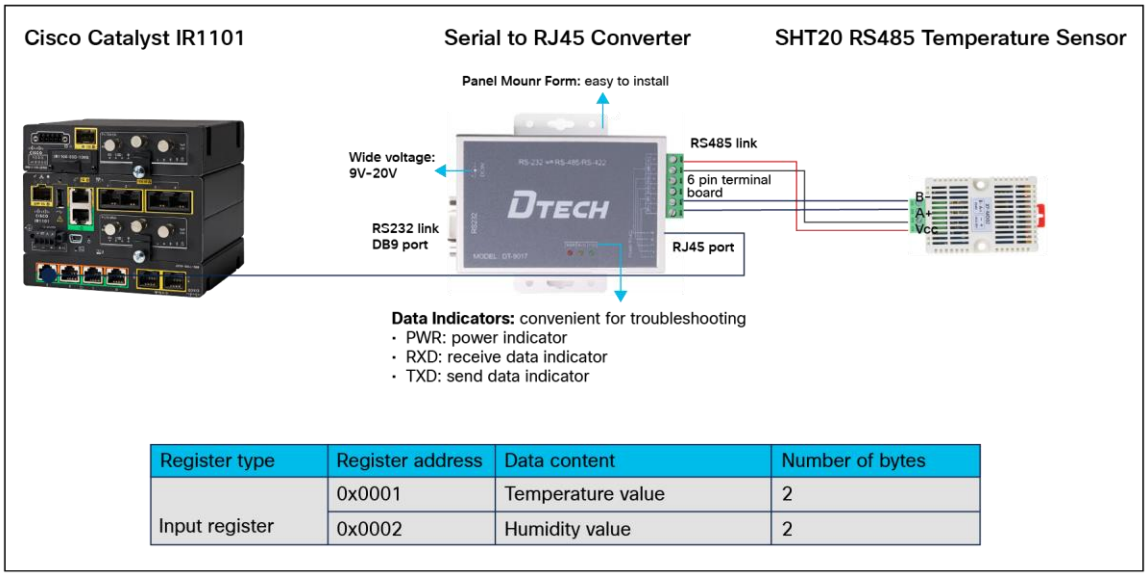


Figure 2.
Modbus architecture, server-to-client communication

The RS-485 protocol uses data terminals A- and B+ for data, which is typical for a Modbus sensor, PLC, or other client device. As shown in the figure below, the temperature sensor uses A- and B+ terminals going to the serial-to-RJ-45 converter. Only two wires and a common ground are needed for the RS-485, which employs differential signaling. Differential signals work by connecting one wire to the signal and the other wire to the inverse of the signal. As noise tends to couple into both lines equally and thus cancels out at the receiving end, this enhances the signal's noise tolerance and ability to recover the signal at the far end of the cable. Refer to the Connecting and Troubleshooting IoT Devices Using Asynchronous Serial Interfaces white paper (https://www.cisco.com/c/en/us/products/collateral/routers/829-industrial-router/white-paper-c11-743674.html) for more examples on how to convert serial-to-RJ-45 connectors and serial pinouts.



Register type	Register address	Data content	Number of bytes
Input register	0x0001	Temperature value	2
	0x0002	Humidity value	2

Figure 3.
Example implementation using the SHT20 temperature sensor

IOx and IR1101 configuration

We will use Cisco IOx to translate Modbus packets for consumption on IP networks. Cisco IOx is an application hosting environment[2] that is used by organizations ranging from manufacturing and energy corporations to public sector entities such as cities and transportation authorities that use IoT technologies to produce effective business outcomes. Cisco IOx allows you to execute IoT applications at the edge with secure connectivity, using Cisco IOS XE software to deliver powerful services for rapid, reliable integration with IoT sensors and the cloud. In the case of the Modbus sensor application, we are aiming to deploy a very lightweight Docker application that can use "prebuilt" packages, such as the Python Modbus Package (minimalmodbus), to communicate with Modbus clients.

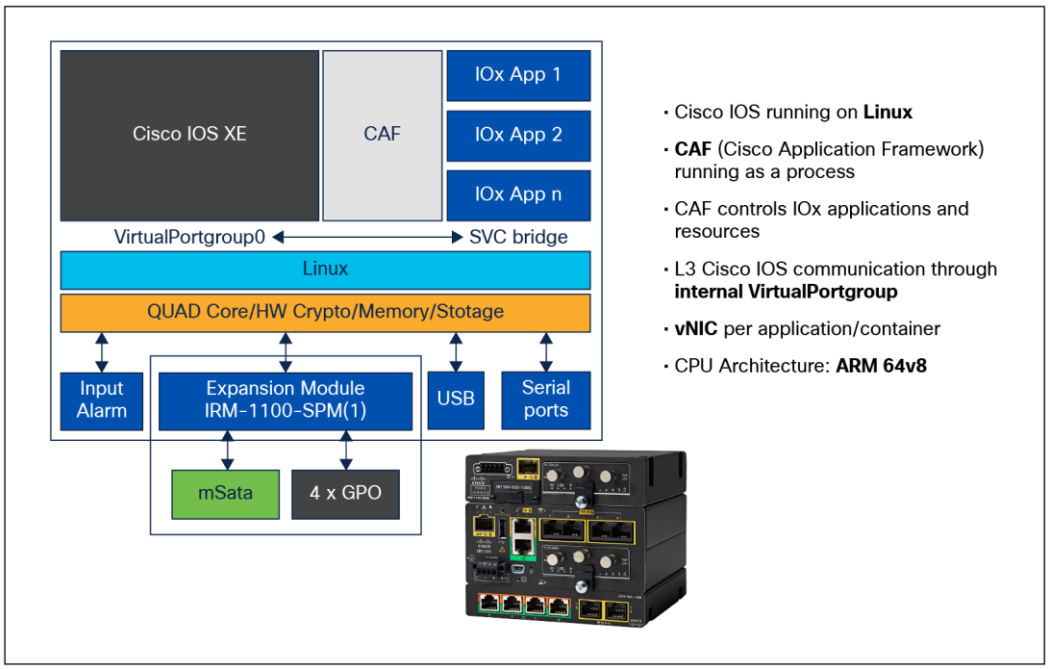


Figure 4.
IOx architecture on the industrial routers

Transmitting serial data on an IoT router can be achieved by only one device at a time. In the case of RS-485 multi-drop, the router's RS-485 serial interface is configured as shown below. This allows access to more than one device, making multidevice operation possible through the configuration of RS-485 media-type and half-duplex. For scenarios in which access is needed for different use cases, such as reverse Telnet and access from IOx applications, use the configuration below:

```
Interface async 0/4/'x' <=== 'x' denotes ports 0 to 3 on the serial expansion module of the IR1101
  no ip address
  media-type rs485
  encapsulation relay-line
  half-duplex
```

Consequently, to provide RS-485 serial port access to IOx applications on the Catalyst IR1101, use the relay command below. This essentially allows IOx to create a TTY serial relay to the RS-485 serial RJ-45 port on the IR1101, as shown in Figure 1:

```
relay line 0/4/'x' 0/0/0
```

Application deployment

Steps needed to deploy an IOx application that will enable Modbus communications:

1. Create the application (Python is used as an example).

```
import minimalmodbus (Modbus connection python package)

//Create the Modbus object to define characteristics of the connection

Modbus_client_instance = minimalmodbus.Instrument('/dev/ttySerial',1)
← ttySerial based on Serial relay configuration from IR1101 serial interface configuration
Modbus_client_instance.serial.timeout = 5
Modbus_client_instance.mode = minimalmodbus. MODE_RTU ← RTU serial mode
RTU_Data = Modbus_client_instance.read_register(registeraddress=1,functioncode=4)
← RTU_DATA can be sent over an IP Network
```

2. Create the Dockerfile to build the application, as shown below. The Dockerfile specifies the version of the application, in this case Python 3.8, that will do the Modbus conversion.

```
FROM arm64v8/python:3.8-slim
COPY package_config.ini .
ADD app.py /
WORKDIR /
RUN pip install minimalmodbus
ENTRYPOINT python app.py
```

3. Create the definition file called package.yaml.

```
descriptor-schema-version: "2.8"
info:
 name: modbusapp
 description: "Simple Docker Style app the converts Modbus to IP"

app:
 # Show app type (vm, paas, lxc etc.,)
 cpuarch: aarch64
 type: docker
 resources:
  profile: tiny
  network:
   -
    interface-name: eth0
  devices:
   -
    type: serial
    label: HOST_DEV1

# Specify runtime and startup
 startup:
  rootfs: rootfs.tar
  target: ["python", "/app.py"]
```

4. Containerize the application and use the IOx client to package the application as a tar file.

```
docker build -t <app name goes here> .
<your ioxclient path>/ioxclient docker package <app name goes here> .
```

5. The tar file can then be uploaded to the router using one of two mechanisms:
    - IoT operations dashboard to deploy the app on the industrial router
    - Local WebUI on the industrial router
6. Make sure to select “async0” as the serial port in the application deployment page with the IOx manager.

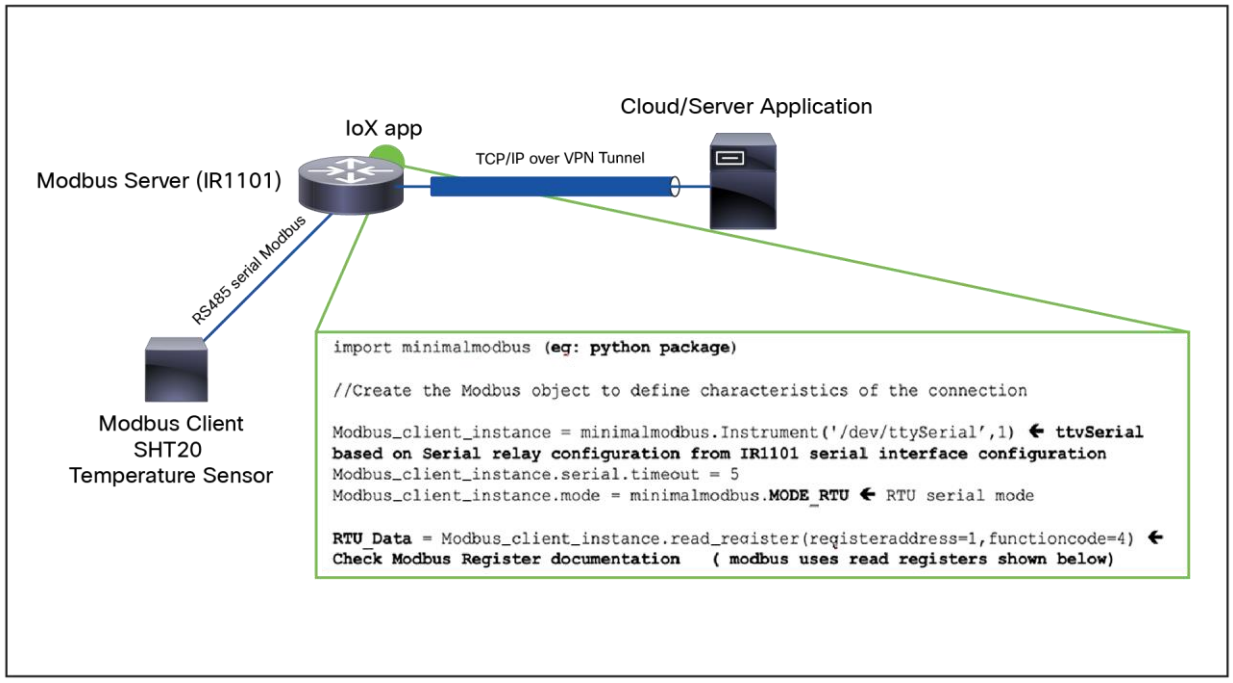


Figure 5.
IOx application deployment on industrial IoT routers

The **RTU_Data** variable in the example above stores the Modbus sensor data, in this case, temperature. The data can be formatted, filtered, etc., and other code (not shown in the figure) can be added to move data northbound to an MQTT broker, a time-series database, or another data lake in the cloud. All this is done securely over a VPN tunnel configured through Cisco IOS XE. The data can be sent to multiple destinations, or it can be kept on-premises.

In summary, it's easy to convert non-native protocols to IP in just a few steps

This approach is highly beneficial to existing customers that need to quickly have the new industrial routers communicate over protocols that are not natively supported on Cisco IOS XE, in this case Modbus.

- Leverage existing IOx and edge compute resources to communicate with Modbus devices.
- Take advantage of Cisco IOS XE and security mechanisms such as FlexVPN tunneling and zone-based firewalls to move data securely.
- Use the serial RS-485 and RS-232 ports on the industrial routers as a convenient way to connect to the Modbus clients and other PLCs and/or sensors that may communicate over Modbus, either by tunneling the serial traffic (raw socket feature) or translating the protocol as described in this paper.

Explore the Cisco Catalyst industrial routers

References:

1. https://www.modbus.org/specs.php
2. https://developer.cisco.com/docs/iox

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-3764372-00 08/23