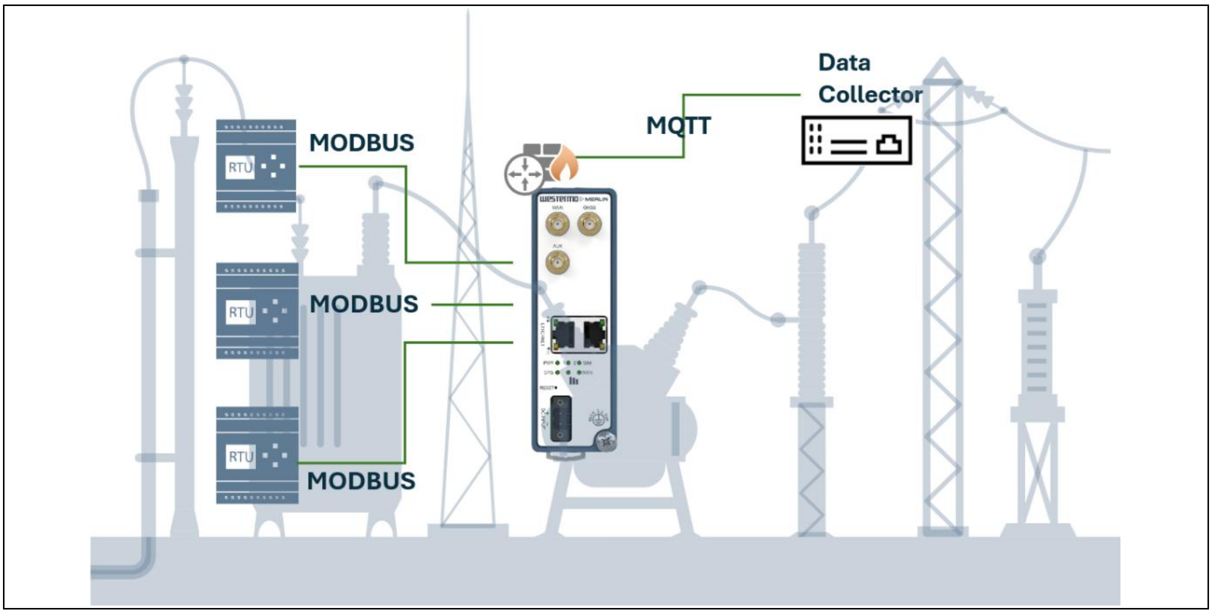
Merlin SCADA Protocol Conversion

Conversion scenario: MODBUS to MQTT

AN-004-WIE



Hardware Models	Merlin Series
Revision	1.00
Approval status	Approved
Date:	16 September 2025
Brief description	Merlin SCADA Protocol Conversion Scenario: Modbus to MQTT

1 Overview

This application note demonstrates how to set up the IECD application for protocol conversion. The example scenario illustrates **MODBUS** as the Master protocol and **MQTT** as the server protocol. The application note will show the typical steps required for protocol conversion configuration through web management.

A knowledge of SCADA protocol (Modbus/MQTT*) parameters is required, and the setup is common for all Merlin and Westermo Ireland/Virtual Access brand routers available.

The IECD application is available in the SCADA section. You must configure Master and Server interfaces as well as at least one Data Point for signal mapping. The interfaces used to communicate with local RTUs are referred to as the Master Protocol, while the interface used to communicate with the external Master is referred to as the Server (Slave) Protocol.

The table below shows the supported MQTT protocol conversion scenarios.

Master Protocol	Server Protocol
IEC101	MQTT
IEC104	MQTT
DNP3	MQTT
MODBUS	**MQTT < This Application Note**
IEC61850	MQTT

1.1 Firmware information

Firmware version: SXL-**25.04.88.000**-Beta or newer.

1.2 Assumptions

This application note shows the Merlin 3100 router and assumes the router has a factory default configuration. Instructions can be applied to other routers in the Merlin Series.

1.3 Corrections

Requests for corrections or amendments to this application note are welcome and should be addressed to support.ie@westermo.com

Requests for new application notes can be sent to the same address.

2 Configuring an IECD application

Browse to the IECD configuration page via the WebUI, enable **IECD** and follow the steps below.

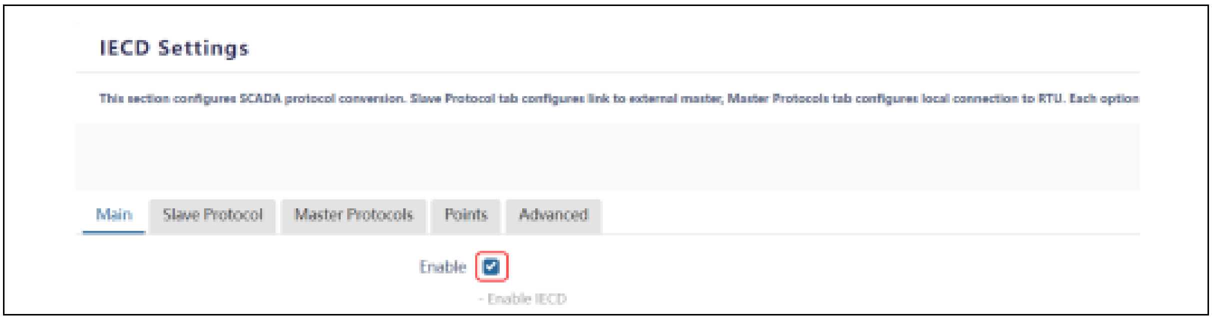


Figure 1: Main Settings

2.1 Setting up the scenario

To set up the scenario, you must configure three key elements: Enable application, the Server protocol, the master protocol and at least one data point for conversion.
Enable the application and click **SAVE & APPLY**. Remember to save and apply your WebUI configuration regularly

3 Configuring server protocol MQTT

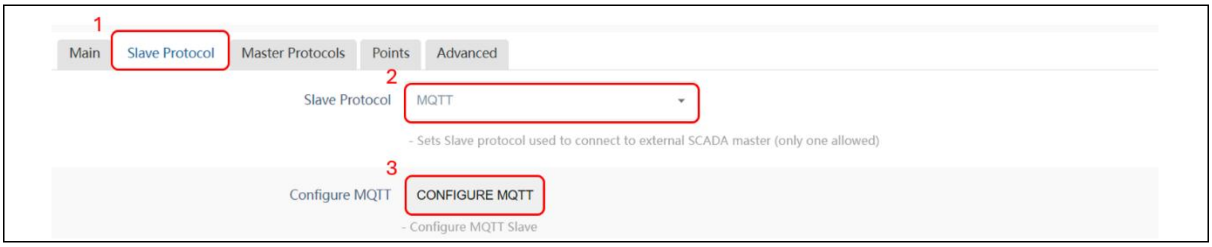


Figure 2:Slave Protocol tab in MQTT Server Settings

1. Click **Slave Protocol**.
2. Select **MQTT** as the Server protocol.
3. Click **CONFIGURE MQTT**.

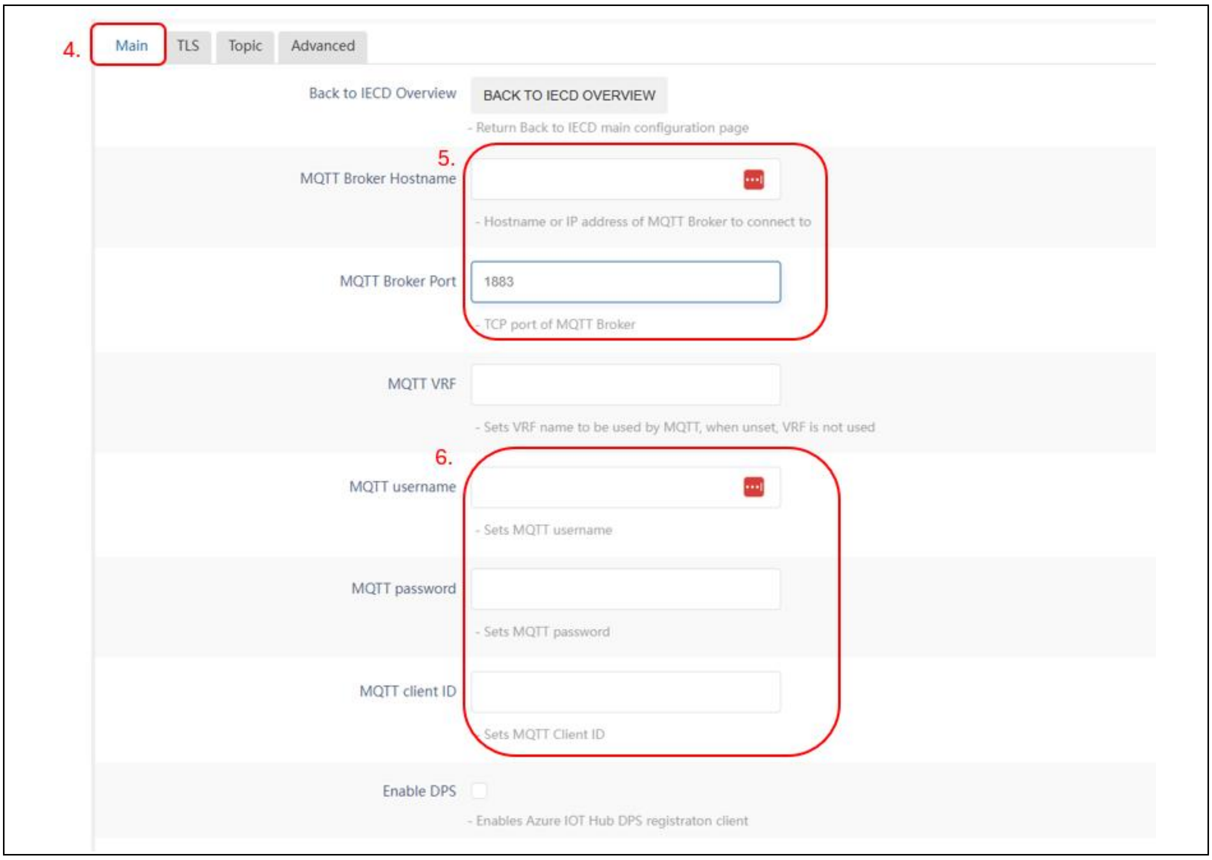


Figure 3:Main tab in MQTT Server Settings

4. Click **Main**.
5. Set the **Broker Hostname** and **Port**.
6. Set the MQTT username and password and client ID (optional).

Note: Correct DNS configurations are required when configuring FQDNs.

3.1 Configuring MQTT TLS

At minimum, to enable TLS communication, user needs to provide a Certificate Authority's (CA) Public Certificate file. More advanced configuration require upload of additional files such as Client's Public and Private Keys.

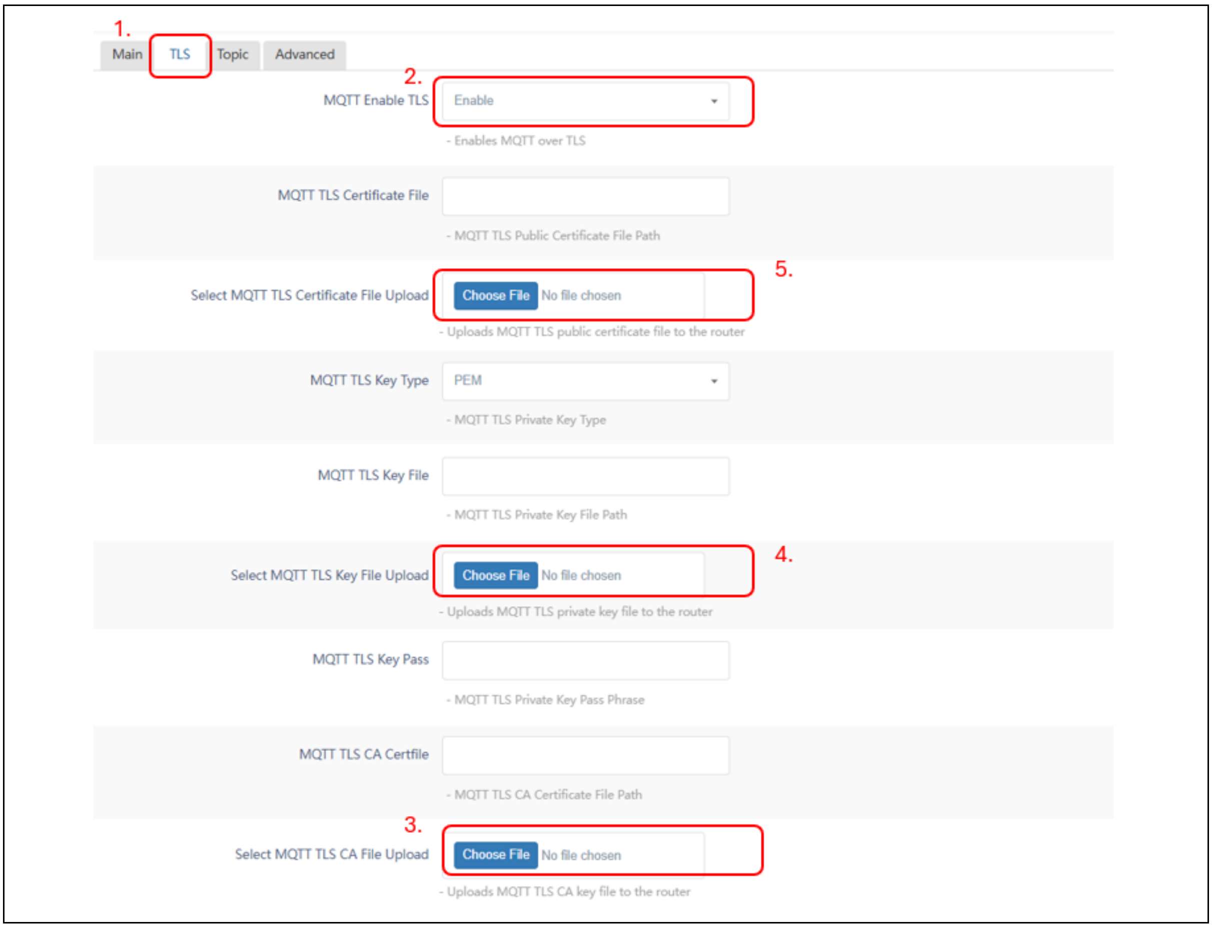


Figure 4:MQTT TLS Server Settings

1. Click **TLS** tab.
2. Select Enable in **MQTT ENABLE TLS**.
3. Click upload to upload CA public key.
4. Click upload to upload Client Private key (Optional).
5. Click upload to upload Client Public key (Optional).

Click **SAVE & APPLY** at the bottom of the page.

4 MQTT Topics

MQTT topic is a fundamental concept in MQTT's Subscriber-Broker-Publisher communication model. IECD application utilises this concept in the following way:

- Monitoring Direction Topic – Traffic originating from the end device – usually events on value changes. (MQTT Server will publish message on value change)
- (Optional) Control Direction Topic – Traffic sent to the end device – usually commands. (Described in Section 6)

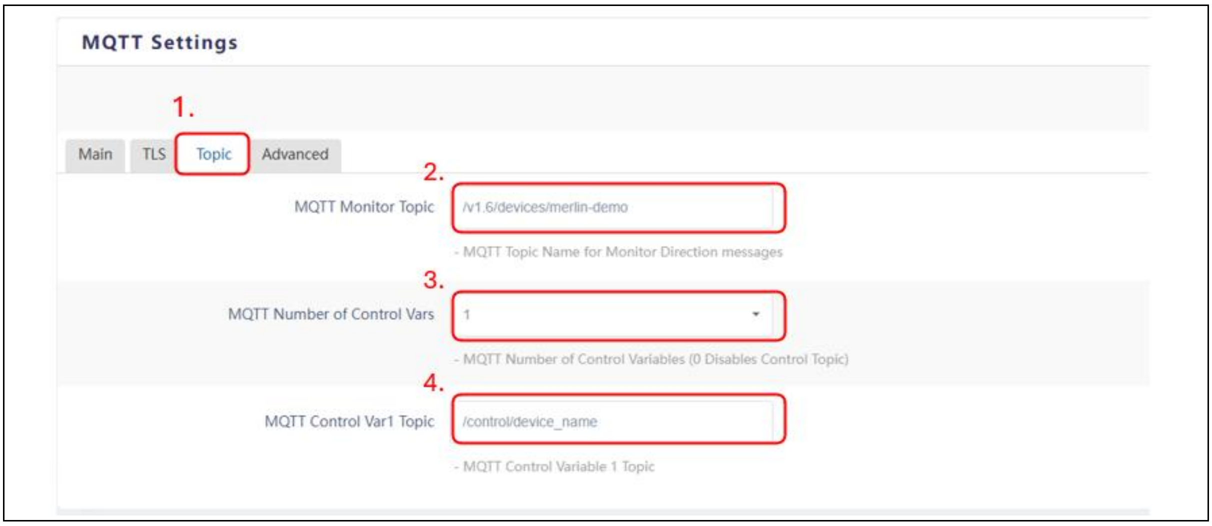


Figure 5: MQTT – Topic Settings

Monitoring Topic is mandatory:

1. Go to **Topic** tab from MQTT settings page.
2. Configure **Monitoring Topic** string.

Control Topic is Optional:

3. (Optional) Select number of **Control Topics**.
4. (Optional) Configure **Control Topic** string.

5 MQTT message (payload)

MQTT payload is configured via **MQTT Payload Format** option.

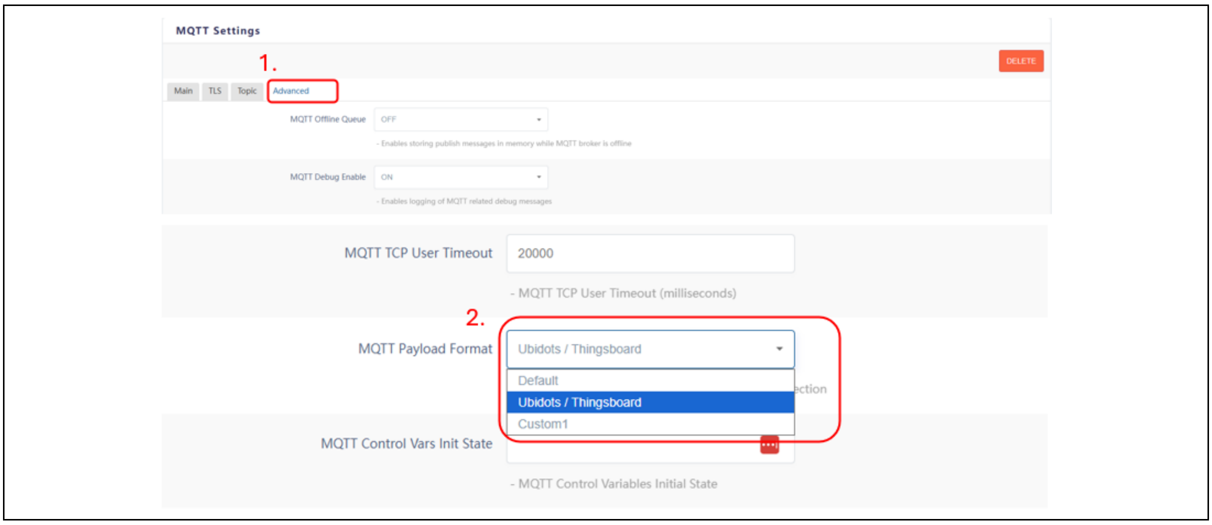


Figure 6: MQTT – Payload settings

1. Go to **Advanced** tab from MQTT settings page.

2. Configure **MQTT Payload Format** option.

There are three available **Format** option:

- Default
- Ubidots/Thingsboard
- Custom1

Please note, there is a possibly to implement a new custom MQTT payload, for all such inquiries please contact us at support.ie@westermo.com.

5.1 MQTT format default

MQTT – Modbus message content example:

```
Tx-Time:2020.05.06 21:39:56.690, ix:0, Value:ifnum:0, devaddr:10, regtype:Holding Register, regaddr:10000, val:0
```

Content Element	Explanation
timestamp	Transmit timestamp in the form YYYY.MM.DD HH:MM:SS.MMM
Internal index	Internal router index of data point
ifnum	Router Modbus interface number: 0 | 1
devaddr	Modbus device address 1..247
regtype	Register type: hr - Holding Register ir - Input Register co - Coil di - Discreet Input
regaddr	Register address/index#
regvalue	decimal register value

5.2 MQTT format Ubidots/Thingsboard

MQTT - Modbus message content example:

```
{"RegisterHR":"123"}
```

Content Element	Explanation
Name (“**RegisterHR”)**	Name is the value configured in the **option name** of config data point block
Value (“123”)	Actual value data

5.3 MQTT format custom1

MQTT - Modbus Custom1 format’s message content example:


```
{
"client":"Test",
"site":"Test_Site",
"timestamp":"1679739205",
"data":[{
  "timestamp":"1679739205",
  "sensor_name":"testp1",
  "sensor_number":"1000",
  "sensor_register_type":"DI",
  "device_address":"3",
  "device_ifnumer":"0",
  "sensor_value":"0"
}]
}
```


Configurable System option name / MQTT Payload Content per Element	**Explanation**
MQTT Client Name / client	value of config option mqtt_client_name
MQTT Site Name / site	value of config option mqtt_site_name
System clock / timestamp	UTC time of sending the message (seconds)
System clock / timestamp	data point change time UTC (seconds)
Datapoint name /sensor_name	name of the sensor
Datapoint reg addr / sensor_number	number of the sensor
Datapoint reg type / sensor_register_type	Modbus register type, one of DI, CO, HR, IR
Datapoint Mbus addr / device_address	address of the device
Datapoint / device_ifnumer	interface number of the device
Actual Value / sensor_value	register value

6 MQTT Commands

MQTT control commands are supported via **Control Topic** concept. The device need to be configured with a Control Topic, as described in the earlier section. Commands must follow the specific strings detailed in the sections below. Please note, commands must be published to the control topic by an external controller in the exact string format.

6.1 MQTT to MODBUS specific commands

MQTT – MODBUS command example:

```
“write hr ifnum=0 devaddr=10 regaddr=10000 val=123”
```

Content Element	Explanation
regtype	register type: hr - Holding Register ir - Input Register co - Coil di - Discreet Input register type can be specified without explicitly using "regtype="
ifnum	router Modbus interface number: 0 | 1
devaddr	Modbus device address 1..254
regaddr	register address / index
regvalue	decimal register value

6.2 MQTT to IEC104/101 specific commands

Interrogation Command example

"read all" – reads all points, returned in single MQTT message

Write command example:

"write ioa=100 val=on"

Content Element	Explanation
ioa	Decimal Information Object Address
val	Command value 45, 46: value = "on | off" 48,49: value = signed integer 50: value = floating point number
qval	Command qualifier (if none given, default of 0 is used) 0=no additional information 1=Short pulse 2=Long pulse 3=Persistent Output

6.3 MQTT to IEC61850 specific commands

Write command example:

```
"write linkno=0 vtype=bool ld=testIED1 ln=GGIO1 do=SPC1 val=1"
```

Content Element	Explanation
link no	IEC61850 link number (zero based
vtype	bool, int32, float
ld	IEC61850 logical device name
ln	IEC61850 logical node name
do	IEC61850 data object name
val	Value text

6.4 MQTT to DNP3 specific commands

Write command example:

```
"write binary ix=0 val=on"
```

Content Element	Explanation
Cmd_type	"**binary**" - execute binary output command "**analog**" - execute analog output command
ix	Data point index
val	"binary" command type: "on" | "off" "analog" command type: 16-bit signed integer

7 Configuring a master protocol

Modbus master can be configured to connect to the MODBUS RTU using 1. Serial or 2. TCP/IP. In a Multi RTU setup, users have the flexibility to choose either serial or TCP for any of the communication links.

7.1 Configuring MODBUS Master

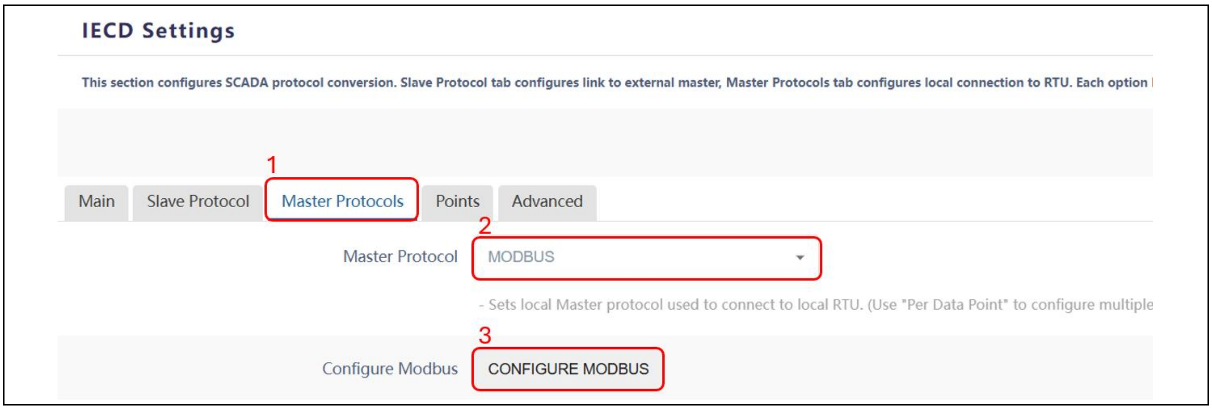


Figure 7: Master Protocols tab in IECD Settings

1. Click on **Master Protocols** tab.
2. Select **MODBUS** as the Master protocol from the drop-down list.
3. Click on **CONFIGURE MODBUS**. Please note that up to two links can be selected for the Modbus Master (Multi RTU).

7.2 Configuring MODBUS Master (SERIAL)

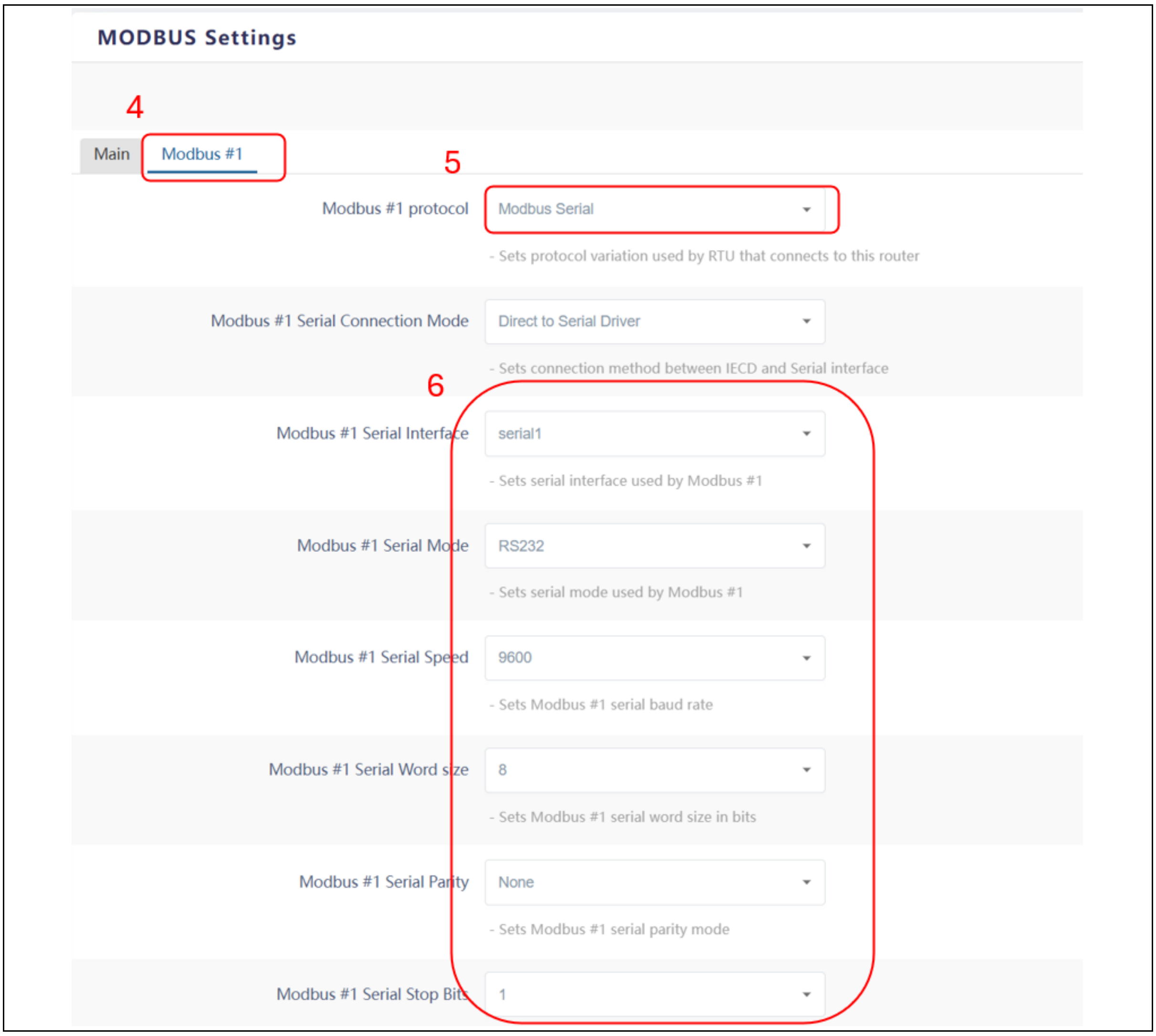


Figure 8: MODBUS #1(serial) tab in Master Settings

4. Click on **MODBUS#1**.
5. Select Modbus #1 protocol as **Modbus Serial**.
6. Configure Serial related settings.

7.3 Configuring MODBUS Master (TCP)

Figure 9: MODBUS #1(TCP) tab in Master Settings

4. Click on **MODBUS#1**.
5. Select Modbus #1 protocol as **Modbus TCP**.
6. Set the **local IP** (optional)

For Modbus Master interface it is required to configure a TCP Route. The user can specify multiple routes; Modbus protocol supports up to 247 routes.

Figure 10: MODBUS TCP route Add New

7. Navigate to the bottom of the page, find the Modbus TCP Route section, and click on **ADD**. This will reveal the following parameters.

Figure 11: MODBUS TCP Route Settings

8. Set the **Link No**.
9. Set a **Dev Addr**.
10. Set the **IP Address** of the Modbus RTU.
11. Set the **TCP Port** number. The default port for Modbus is 502.
12. Click on **Save & Apply** button on the bottom of the page, to save the configuration and run/restart IECD application.

Figure 12:The SAVE & APPLY button on MODBUS Master Settings

For Modbus Master interface it is required to configure a TCP Route. The user can specify multiple routes; Modbus protocol supports up to 247 routes.

Scroll to the bottom of the page and click **SAVE & APPLY**.

8 Configuring data point (signal)

Data points are essential for protocol conversion, requiring users to develop a data addressing/type map to facilitate the conversion between two distinct protocols. Each data point represents an individual value, such as mapping a Modbus Holding register to an IEC104 Measured Value. The configuration of data points adheres to the same principles as configuring communication interfaces, necessitating appropriate values for both the slave and Master sides.

8.1 Points configuration – manual

Navigate to the main IECD page.

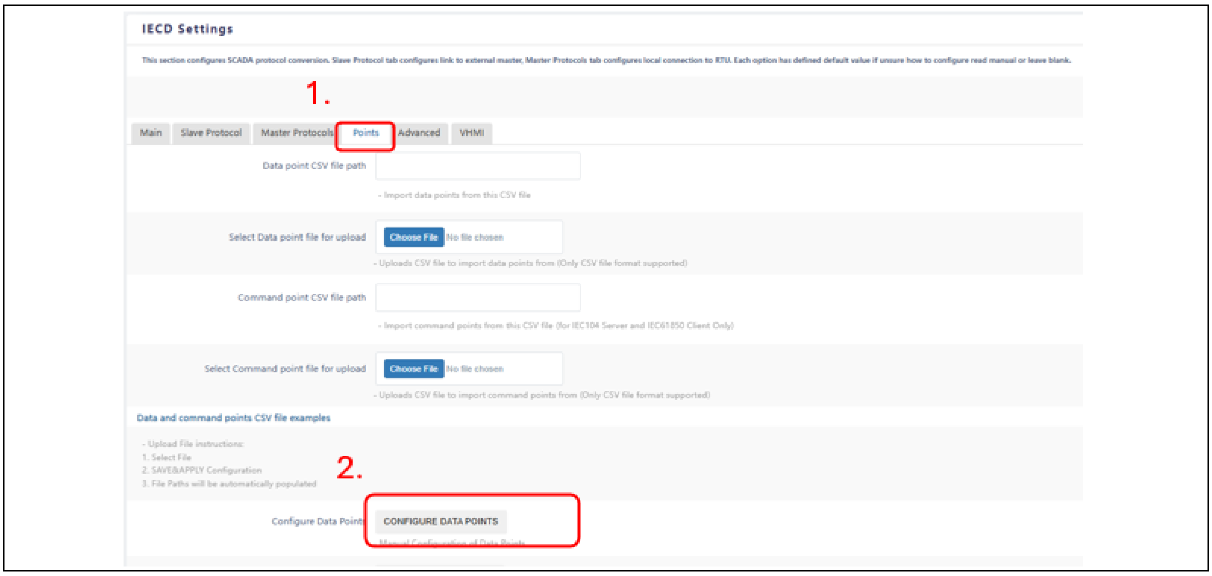


Figure 13:Points tab

1. Click **Points**.
2. Click **CONFIGURE DATA POINTS**.

Figure 14:Data point configuration

3. Click **ADD** to create a point.
4. Click **EDIT** to configure the added point.

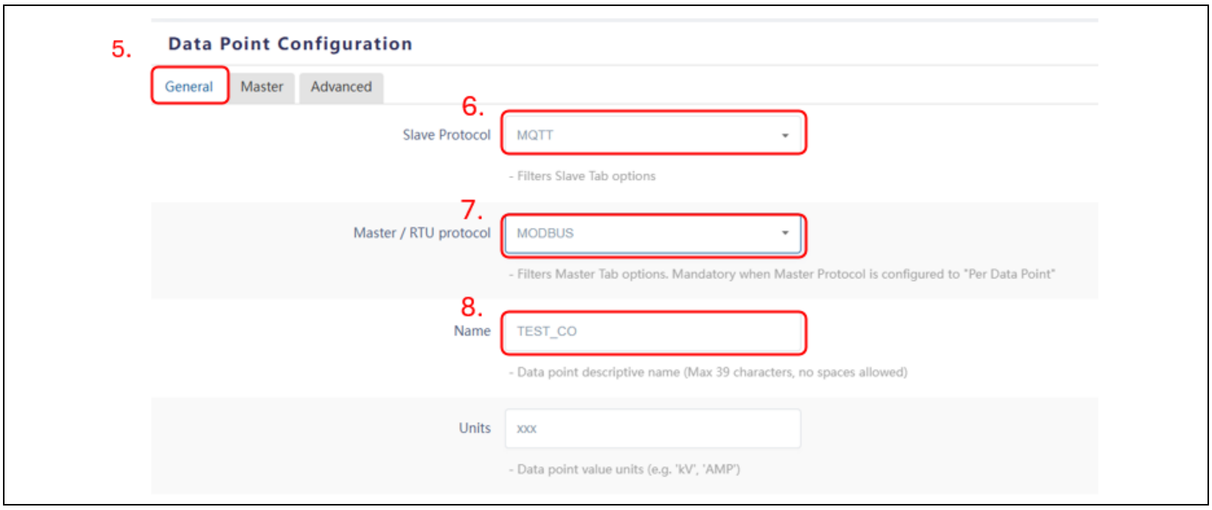


Figure 15:Data point configuration - general

5. Go to **GENERAL** tab.
6. Set the Slave Protocol as **MQTT**.
7. Set the Master/RTU protocol as **MODBUS**.
8. Create unique **name** for this data point.

Figure 16: Data point configuration - Master

9. Click on Master tab.
10. Select the **Modbus Link Number** from the drop-down list.
11. Specify the **Modbus Devaddr**.
12. Select the **Regtype** from the drop-down list.
13. Set the **Register Index**.
14. Select the **Data Format** from the drop-down list.
15. Click on **Save & Apply** button on the bottom of the page.
16. There are no configurable data point options for MQTT as the Master protocol.

8.2 Points configuration – CSV file

Data/command points can be configured via CSV file upload. The CSV file shall include all required options in its header and their corresponding values in subsequent rows. Further instructions are available on the page via 'Data and command points CSV file examples' link, highlighted in Figure 17 below.

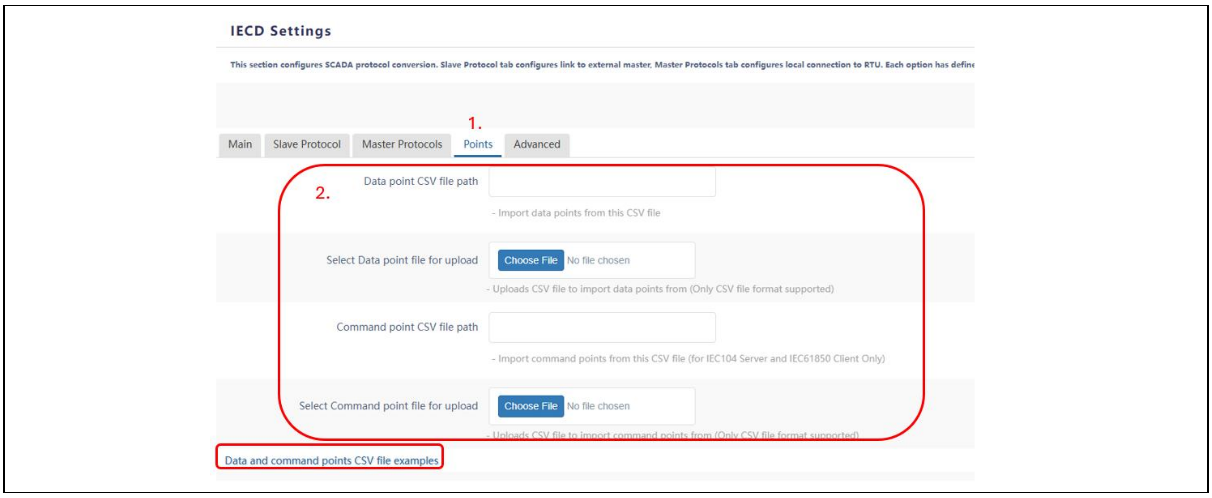


Figure 17: Points tab - CSV configurations

1. On the IECD Settings page, click **Points**.
2. Click **Choose File** and select the CSV file.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	dummy_sla	log_vaeven	di_invert	vhmi_on	vhmi_rega	vhmi_dwor	rtu_proto	modbus_d	modbus_fi	modbus_if	modbus_d	modbus_re	modbus_in	name		modbus_bitmask
2	mqtt	0	0	0	0	0	modbus	1	0	0	0	3	0	FC01		
3	mqtt	0	0	0	0	0	modbus	1	0	0	0	0	1	FC02		

Figure 18: CSV file format

9 Troubleshooting

9.1 IECD status via the WebUI

Basic troubleshooting such as verifying connections quality, values, mapping or addressing can be accessed via the dedicated IECD status page.

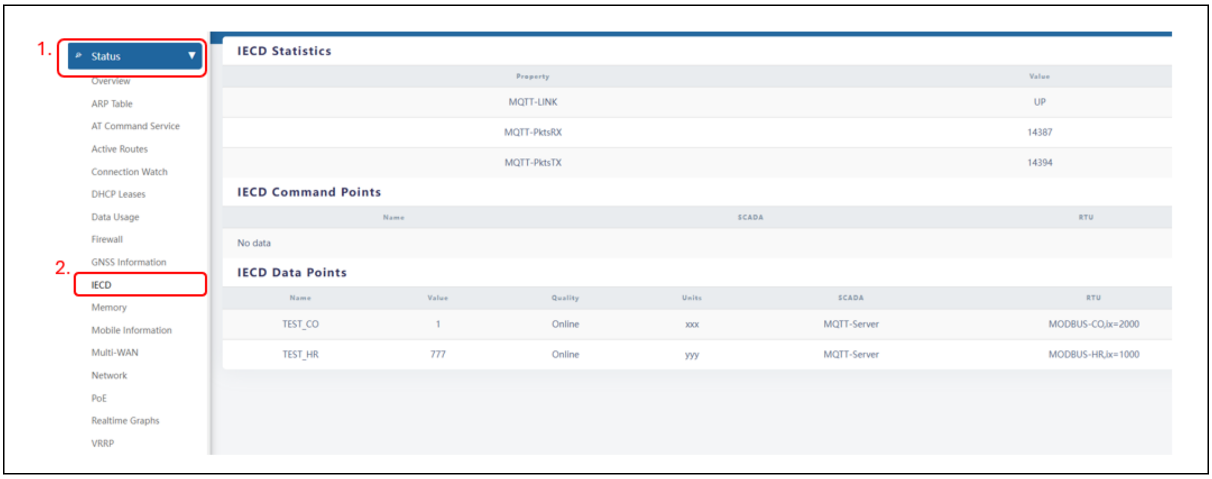


Figure 19:IECD status page

1. On the main page, click **Status**.
2. Click **IECD**.

9.2 IECD status via CLI

```
root@FC-We-Merlin4407:~# iec show stats

2025.07.22 11:58:44.411

Running protocols: (0x00004030)
* MQTT Slave
* Modbus Master
* (Modbus Master ifnum #0)

Serial interfaces:
        0: /dev/serial1 : -
        1: /dev/serial2 : -

Modbus 0 stats:
========================
Modbus DL Frames Rx 787192 Tx 787276 TxErrs 0
Modbus DL CRCErrs 0 Bad Addr 0 LengthErrs 0 SessionClose 0
Modbus App PDU Rx 787192 PDU Tx 787276 PDU Rx Errors 0 PDU Rx Exception 0
Modbus App PDU Rx Timeout 128 Unknown DevAddr 0 Rx Unexpected FC 0
Modbus App PDU TxQ Overrun 0

MQTT (MQTTS): status:
=============
State: AppConnected
Offline Queue ON: No
Output control variables:
v[0] = 0
ctlVarsGroupState = 0 (mode: AND)

MQTT PktsRx 64953, PktsTx 64987, PktsTxDropped 0, MsgRxDropped 0, MsgTxDropped 0, phTxBlocked 0, phTxErrs 0, LastErr=0
```

Figure 20: IECD status CLI output

1. To check the IECD status using CLI, connect to the router via SSH and enter the command `iec show stats`.

2. This will display the running protocols, serial interface in use, server and master protocol statistics such as the number of transmitted and received frames, and more. This information serves as a starting point for basic troubleshooting.

9.3 Configuring main debug level

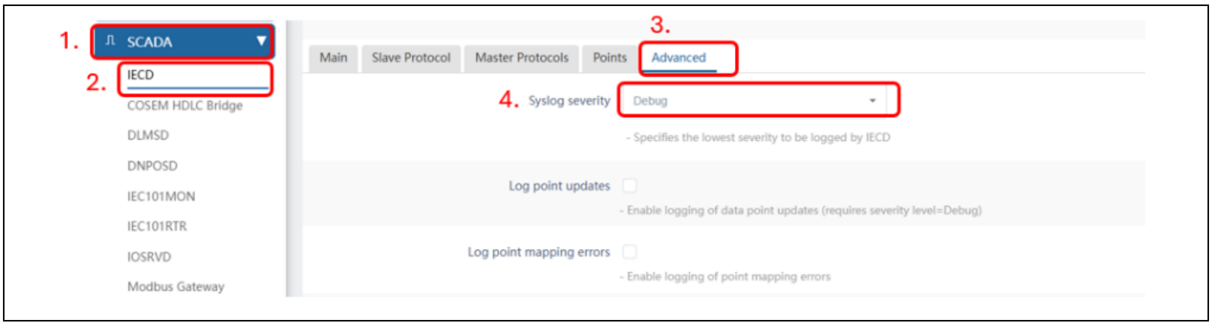


Figure 21: Enabling IECD logs

1. Click **SCADA**.
2. Click **IECD**.
3. Click **Advanced**.
4. From the Syslog severity drop-down menu, set the Syslog severity to **Debug**.
5. To save the configuration and restart the IECD application, scroll to the bottom of the page and click **SAVE & APPLY**.

9.4 Enabling debug per protocol – MQTT (Server)

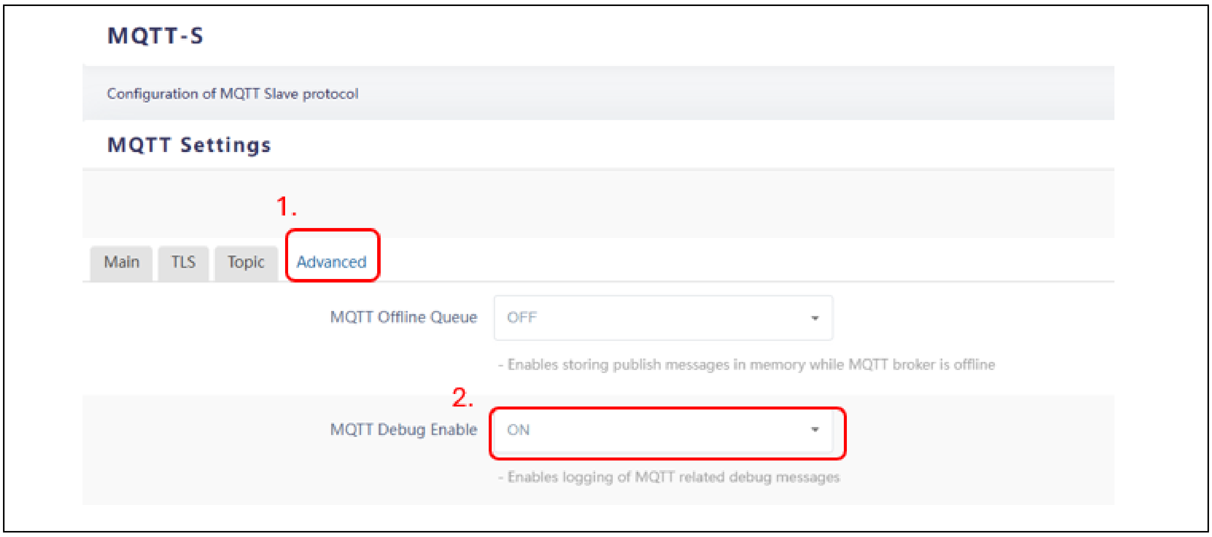


Figure 22:Enabling server protocol logs

From MQTT Settings page.

1. Click **Advanced**.
2. Enable **MQTT Debug Enable**.

9.5 Enabling debug per protocol – MODBUS (Master)

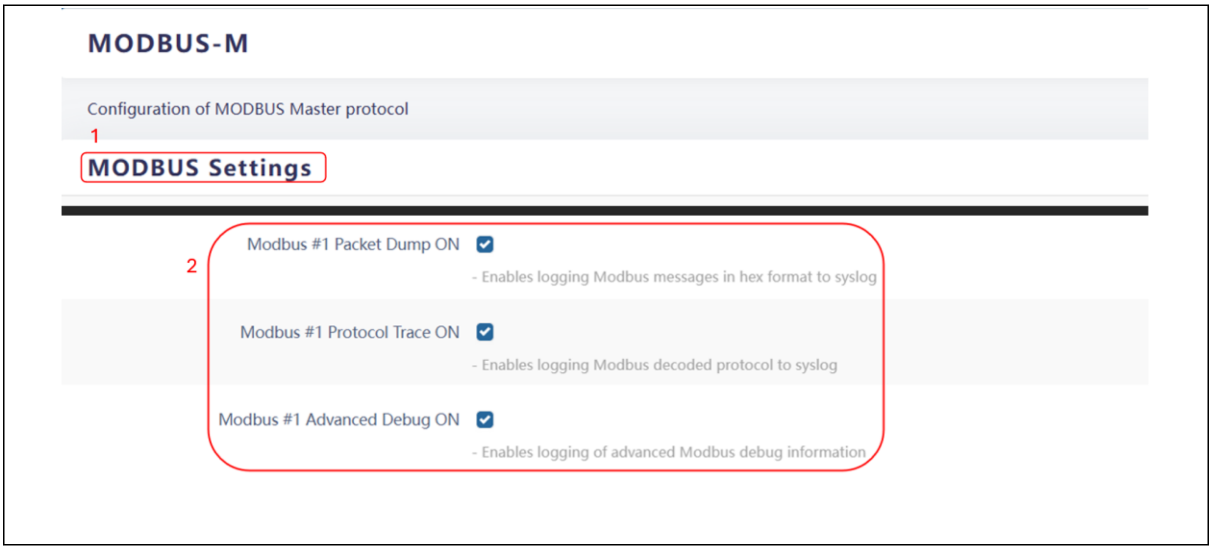


Figure 23: Enabling master protocol logs

1. Click Master Protocol and then CONFIGURE MODBUS.
2. Enable additional logging settings.
3. To save the configuration and restart the IECD application, scroll to the bottom of the page and click **SAVE & APPLY**.

9.6 Accessing logs using WebUI

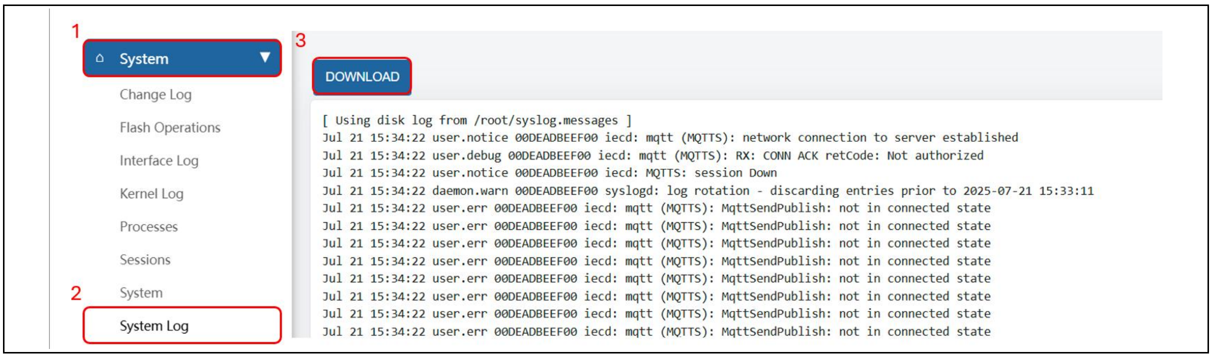


Figure 24: Downloading system logs

1. To check the logs, in the left-hand menu, click **System**.

2. Click **System Log** and search for logs related to IECD. These logs will have the name **iecd**.

3. To download the logs, click **Download**.

9.7 Accessing logs via CLI

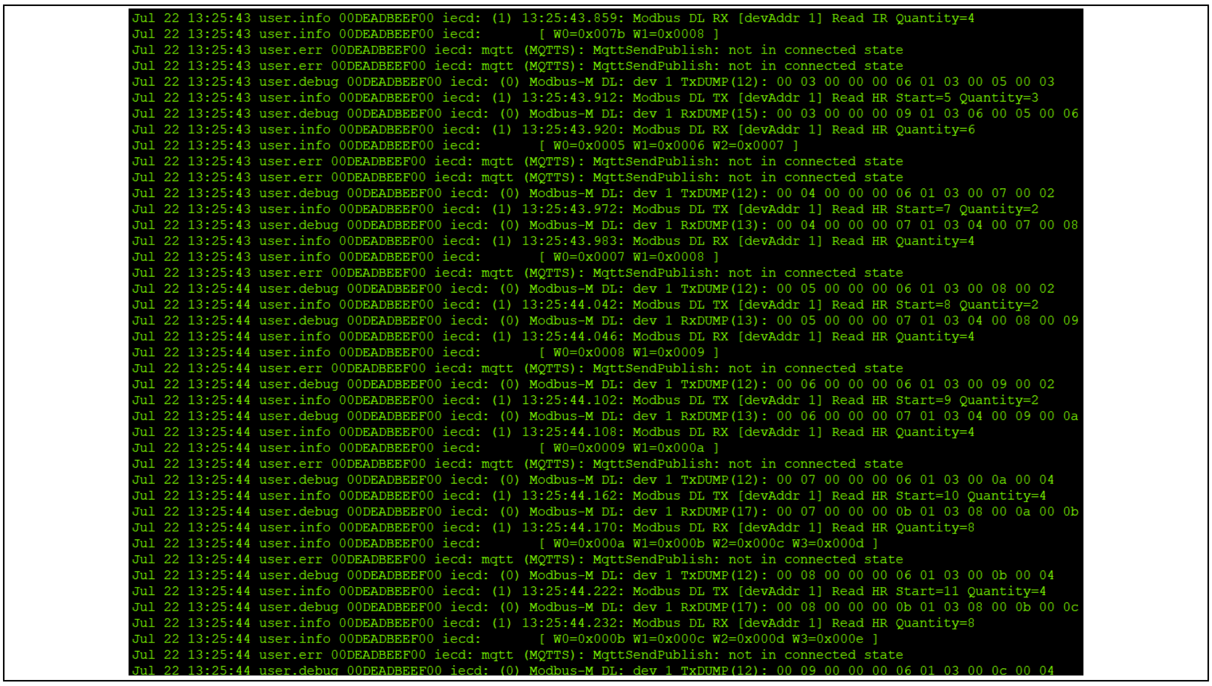


Figure 25: Enabling CLI logs output

1. To check the IECD logs using the CLI, connect via SSH and enter the command `logread -f`.

2. Logs related to the server and master protocols will be displayed.

Note: users can download the log file, as `syslog.messages` in the root directory, via WinSCP.

10 Virtual Human Machine Interface (vHMI)

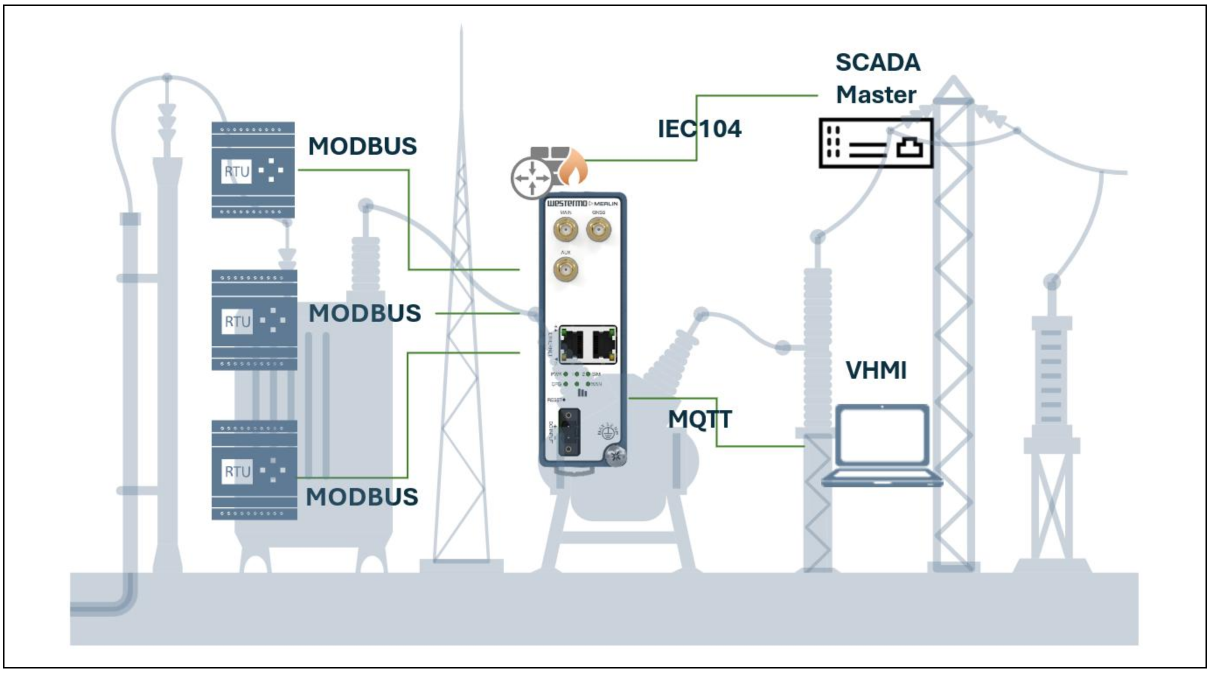


Figure 26: MQTT VHMI Topology

The above figure shows a typical topology for using the VHMI feature. This feature can also be applied to cloud services monitoring, enabling advanced functions such as tracking or monitoring resource generation and consumption.

VHMI is a secondary monitoring interface that supports communication via Modbus and MQTT. The VHMI feature is applicable only to existing protocol conversion scenarios; for example, where there is a current Modbus to IEC014 protocol conversion and the user would like to introduce a secondary interface to monitor the end device. For instance, a maintenance engineer arriving on site may wish to connect an HMI and retrieve values from the end RTU. This feature is accessible via the VHMI tab on the main IECD page.

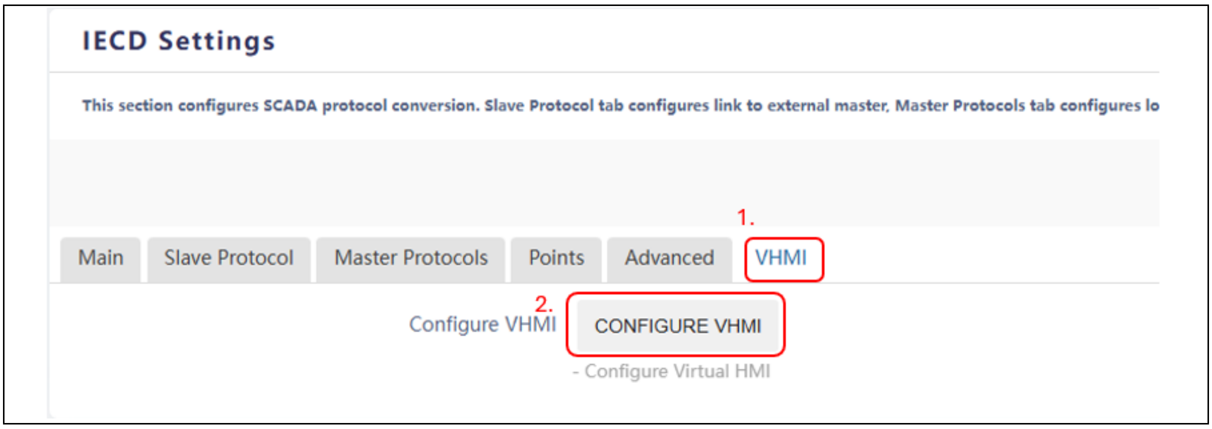


Figure 27: VHMI tab

1. Go to **VHMI** tab from main page.
2. Click **CONFIGURE VHMI** to access the settings.

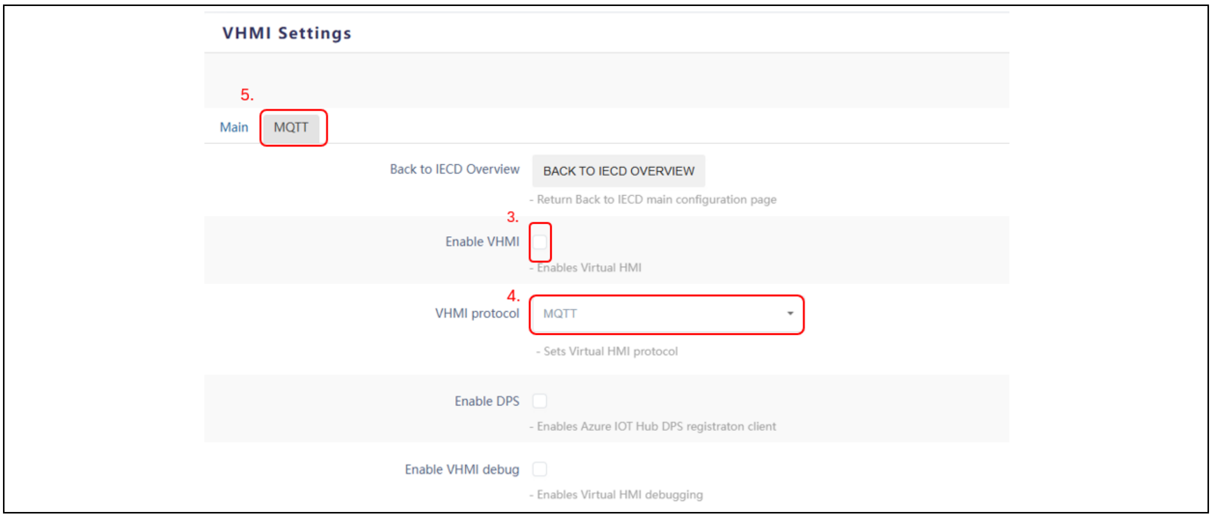


Figure 28: VHMI Settings – Main Tab

3. Enable VHMI.
4. Select MQTT from **VHMI protocol**.
5. Click **MQTT** tab.

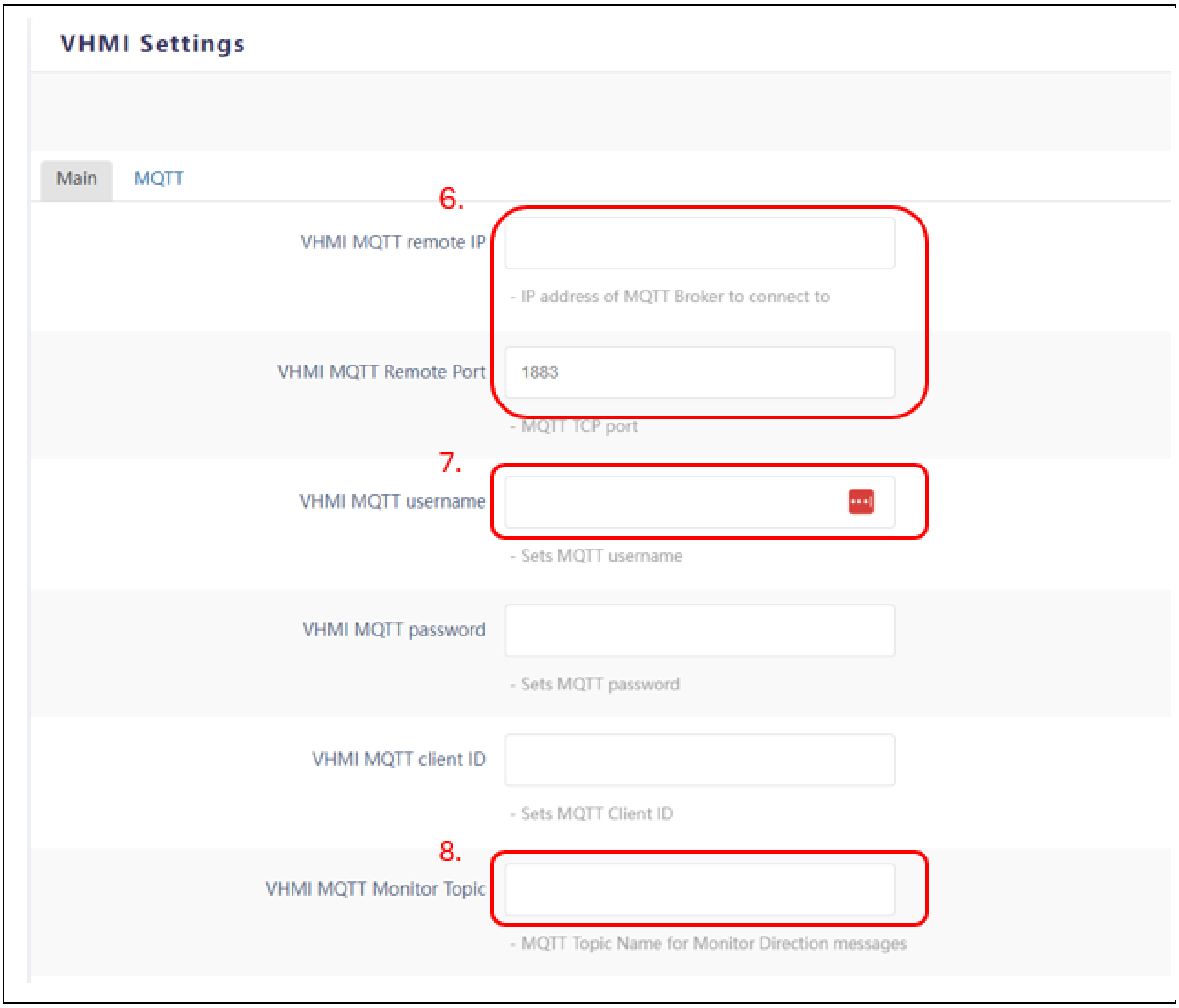


Figure 29: VHMI Settings - MQTT Tab

6. In **MQTT** tab, configure **IP:Port** settings.
7. Configure **MQTT username**.
8. Confgure **Monitoring Topic**.

Note: TLS and other advanced setting are available in MQTT this tab.

To save the configuration and restart the IECD application, scroll to the bottom of the page and click **SAVE & APPLY**.

11 Cloud Services

11.1 Ubidots

Ubidots is an Internet of Things development platform designed to help IoT entrepreneurs, system integrators, and businesses launch and scale their IoT projects. It offers tiered deployment plans including a free STEM option for hobbyists and educational users, a standard Cloud SaaS tier, and an Enterprise tier with enhanced support and private cloud capabilities. With its intuitive tools, APIs, dashboards, and visualization capabilities, Ubidots simplifies IoT data collection, management, and action.

Integrating the protocol conversion with Ubidots over MQTT is simple. User needs to generate Token and use it as a username, leaving the password blank. To add TLS security, which is highly recommended, user needs to download CA certificate and follow instructions from the section 3.1.

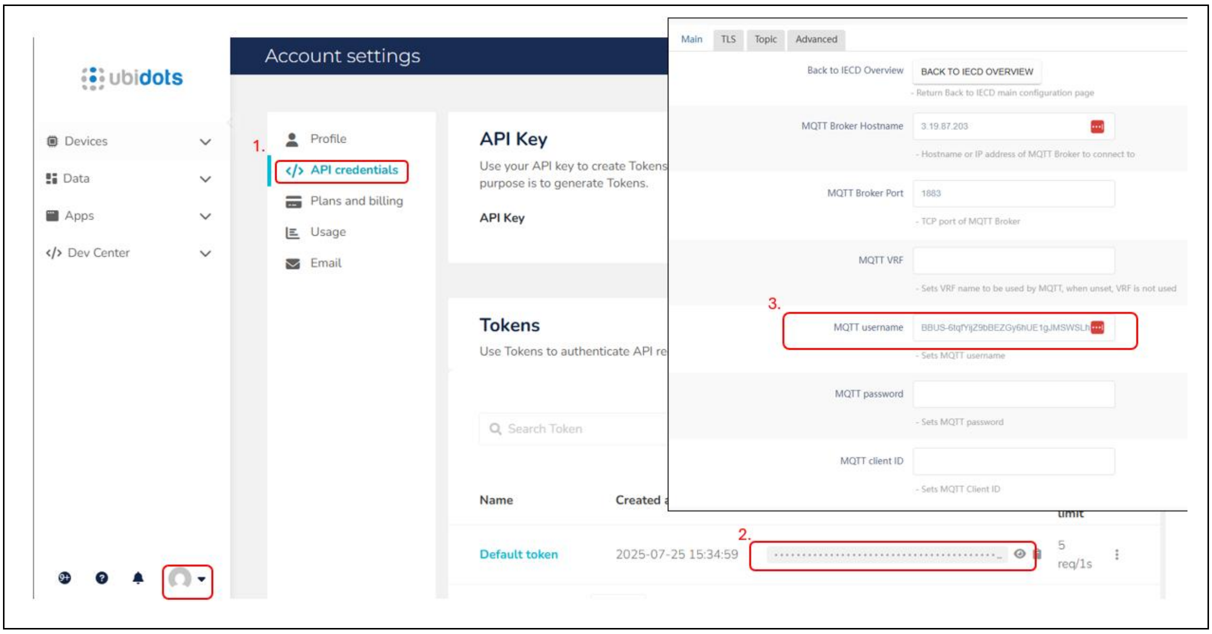


Figure 30: Ubidots – API Setting - Tokens

Go to **UBIDOTS** portal, from main page access Account Settings.

1. Click **API credentials**
2. Copy selected token
3. Go to MQTT Settings, from WebUI of Merlin device, and paste the token value as username. (Keep password field blank)

The next step is to add a device in Ubidots portal:

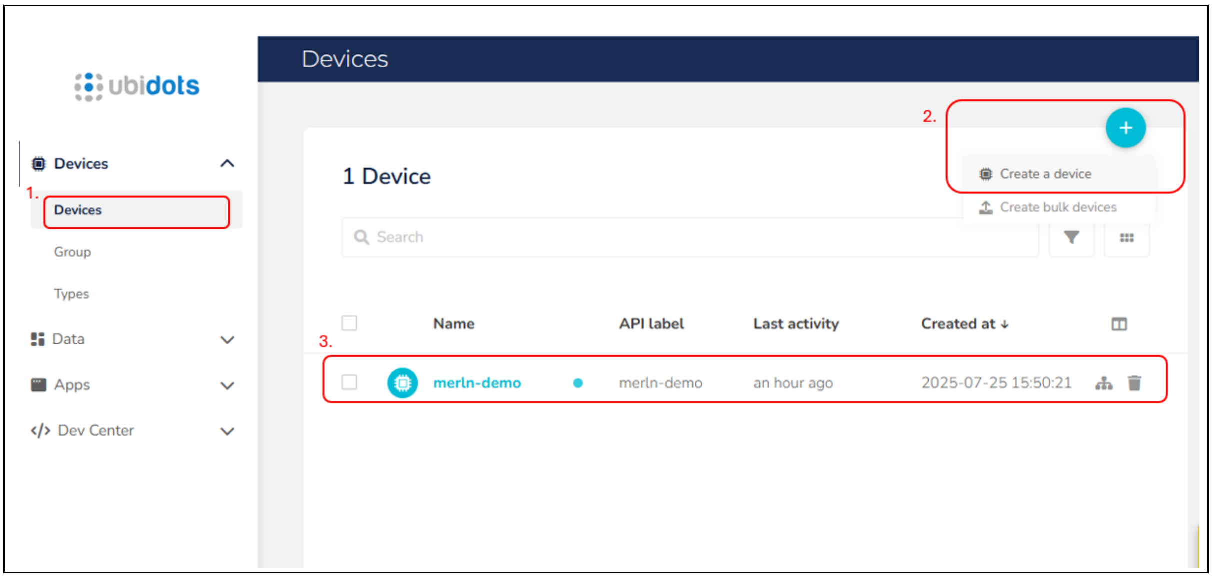


Figure 31: Ubidots - Add new device settings

1. Click **Devices**, in Devices section
2. Click **ADD** and create a device. (Device's name is used in MQTT Monitoring Topic)
3. Once the device is added, you can access it from the list in the same section.

The default Ubidots monitoring topic is **/v1.6/devices/<API-LABEL>**. For Merlin devices, this becomes /v1.6/devices/**<DEVICE-NAME>**. The complete topic for a data point is /v1.6/devices/**<DEVICE-NAME>/<DATA-POINT NAME>**.

For this demonstration, two data points were configured with unique names in Data Points configurations of Merlin Device, these names, along with the latest values, are automatically displayed in the Ubidots portal Devices/<Device-name>/Variables section.

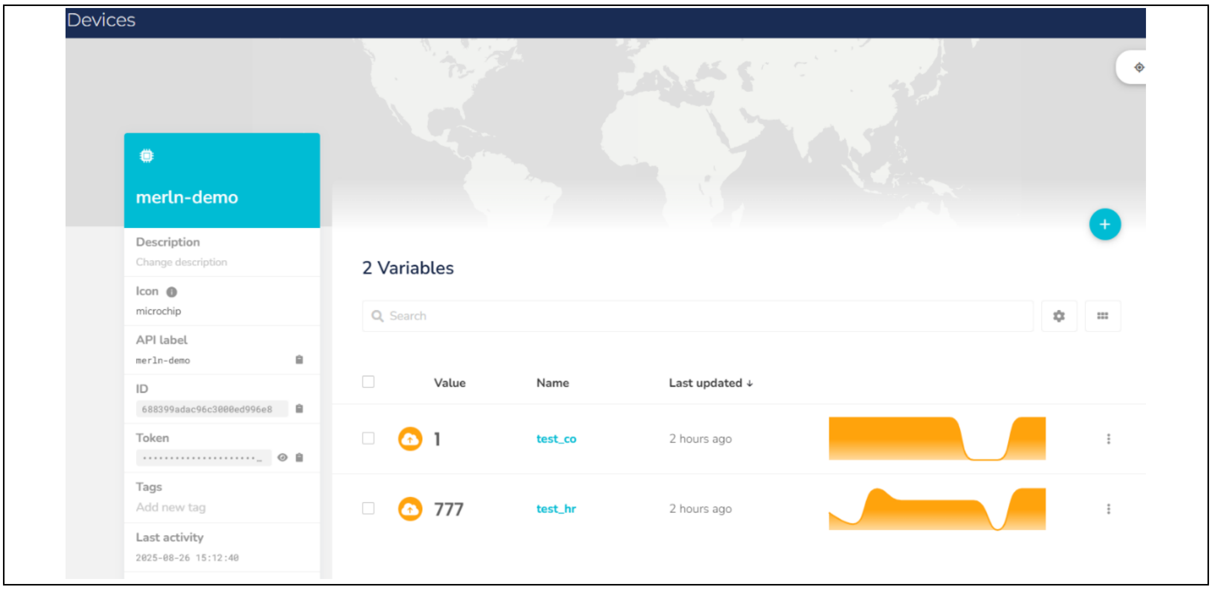


Figure 32: Ubidots - Device View

Detailed instructions can be found in **Help Centre / Integration Guides** page.