AIR QUALITY SENSOR

Configuration

Rob Miles
Version 1.1

Contents

Settings Storage

The settings for device are stored internally. The device can be configured for multiple WiFi networks, MQTT connections over secure or insecure sockets and LoRa connections authenticated by OTA (over the air) or ABP (access by personalisation).

It is also possible to enable or disable different hardware elements. It is possible to control WiFi, MQTT, LoRa, Real time clock and GPS connections in this way.

Sending Commands

The host sends the device commands to read and write settings in the device. The configuration messages can be sent via the serial port or over an MQTT connection to a designated endpoint.

Every command will receive an acknowledgement. Note that not all configuration commands take effect immediately. Some require the device to be reset for the new setting to take effect. There is a remote command that can be used to force a reset of the device.

Serial connection

You can use a terminal program (for example PuTTY or the Arduino IE Serial Monitor) to communicate with a sensor connected to a host computer via the serial port. The baud rate is 115200 and data is sent with with 8 data bits and no parity. A command can be terminated by a carriage return (0x0D) or a line feed (0x0A). Acknowledgements to each command are sent back to the serial port.

MQTT connection

MQTT messages sent to the command topic will be interpreted as configuration commands. The command topic can be set the mqtt settings. The default command topic is:

```
airquality/commands
```

When a command has been performed the result is published to the mqtt publish topic. The default publish topic is:

```
airquality/data
```

Command Structure

Each command is a json snippet with the following overall structure:

```
{"v":ver,"t":"target","c":"family","o":"option","val":value,"seq":num}
```

The **v** element of a command is the version of the protocol. The ***ver*** value gives the protocol version expressed as an integer. At present there is only one version of the protocol, this is version 1. If an incorrect version number is used by the host the device will reply with an error message which includes the version number of the protocol in use by that device.

The **t** element of a command identifies the target device for the command. Each device has a unique name. The ***target*** value gives the name of the target device. If this name does not match that in the device the command is rejected with an error. The only command that does not require a target value is the **getdevname** command which can be used to determine the name of a device. The default name of a device is **sensor01**.

The **c** element is the command family of the command. The ***family*** value gives the command family that this command is part of. There are four command families:

- node – commands to configure the node

- wifi – commands to configure wifi settings
- lora – commands to configure LoRa settings
- mqtt – commands to configure MQTT settings

The **o** element is the option of the command family that is in use. The ***option*** value of the command specifies the option from the command family which is being used.

The **val** element is provided to set the value of a given command option. The contents of ***value*** are provided to set the value of a given option. They may be numeric or a string. This element is optional for many commands. If the **val** element is left off the command is usually interpreted as a request to read the given setting.

The **seq** element is provided to allow the sender to uniquely identify a given command that has been issued. The contents of ***num*** are an integer sequence number which is set by the host. The reply message from the device will contain the same sequence number. This element is optional for all commands.

Reply structure

Each command will receive a reply from the device. The reply will contain an error number which will be zero if the command was completed successfully. Appendix 1 of this document, Message Numbers gives the meaning of each error number.

The reply may also contain values that have been requested, along with a sequence number if one was given in the command.

Sample Commands

Sending a setting value

The following command sets the text to be displayed on the top line of the sensor. The command is being sent to the device **sensor01**. It uses the **spltop** option in the **node** family command.

```
{"v":1, "t":"sensor01", "c":"node", "o":"spltop", "val":"Hello Mum"}
```

The sensor will set the top line of the display to "Hello Mum". There is no sequence number, so the node replies with a message that indicates that the command was successful:

```
{"error":0}
```

Reading a setting value

A host can read back the value of a setting by omitting the **val** element of the message:

```
{"v":1, "t":"sensor01", "c":"node", "o":"splbtm","seq":123}
```

This command contains a sequence number, which is echoed in the message from the sensor.

```
{"val":"Hello Mum","error":0,"seq":123}
```

Device names

Each sensor can have a unique name. This means that multiple sensors could be connected to the same MQTT endpoint or shared serial connection and could be uniquely addressed.

The default name of a sensor is **sensor01** which is set when the sensor is powered up for the first time.

Getting a device name

The node option **`getdevname`** gets the name of a sensor. It does not require a target device to be specified:

```
{"v":1, "c":"node", "o":"getdevname"}
```

The device will respond with its name:

```
{"nodename":"sensor01","error":0}
```

Setting a device name

To set a device name you can use the **`devname`** command option:

```
{"v":1, "t":"sensor01", "c":"node", "o":"devname", "val":"Ferens"}
```

This would set the device name for the sensor to "Ferens".

Note that the device name is only used when processing incoming commands. It is independent of the MQTT sensor name. If you have no need to direct commands to different sensors you can leave the setting at **`sensor01`**.

Protocol Versioning

Commands are pre-ceded by a version number. A given device uses a version of the protocol and will reject any messages with a different version number. You can find the protocol version in a device by using the ver command. This command does not require a version number or a target:

```
{"v":1, "c":"node", "o":"ver"}
```

The reply to this command is the version number of the command protocol in this sensor:

```
{"version":1,"error":0}
```

Get Device Name

Gets the name of the device. This is the name that is used to send commands to the device. Note that this is not the MQTT or LoRa name of the device.

```
{"v":1, "c":"node", "o":"getdevname"}
```

The reply is the name of the device.

```
{"nodename":sensor01,"error":0}
```

Set the Device Name

Sets the name of the device. The name of the device is used to identify commands that are to be performed by this device. It is **not** the same as the MQTT name of the device, which is set using an MQTT command.

```
{"v":1, "t":"sensor01","c":"node","o":"devname","val":"sensor02"}
```

The above command would change the name of the device from `sensor01` to `sensor02`.

MQTT Commands

This command family can be used to manage a connection to an MQTT server. The current value of the setting can be read back from the sensor by leaving out the “v” element.

Set MQTT status

Enables or disables the MQTT system. When on the sensor will connect to the MQTT server on powerup using the stored connection settings. If the server is not found or the WiFi is disconnected the sensor will repeatedly retry the connection. The MQTT system can be on or off.

The sensor must be reset if this setting is changed from off to on.

```
{"v":1, "t" : "Sensor01", "c" : "mqtt", "o" : "state", "v" : "on"}
```

```
{"v":1, "t" : "Sensor01", "c" : "mqtt", "o" : "state", "v" : "off"}
```

Force MQTT send

Forces the transmission of an MQTT message. If the air quality sensor is turned off it will be powered up and the results averaged before transmission. This transmission is made irrespective of any timed transmissions.

```
{"v":1, "t" : "Sensor01", "c" : "mqtt", "o" : "send"}
```

Set MQTT transmission gap

Sets the gap between successive transmissions. The value is set in seconds. The minimum gap between MQTT transmissions is 60 seconds and the maximum is 30000

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "gap", "val":600}
```

Set MQTT transmission retry time

If the MQTT transmission fails the sensor will retry the transmission. This command sets the time interval before the retry is attempted. The interval is given in seconds. The minimum interval is 1 second and the maximum is 3000

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "retry", "val":20}
```

Set MQTT publish topic

This sets the topic on the MQTT server to which the sensor will send reading values.

```
{"v":1, "t":"sensor01", "c":"mqtt", "o":"publish", "val":"sensor01/data"}
```

Set MQTT command topic

This sets the topic on the MQTT server which the sensor will subscribe to for the reception of configuration commands.

```
{"v":1, "t":"sensor01", "c":"mqtt", "o":"subscribe", "val":"sensor01/commands"}
```

Commands will be acted on by the sensor upon receipt, and responses will be sent to the publish topic.

Set MQTT device name

This sets the MQTT device name for this sensor. Note that this can be different from the command name. although both are set to “sensor01” when the device is initialised.

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "id", "val":"sensor01" }
```

Set the MQTT host address

This sets the address of the server to be used for MQTT.

```
{"v":1,"t":"sensor01", "c":"mqtt", "o":"host","val":"mqtt.connectedhumber.org"}
```

Set the MQTT host username

This sets the username to be used on the MQTT host.

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "user", "val":"username" }
```

Set the MQTT host password

This sets the password to be used on the MQTT host.

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "pwd", "val":"123456" }
```

Set the MQTT host port

This sets the MQTT port to be used on the host. Note that there are different port numbers for secure sockets and open socket connections. Use secure sockets when connecting to an Azure IoT hub MQTT server:

- open sockets 1883
- secures sockets 8883

```
{"v":1, "t" : "sensor01", "c" : "mqtt", "o" : "port", "val":1883 }
```

LoRa Commands

These commands can be used to configure and test a LoRa connection.

Force LoRa send

This command forces the LoRa service to send a packet.

```
{"v":1, "t" : "Sensor01", "c" : "lora", "o" : "send"}
```

If the air quality sensor is turned off it will be powered up and the results averaged before transmission. This transmission is made irrespective of any timed transmissions.

Force LoRa test

This command forces the LoRa service to send a test packet instantly. The most recently loaded values are sent immediately.

```
{"v":1, "t" : "Sensor01", "c" : "lora", "o" : "test"}
```

Set LoRa status

Enables or disables the LoRa system. When on the sensor will connect to the LoRa gateway on powerup (if OTA authentication is being used) using the stored connection settings. The LoRa system can be on or off.

The sensor must be reset if this setting is changed from off to on.

```
{"v":1, "t" : "Sensor01", "c" : "lora", "o" : "state", "val" : "on"}
```

```
{"v":1, "t" : "Sensor01", "c" : "lora", "o" : "state", "val" : "off"}
```

Set LoRa transmission gap

Sets the gap between successive LoRa transmissions. The value is set in seconds. The minimum gap between transmissions is 60 seconds and the maximum is 30000

```
{"v":1, "t" : "sensor01", "c" : "lora", "o" : "gap", "val":600}
```

Setting LoRa for Authorisation by Personalisation (ABP)

When using LoRa "Authorisation By Personalisation" (ABP) a sensor must be configured with the application key, network key and device ID. These are supplied as sequences of HEX digits. They are normally obtained from The Things Network (TTN) device configuration page for the application within which the sensor is to be used.

```
{"v":1, "t" : "sensor01", "c" : "lora", "o" : "abpapp", "val":"AD7A768C296F2CA5C4C03E85862AE688"}
```

The above command sets the application key for a LoRa connection.

```
{"v":1, "t" : "sensor01", "c" : "lora", "o" : "abpnwk", "val":"117490D907331B077C5DAE9777FFE43C"}
```

The above command sets the network key for a LoRa connection.

```
{"v":1, "t" : "sensor01", "c" : "lora", "o" : "abpdev", "val":"26011BEE"}
```

The above command sets the device ID for a LoRa connection.

The sensor must be reset if any of these settings are changed.

WiFi configuration

The sensor can store SSID and password details for up to five connections. When the sensor loses a network connection it will repeatedly search for connections that it knows about.

Turn WiFi on and off

When the WiFi is turned off the sensor will not search for stations and any services that require a WiFi connection (for example MQTT) will not be started. There are two commands to turn the WiFi on and off:

```
{"v":1, "t" : "Sensor01", "c" : "wifi", "o" : "on"}
```

```
{"v":1, "t" : "Sensor01", "c" : "wifi", "o" : "off"}
```

Set WiFi SSID

Each wifi setting slot is identified by a value in the range 0 to 4. Attempts to set a value in a slot outside this range will result in an error. The setting in location 0 is initialised to a Connect Humber configuration connection.

```
{"v":1, "t":"sensor01", "c" : "wifi",  "o" : "ssid", "set":1, "val":"host"}
```

The above command would set the ssid for slot 1 to be **`host`**.

Set WiFi password

Each wifi setting also has a password which can be set:

```
{"v":1, "t" : "sensor01", "c" : "wifi", "o" : "pwd", "set":1, "val":"password"}
```

The above command would set the password for slot 1 to be **`password`**.

Note that it is not possible to read back the value of a WiFi password.

Node settings

The node settings can be used to configure a range of elements on the device. They can also trigger actions including reset the device and clear the device settings.

Get the protocol version

This command returns the version number of the protocol implemented by this sensor.

```
{"v":1, "c" : "node", "o" : "ver"}
```

Note that this command does not need a target device to be specified.

Get the device command name

This command returns the device name.

```
{"v":1, "c" : "node", "o" : "getdevname"}
```

Reset the device

This command resets the device. It is the equivalent of turning the device of and then on again. It has no effect on the contents of settings.

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "reset"}
```

Clear the device settings

This command clears all the device settings and returns them to the original values.

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "clear"}
```

Note that there is no confirmation dialog for this command, the settings will be cleared and the device then reset.

Set the device command name

This command sets the device name for use in commands. The device name is the name that will serve as the target for commands.

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "devname", "val":"sensor01"}
```

Set the banner display

The sensor can display a two-line banner as it is operating. This can be used to identify the device or allow a remote host to display messages on the sensor screen. There are two commands that can be used to set the top and bottom lines of this display:

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "spltop", "val":"Connected"}
```

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "splbtm", "val":"Humber"}
```

The two lines will set the top and bottom elements of the banner display.

Set the air quality sensor warmup time

The air quality sensor is switched off between readings. This reduces wear on the sensor and also reduces the amount of airflow through the sensor and hopefully the amount of dust that accumulates in it. This command allows the setting of the number of seconds before a reading is required that the sensor is powered up. If the warmup time is made greater than the gap between readings the sensor will be turned on continuously. The sensor is warmed up for LoRa or MQTT messages, depending on which is being delivered sooner.

```
{"v":1, "t" : "sensor01", "c" : "node", "o" : "warmup", "val":30}
```

Set the GPS receiver state

The sensor can be fitted with a GPS sensor that allows MQTT readings to be tagged with latitude and longitude values. If the GPS is not switched on sensor will not try to connect to a GPS receiver.

```
{"v":1, "t" : "Sensor01", "c" : "gps", "o" : "state", "val" : "on"}
{"v":1, "t" : "Sensor01", "c" : "gps", "o" : "state", "val" : "off"}
```

Set the Real Time Clock state

The sensor can be fitted with a real time clock chip that allows MQTT readings to be tagged with date and time. If the RTC is not switched on the sensor will not try to connect to a real time clock.

Set the display state

The sensor can be forced to display readings if neither MQTT or LoRa are enabled. In this state the air quality sensor runs permanently and the displays are always updated. Display state should only be switched on when you are testing the air quality and environmental sensors.

```
{"v":1, "t" : "Sensor01", "c" : "display", "o" : "state", "val" : "on"}
{"v":1, "t" : "Sensor01", "c" : "display", "o" : "state", "val" : "off"}
```

Message Numbers

If the contents of a command are inconsistent the value in the reply element in the message can be used to determine the error state. These are the message numbers and their meanings.

```
#define WORKED_OK 0
#define NUMERIC_VALUE_NOT_AN_INTEGER 1
#define NUMERIC_VALUE_BELOW_MINIMUM 2
#define NUMERIC_VALUE_ABOVE_MAXIMUM 3
#define INVALID_HEX_DIGIT_IN_VALUE 4
#define INCOMING_HEX_VALUE_TOO_BIG_FOR_BUFFER 5
#define VALUE_MISSING_OR_NOT_A_STRING 6
#define INCOMING_HEX_VALUE_IS_THE_WRONG_LENGTH 7
#define INVALID_OR_MISSING_TARGET_IN_RECEIVED_COMMAND 8
#define COMMAND_FOR_DIFFERENT_TARGET 9
#define INVALID_OR_MISSING_DEVICE_NAME 10
#define INVALID_DEVICE_NAME_FOR_MQTT_ON 11
#define STRING_VALUE_MISSING_OR_NOT_STRING 13
#define STRING_VALUE_TOO_LONG 14
#define INVALID_LORA_ACCESS_SETTING 15
#define INVALID_COMMAND_NAME 16
#define JSON_COMMAND_COULD_NOT_BE_PARSED 17
#define JSON_COMMAND_MISSING_VERSION 18
#define JSON_COMMAND_MISSING_COMMAND 19
#define JSON_COMMAND_MISSING_OPTION 20
#define INVALID_MQTT_STATUS_SETTING 21
#define INVALID_LORA_STATUS_SETTING 22
#define MISSING_WIFI_SETTING_NUMBER 23
#define INVALID_WIFI_SETTING_NUMBER 24
#define INVALID_GPS_STATUS_SETTING 25
#define INVALID_RTC_STATUS_SETTING 26
#define INVALID_DISPLAY_STATUS_SETTING 27
```