Developer's Guide for Private MQTT Messaging

of SenseCAP SensorHub

SENSECAP

Seeed Technology Co., Ltd.

Table of Content

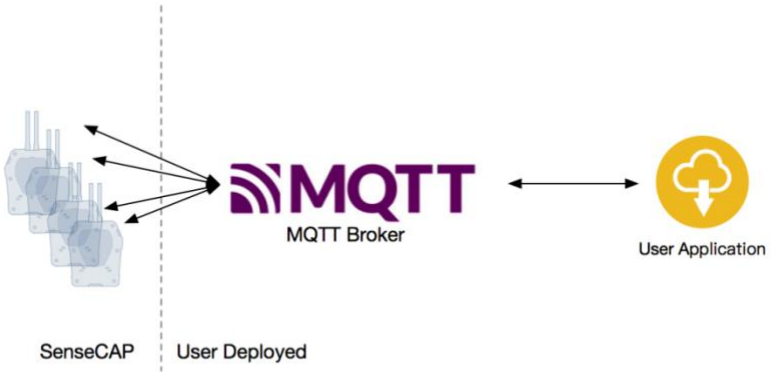
1. Overview

SenseCAP Sensor Hub – 4G is a compact solution for remote environmental monitoring. It consists of a powerful data logger that can connect a maximum of 40 RS-485 sensors and a wide range of sensor selection, you can use it for applications like weather station, air quality monitoring. It supports uploading data to Seeed's SenseCAP server and users' private server.

This guide will show you the message flow of the MQTT connection between SensorHub and the private MQTT broker, and also the format of the uplink & downlink messages.

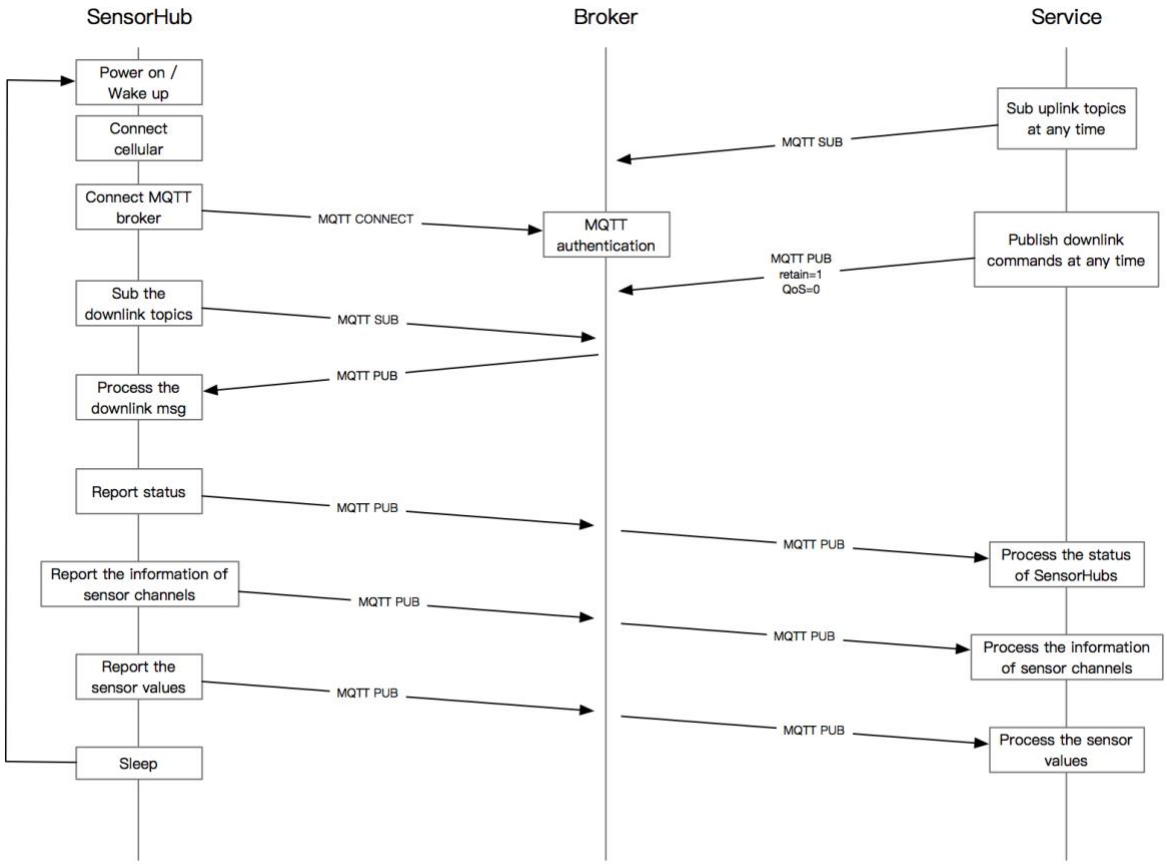
1.1 Architecture and Message Flow

The diagram of the architecture is shown below,



SensorHub connects users' private MQTT broker via the MQTT protocol. Based on the MQTT protocol, we defined several topics, carrying the uplink and downlink messages. Generally there're three methods to develop the private MQTT message service, ①Modify the MQTT broker to add the message processing for SensorHub, ②Develop another standalone application which connects to the standard MQTT broker as a MQTT client, make subscriptions and publish messages, ③mix the above two.

Users can configure the SensorHub to connect to a private MQTT broker by the companion GUI tool. For more details, please refer to <Sensor Hub 4G Data Logger User Guide>. After the configuration, the message flow between SensorHub and the cloud side is shown as the following diagram.



We will give more details on how to develop cloud service in chapter 4. Service Development Guide.

1.2 The MQTT Connection Parameters from the Device

The connection parameters used by the SensorHub when it's doing MQTT connection is shown below,

- Broker Address- configured via the Configuration Tool
- Port - configured via the Configuration Tool
- ClientId – "d-6-<EUI>", where the EUI is the device's EUI on the product label
- Username - configured via the Configuration Tool
- Password - configured via the Configuration Tool
- Timeout – `60 seconds`
- CleanSession - `false`
- `TLS - not applicable for SensorHub`

1.3 The Private Topics

For every device that connects the broker, it has its private topics to publish messages to / subscribe. Once the connection is authorized by the broker, the device will firstly subscribe a specified topic through which the device can receive the downlink command from the cloud side, and in parallel upload its status and measurements through an uplink topic.

- Publish（Uplink）: `iot/ipnode/<deviceEui>/update/#`
- Subscribe（Downlink）: `$SHADOW/ipnode/<deviceEui>/get/config`

The "#" in the uplink topic is a wildcard defined by the MQTT protocol. In the next chapter, we will introduce separately the uplink messages and their related topics.

> **INFO**: $SHADOW is only a string, not special symbol. <deviceEui> stands for the EUI on the product label.

2. Uplink

2.1 Introduction

There're three types of uplink messages, including report of device status, report of information of sensor channels and report of sensor values(aka measurements). These messages are one-way messages, not expecting the response from the cloud side, so we call them "Events".

> **NOTICE:** The QoS of all the uplink messages is 1.

2.2 Basic Message Structure

- Topic： `iot/ipnode/<DeviceEui>/update/event/<eventName>`
- Payload:

```
{
"requestId": "aaaa-aaaa-aaaa-aaaa",
"timestamp": "<millisecond timestamp>",
"intent": "event",
"type": "simple",
"deviceEui": "local",
"events": [{
      "name": "<eventName>",
      "value": {   },
      "timestamp": "<millisecond timestamp>"
 }]
}
```

The above is the basic structure of the uplink message and its related topic. The <eventName> in the topic is a placeholder which can be replaced with one of the event names in table 1. The “Payload” is the MQTT message body, which is a JSON string.

Table 1 Definitions of Event Names

eventName	**Description**
change-device-status	for report of device status and information
update-channel-info	for report of information of sensor channels, i.e. whether a RS485 port has any kind of sensor connected and what kind sensor it is
measure-sensor	for report of sensor values(measurements from the sensors)

Field of Payload	**Description**

requestId	The randomly generated unique ID of the message(in most cases the service side can ignore this)
timestamp	Millisecond timestamp, obtained from the cellular module. Please note that this timestamp is the moment when the message is being sent, not the moment of sensor sampling.
intent	Fixed value “event”
type	Used for packet segmentation. - simple, the packet is a single complete packet - cev, this is a segment of a giant packet - fev, this is the last segment of a giant packet When omitted, the packet is a single complete packet. For more details please see the next section 2.2.1 .
deviceEui	Fixed value “local”, means the message comes from this device, the EUI can be extracted from the topic
events	The array of events
--name	The name of the event, equals to the <eventName> in the topic
--value	The content of the event, separated introductions on different event types will be done in the following sections.
--timestamp	the timestamp of when the event occurs, millisecond timestamp, take the “measure-sensor” event as an example, this timestamp stands for the moment of sampling

2.2.1 Packet Segmentation

SensorHub is a RAM limited device, so when it’s sending a giant packet, segmentation is needed.

If the type field of the message equals to “cev”, it means that this message is a segment of a giant packet, i.e. slice packet. The slice packet has a complete JSON structure, but the events array is a slice of the original array in the original giant packet, to fit the RAM requirement of SensorHub.

When SensorHub has plenty of sensors connected or one of the sensor has several measurements, the “update-channel-info” message and the “measure-sensor” message will possibly be segmented.

If the type field of the message equals “fev”, it means that this is the last segment of the giant packet.

In general, the service side need not to handle the segmentation, except that it has the following requirements.

- The service needs extremely quick sense on the change of sensor channels.
- The minimum storage unit in the service is a whole collection/sample. For an example, the SensorHub has several sensors connected, among those sensors 10 measurements will be produced, the service must collect all the 10 measurements before it save them to database.

We will give some advices in section 4.2 on how to handle packet segmentation and how to ensure the idempotence of message.

2.3 Report Device Status

- Topic: `iot/ipnode/<deviceEui>/update/event/change-device-status`
- Payload

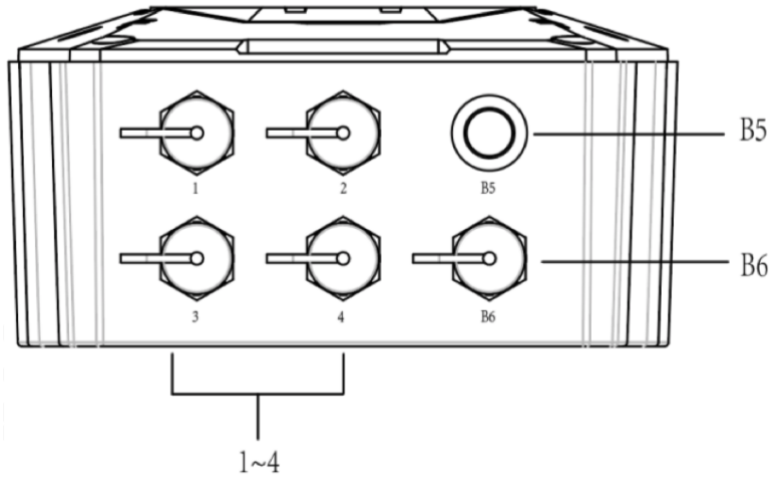
```
{
"requestId": "aaaa-aaaa-aaaa-aaaa",
"timestamp": "<millisecond timestamp>",
"intent": "event",
"deviceKey": "", //not used
"deviceEui": "local",
"events": [{
   "name": "change-device-status",
   "value": {
      "3000": "99", // battery
      "3001": "1.0.0", //hardware version
      "3502": "1.1.0",  //firmware version
      "3013 ": "30", // onboard temperature
      "3015": "89860412091880130000", // SIM card CCID
      "3567" : "EC25EUXGAR08A05M1G",  // cellular module model
     "3017": 3,  //battery charge status
     "3900": "60", //sample interval,  seconds
     "3009": -65, // cellular RSSI
     "3012": 179, //network latency, milliseconds
     "3018": 1, // sample counter
     "3005": "60" // report interval, seconds
   },
   "timestamp": "<millisecond timestamp>"
}]}
```

Table 2 The Fields of Device Status

Field ID	Description
3000	Percentage of battery， 0~100
3001	Hardware version *
3005	Status report interval, seconds
3009	Cellular Signal RSSI
3012	Network latency, milliseconds
3013	onboard temperature
3015	SIM card CCID *
3017	Battery charging status, 1- charging , 2 – full, 3 – no charging
3018	sample counter, reset to 0 after reboot
3502	firmware version *
3567	The model of cellular module *
3900	sample interval, seconds

INFO： Fields with ID 3001, 3015, 3502 and 3567 are reported once at power-up, others are reported periodically.

2.4 Report Sensor Channels



As shown above, the SensorHub has 4 RS485 ports on the bottom side of the enclosure, using some kind of aviation plug, we call them “physical ports”. For every physical port, the bus protocol based on RS485 – Modbus is used to interact with sensors. Several Modbus slaves can be connected to the bus, means that we can connect several sensors to one

physical port, we call every sensor/Modbus slave a “virtual port”. Up to 10 virtual ports can be allocated for one physical port. Take physical port 1 as an example, its associated virtual port number will be 10 ~ 19. The SensorHub will scan every physical port on bootup, detecting whether any sensor is connected, and assign a virtual port number to that sensor with a particular logic.

The number of virtual port consists of 2 digits, the high digit is the physical port number [1, 4], while the low digit is the index assigned [0, 9].

The definition of sensor channel report message is,

- Topic：`iot/ipnode/<deviceEui>/update/event/update-channel-info`
- Payload：

```
{
"requestId": "aaaa-aaaa-aaaa-aaaa",
"timestamp": "<millisecond timestamp>",
"intent": "event",
"type": "simple",
"deviceEui": "local",
"events": [{
   "name": "update-channel-info",
   "value": [{
      "channel": "10",
      "channelType": "1",
      "sensorId": "0110001003900000",
      "sensorType": "101E",
      "status": "normal"
   }, {
      "channel": "11",
      "channelType": "1",
      "sensorId": "0111001003900000",
      "sensorType": "1011",
      "status": "normal"
   },{
      "channel": "12",
      "sensorId": "0112001003900000",
      "sensorType": "6000",
      “measurementIds”:[4097,4098],
      "status": "normal"
   }, {
       "channel" : "13",
       "status" : "idle"
   },...],
      "timestamp": "<millisecond timestamp>"
```

}]}

Field	Description
channel	the number of the virtual port
channelType	The interface type of the channel, depending on the interface of the sensor 1: 485 Sensor； 2: Seeed Uart Sensor； 3: 485 Output； 4: Seeed Uart Output；
sensorId	The unique ID of the sensor on this channel
sensorType	The type of this sensor, refer to 5.2 Sensor Types
status	The status of this channel "normal" – with sensor connected, the sensor works well, "idle" – no sensor connected, or the sensor can't be recognized "abnormal" – certain sensor is detected at the initial scan, but the subsequent communication fails
measurementIds	The IDs of the measurements produced by the sensor(on this channel), only applicable for the channels which have user-defined sensor connected(not Seeed pre-defined sensors). These measurement IDs are defined by the user with the Configuration Tool.

Notes：

1. If the status of the channel is idle or abnormal, only "channel" and "status" fields are included in the JSON object of this channel, other fields are omitted.

2. If the sensor is a user-defined sensor, the "channelType" field will be omitted from the JSON object.

2.5 Report Measurements

- Topic： `iot/ipnode/<deviceEui>/update/event/measure-sensor`
- Payload

```
{
"requestId": "aaaa-aaaa-aaaa-aaaa",
"timestamp": "<millisecond timestamp>",
"intent": "event",
"type": "simple",
"deviceEui": "local",
"events": [{
   "name": "measure-sensor",
   "value": [{
      "channel": "10",
      "measureTime": "<millisecond timestamp>",
      "measurements": {
         "4097": "30.1",
         "4098": "50.5",
      }, {
      "channel": "11",
      "measureTime": "<millisecond timestamp>",
      "measurements": {
         "4099": "100"
      },
   }, ...]
}]}
```

Field	**Description**
timestamp	The timestamp of the moment when the message is being sent
intent	Fixed value "event"
type	See 2.2 Basic Message Structure
deviceEui	See 2.2 Basic Message Structure
events	The array of events
--name	Fixed value "measure-sensor"
--value	The object of measurement
----channel	The source virtual port number
----measureTime	The timestamp of the moment when the sample is done
----measurements	The measurement value(s), one sensor can output several measurements, the “key” of this object is one of the measurement IDs

	listed in 5.1 Measurement IDs. For an example, "4097": "30.1", means the air temperature is 30.1℃.

3. Downlink

3.1 Introduction

With downlink messages the service side can issue simple control commands to the device. Currently SensorHub supports the following commands:

- Modify the sample interval
- Reboot

Since SensorHub is a low power device with deep sleep mode, it only connects to the MQTT broker when it's awake, we recommend to mark the downlink message with the **retain** flag. The firmware of SensorHub will ensure that the retained message is processed only once.

To reduce the consumption of data traffic, all the JSON string of downlink messages should be compressed before it's sent, removing new lines and blank spaces. If not, the firmware of SensorHub will fail to parse the message.

3.2 Message Structure

- Topic：`$SHADOW/ipnode/<DeviceEui>/get/config`
- `Payload`

```
{
  "timestamp": <millisecond timestamp>,
  "desire": {
    "3900": {  // Modify the sample interval
      "ver": "<millisecond timestamp>",
      "value": <new interval, seconds>
    },
    "3910": {  //reboot
      "ver": "<millisecond timestamp>"
    }
  }
}
```

Field	Description
timestamp	The timestamp of the moment when the message is being sent
desire	The desired command
--KEY	The command ID
---ver	The version of the command, using millisecond timestamp, through which the firmware do deduplication.
---value	The value of the control

- The downlink message must be, QoS=0, Retain=1.
- On receiving the command the device will firstly compare the version of the command with the one saved in the device, if the version of the command is newer, execute the command and save the version to Flash.
- “3900” is the command ID for modifying the sample interval
- ”3910” is the command ID for reboot
- An example, to change the sample interval to 120 seconds: {"timestamp": 1601255713000,"desire": {"3900": {"ver":"1601255713000","value":120}}}

4. Service Development Guide

4.1 Step-by-Step

1. Prepare the MQTT broker, configure the authentication method to username&password.

2. Configure the SensorHub via the Configuration Tool, achieving that the SensorHub connects to the broker successfully. Please refer to <Sensor Hub 4G Data Logger User Guide> for detailed instructions.

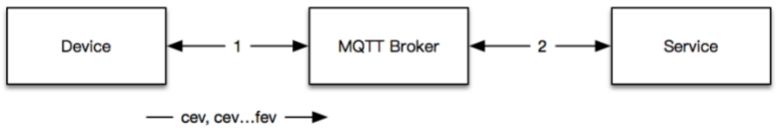
3. Utilizing a MQTT client tool, connect the client to the broker and subscribe the uplink topics, make sure that the tool can also receive the messages published by SensorHubs.

4. Program the service application, and please notice that use QoS 1 to subscribe the uplink topics. This is really not the major object of this documentation and it’s more related to your system architecture and requirements on the service.

5. If downlink is required for your service, just publish downlink messages with QoS 1 and retain flag to the specific device's topic. The control is one-way, the device will not respond any confirmation. If your service needs strict close-cycle control, the device status report message can be used.

4.2 Assembly of Packet Segments

Due to the uncertainty of the network transfer, many network protocols have the mechanism to handle timeout and retransmission, including the MQTT protocol. In the diagram below, the communication path 1 is simple, it's generally TCP persistent connection, with a balancer in some cases. Once the MQTT connection is established, the path 1 can be treated as a TCP socket. Benefitting from the QoS mechanism of the MQTT protocol, we can assume that the messages arriving at MQTT broker are order-preserved.



The communication path 2 will be more complicated, more attentions should be paid to preserve the order of the messages. This is not only done by the architecture, but also the implementation art of the programming, especially for those programming languages with asynchronous features. If the architecture of path 2 is the bottle neck of implementing idempotence, the consideration of hacking the MQTT broker can be done, in which case the assembly of segments will be done immediately at where the slices are received in the very beginning.

The "requestId" of all the slices of a giant packet are the same, based on this we recommend the following assembly process for segmented packet.

- On receiving a message with type "cev" , push it into the cache/queue indexed with this requestId.
- On receiving a message with type "fev", pop all the slices in the cache/queue, merge the events array of all the slices including this fev slice.

4.3 Logic of Device Offline

SensorHub is a low power device with sleep mode, the MQTT connection can only indicate that the Hub is online, we can't treat the Hub as offline since its MQTT connection is closed. The way we recommend to detect the offline of the Hub is based on the interval, we assume the Hub is offline on some kind of failure since no message is received from it over N intervals. And we suggest N = 2.5.

Once offline is detected, initial diagnosis can be done as the following.

1. Check the power, if the Hub is an AC powered variant, please check the AC adapter or solar system can output power normally, if the Hub is an battery powered variant, please check the wire of the solar panel and whether the panel is covered by stuff.
2. Check the account of SIM card, the remaining data
3. Check the sensor, whether it's damaged, short circuited, water soaked.

5. Appendix

5.1 Measurement IDs

ID	Measurement Name(Chinese)	Measurement Name(English)	Value Range	Unit
4097	空气温度	Air Temperature	-40~90	℃
4098	空气湿度	Air Humidity	0~100	%RH
4099	光照	Light	0~188000	Lux
4100	二氧化碳	CO2	0~10000	ppm
4101	气压	Barometric Pressure	300~1100000	Pa
4102	土壤温度	Soil Temperature	-30~70	℃
4103	土壤湿度	Soil Monisture	0~100	%RH
4104	风向	Wind Direction	0~360	°
4105	风速	Wind Speed	0~60	m/s
4106	pH	pH	0~14	pH
4107	光通量	Light Quantum	0~2000、0~5000	umol/㎡ s
4108	电导	Eletrical Conductivity	0~23	dS/m
4109	溶解氧	Dissolved Oxygen	0~20	mg/L
4110	土壤体积含水量	Soil Volumetric Water Content	0~100	%
4111	土壤电导	Soil Electrical Conductivity	0~23	ds/m
4112	土壤温度(三合一传感器)	Soil Temperature(Soil Temperature, VWC & EC Sensor)	-40~60	℃
4113	每小时降雨量	Rainfall Hourly	0~240	mm/hour
4115	距离	Distance	28~250	cm
4116	浸液	Water Leak	true / false	
4117	液位	Liguid Level	0~500	cm
4118	氨气	NH3	0~100	ppm
4119	硫化氢	H2S	0~100	ppm
4120	瞬时流量	Flow Rate	0~65535	m3/h
4121	累计流量	Total Flow	0~6553599	m3
4122	氧气浓度	Oxygen Concentration	0~25	%vol
4123	水质电导率	Water Eletrical Conductivity	0~20000	us/cm
4124	水质温度	Water Temperature	-40~80	℃
4125	土壤热通量	Soil Heat Flux	-500~500	W/㎡

4126	日照时数	Sunshine Duration	0~24	h
4127	太阳总辐射	Total Solar Radiation	0~5000	W/㎡
4128	水面蒸发量	Water Surface Evaporation	0~200	mm
4129	光合有效辐射	Photosynthetically Active Radiation(PAR)	0～5000	umol/㎡ s
4131	响度	Volume	0~100	dB
4133	土壤张力	Soil Tension	-100~0	kPa
4134	盐度	Salinity	0~20000	mg/L
4135	总溶解固体	TDS	0~20000	mg/L
4136	叶面温度	Leaf Temperature	-40~85	℃
4137	叶面湿度	Leaf Wetness	0~100	%
4146	PM2.5	PM2.5	0~1000	ug/m³
4147	PM10	PM10	0~2000	ug/m³

Seeed will maintain the measurement IDs increasingly, and publish via the URL https://sensecap-statics.seeed.cn/hardware/lorapp/httpserver/src/constants/sensor-name-lang-dictionary.json . Please obtain the latest definitions of measurement IDs from the above address since this documentation may not keep up to date very tightly.

5.2 Sensor Types

Sensor Type	Sensor Name(Chinese)	Sensor Name(English)	Measurement IDs
1001	空气温湿度传感器	Air Temperature and Humidity Sensor	4097，4098
1003	光照强度传感器	Light Intensity Sensor	4099
1004	二氧化碳传感器	CO2 Sensor	4100
1005	气压传感器	Barometric Pressure Sensor	4101
1006	土壤温湿度传感器	Soil Moisture and Temperature Sensor	4102，4103
1008	风向传感器	Wind Direction Sensor	4104
1009	风速传感器	Wind Speed Sensor	4105
100A	pH 传感器	pH Sensor	4106
100B	光通量传感器	PAR Sensor	4107
100C	电导传感器	EC Sensor	4108
100D	溶解氧传感器	DO(Dissolved Oxygen) Sensor	4109
100E	土壤含水量温度电导传感器	Soil Temperature, VWC & EC Sensor	4110，4111，4112
1011	雨量传感器	Rain Gauge	4113

1013	超声波测距传感器	Ultrasonic Distance Sensor	4115
1014	浸液传感器	Water Leak Detector	4116
1015	液位传感器	Liguid Level Sensor	4117
2001	RS485 五合一传感器(类型-A)	RS485 Five-Elememt Sensor(Type-A)	4097，4098，4101，4104，4105
2002	RS485 三合一传感器(类型-A)	RS485 Three-Elememt Sensor(Type-A)	4097，4098，4101
2003	RS485 四合一传感器(类型-A)	RS485 Four-Elememt Sensor(Type-A)	4097，4098，4099，4101
2004	RS485 氨气温湿度传感器(类型-A)	RS485 NH3 Temperature Humidity Sensor(Type-A)	4097，4098，4118
2005	RS485 硫化氢温湿度传感器(类型-A)	RS485 H2S Temperature Humidity Sensor(Type-A)	4097，4098，4119
2006	RS485pH 传感器(类型-A)	RS485 pH Sensor(Type-A)	4106
2007	RS485 土壤水分温度传感器(类型-A)	RS485 VWC Temperature Sensor(Type-A)	4112， 4110
2008	RS485 土壤水分温度电导率传感器(类型-A)	RS485 VWC Temperature EC Sensor(Type-A)	4112， 4110，4111
2009	RS485 涡轮流量计(类型-A)	RS485 Turbine Flowmeter Sensor(Type-A)	4120，4121
200A	RS485 七合一传感器(类型-A)	RS485 Seven-Elememt Sensor(Type-A)	4097，4098，4099，4101，4104，4105，4113
200B	RS485 溶解氧传感器(类型-A)	RS485 Dissolved Oxygen Sensor(Type-A)	4109
200C	RS485 液位传感器(类型-A)	RS485 Liguid Level Sensor(Type-A)	4117
200D	RS485 氧气传感器(类型-A)	RS485 Oxygen Sensor(Type-A)	4122
200E	RS485 水质温度电导传感器(类型-A)	RS485 Water Temperature EC Sensor(Type-A)	4123，4124
200F	RS485 土壤热通量传感器(类型-A)	RS485 Water Temperature EC Sensor(Type-A)英文名不能修改就要新建一个 ID 了	4125
2010	RS485 日照时数传感器(类型-A)	RS485 Sunshine Duration Sensor(Type-A)	4126
2011	RS485 太阳总辐射传感器(类型-A)	RS485 Total Solar Radiation Sensor(Type-A)	4127

2012	RS485 水面蒸发传感器(类型-A)	RS485 Water Surface Evaporation Sensor(Type-A)	4128
2013	RS485 光合有效辐射传感器(类型-A)	RS485 PAR Sensor(Type-A)	4129
2014	RS485 水质温度溶氧传感器(类型-A)	RS485 Temperature and Dissolved Oxygen Sensor(Type-A)	4124, 4109
2015	RS485 土壤热通量传感器(类型-A)	RS485 Soil Heat Flux Sensor(Type-A)	4125
2011	RS485 太阳总辐射传感器(类型-A)	RS485 Total Solar Radiation Sensor(Type-A)	4127
2012	RS485 水面蒸发传感器(类型-A)	RS485 Water Surface Evaporation Sensor(Type-A)	4128
2013	RS485 光合有效辐射传感器(类型-A)	RS485 PAR Sensor(Type-A)	4129
2014	RS485 水质温度溶氧传感器(类型-A)	RS485 Temperature and Dissolved Oxygen Sensor(Type-A)	4124, 4109
2015	RS485 土壤热通量传感器(类型-A)	RS485 Soil Heat Flux Sensor(Type-A)	4125

Versions

Version	Date	Changes	Author
2.0	2021-4-14	Separate from the original combined documentation (v1.x)	Jack