Ammonia Sensor AMON V2.0

User manual

December 2025 - Anemon Sensors

1. General overview

- Sensor name: AMON – version 2.0
- Type: Resistive sensor for detecting and measuring ammonia in gas phase (NH_3)
- Field of application: environmental monitoring, indoor and outdoor air quality, individual exposure
- Developed by: Anemon Sensors

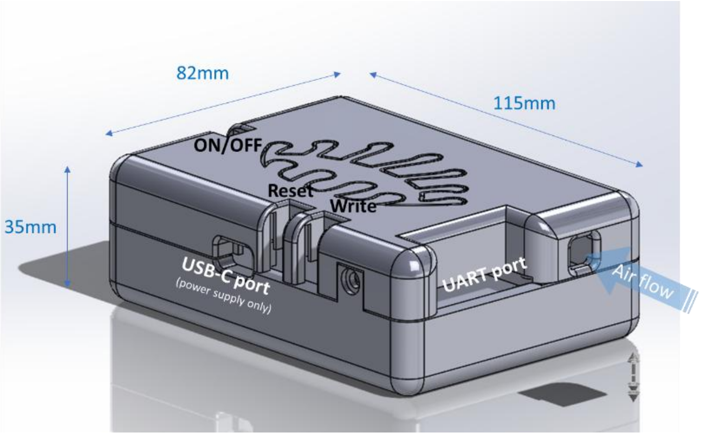
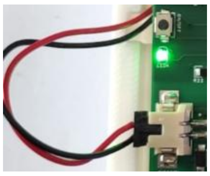


Figure 1 - AMON v2 sensor

2. Starting-up AMON v2 sensor

i. Connect the battery and ensure it is charged by connecting the USB-C cable if necessary:
    - Blue LED: USB-C cable connected
    - Red LED: battery charging
    - Orange LED: battery charged.

ii. To turn the module on and off, press and hold the ON/OFF button. When the sensor is ON, the LED near the ON/OFF button is green.



iii. Communication mode for data retrieval:

- For Wi-Fi transmission:

A shared access point must be created with the computer that will perform the acquisition. This computer must be connected on the same Wi-Fi network as AMON v2 sensor.

Arduino code (ESP32_AMON.ino) is provided to upload your network parameters into ESP32 micro-controller of AMON sensor. See Appendix 1 for complete instructions.
Use the executable software AmonDashboard.exe provided to monitor the data. See Appendix 2 for instructions on installing and using the dashboard. This software is only valid for Wi-Fi communication.
Files ESP32_AMON.ino and AmonDashboard.exe can be downloaded on our website www.anemon-sensors.com\download

- For serial transmission via USB UART (always active, cable provided):

UART transmission is permanently enabled regardless of Wi-Fi configuration.
Connect the supplied UART-USB C cable provided and open a serial communication terminal. Select the appropriate port parameters (115200 baud). The data is presented in a single line corresponding to the ammonia data, the time in ms, the local card time, the raw resistance of the sensor, the measurement frequency, the temperature in °C, the relative humidity in %, the pressure in Pascal, the battery voltage, the AMON module SOC address, and the AMON module ID.

The firmware embeds real-time signal correction and NH_3 estimation.

UART output format (115200 baud, 8N1):

```
NH3,<ts_ms>,<rtc_iso>,<r2_raw_ohm>,<r2_deriv_corr_ohm>,<r2_kalman_corr_ohm>
,<nh3_kal_ppb>,<nh3_deriv_ppb>,<freq_hz>,<kal_a0>,<kal_a1>,<t_c>,<rh_pct>,<
p_hpa>,<vbat>,<soc>,<deviceId>
```

Example:

```
NH3,123456,"2026-01-12T14:22:08",1520.3,1488.1,1495.6,85.2,92.4,123.45,-12.
3,0.87,22.4,45.3,1012.8,4.10,87.5,ESP32_12AB34CD5678
```

See Appendix 1 – section 8 for instructions specifically for wired communication (comm port settings and returned data.

⚠ No dashboard is available in this operating mode. The software must be developed by the user.

i. Adjustment of ammonia range:

By default, the card is configured for sensor element resistance ranges of around 1kOhm and measured concentrations less than or equal to 1 ppm. This setting is made on the card using the selectors circled in yellow in Figure 2 and Figure 3. Only one of the switches must be in the ON position. Sensors are qualified while switch 3 is ON (Low resistance – NH_3 < 1 ppm; C = 47 nF)

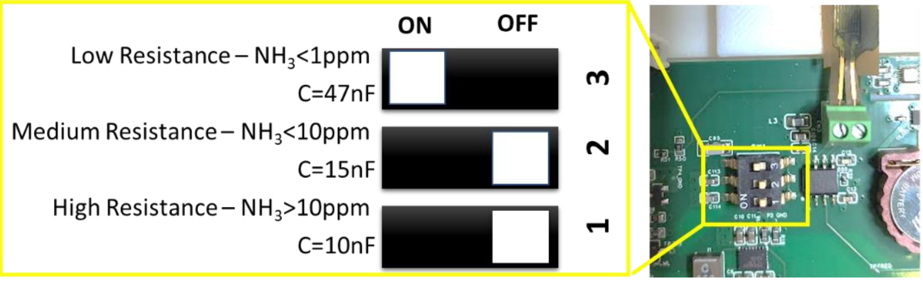

ON
OFF
Low Resistance – NH_3<1ppm
C=47nF
3
Medium Resistance – NH_3<10ppm
C=15nF
2
High Resistance – NH_3>10ppm
C=10nF
1

Figure 2 - AMON v2 ammonia range hardware configuration

3. Key technical specifications

Parameter	Typical value
NH_3 qualification range	10– 1000 ppb*
Resolution	10 ppb
Response time	< 60 s
Operating temperature	10°C à +50°C
Humidity range	20 – 95 % HR (non-condensing)
Power consumption	< 100mA
Power supply	5V USB-C Battery 3,3V-8Ah (>autonomy>70h)
Dimensions	115 mm × 35 mm × 82 mm
Weight	270g
Communication interface	Wi-Fi - UART
Service life	> 3 years**

* Sensors are expected to function for NH_3 concentrations above 1 000 ppb

** Observed service life in "clean" conditions (NH_3 concentrations < 1 000 ppb, 20 °C and relative humidity between 20 and 70%)

4. Hardware architecture

AMON v2 sensor is powered by an 8000mAh lithium-ion battery, which provides over 70 hours of operating time. It can be recharged via a USB-C port and can operate continuously when connected to the mains. An ON/OFF push button is used to start the sensor.

The operating principle is based on converting variations in the resistance of the sensing element into frequency variations using an oscillator module.

The sensing element is placed in a flow channel, with circulation provided by suction from a 2-wire fan. A temperature and relative humidity sensor is placed as close as possible to this channel to monitor these two environmental parameters.

An ESP 32 microcontroller collects the data and converts the signal for transmission via Wi-Fi and the UART serial port. The configuration of the ESP 32 microcontroller can be modified via the UART port. The control protocol is given in section 5.

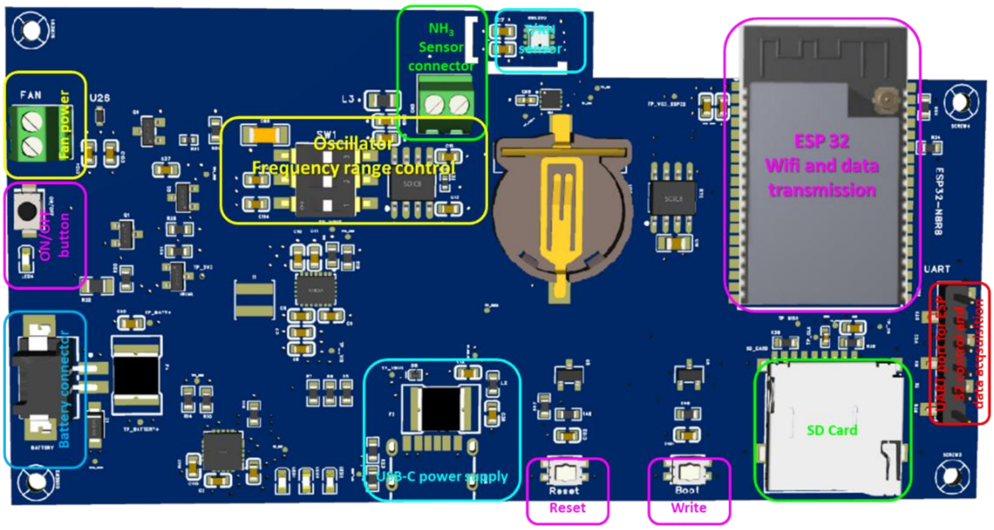


Figure 3 - AMON v2 sensor diagram

5. Recommendations for use

- It is recommended to use the AMON sensor horizontally with the UART connector facing upwards.
- The sensor is not IP65-rated, so it is essential to protect it from rain and condensation when used outdoors.
- A filter can be placed at the inlet of the module's 'sensitive surface' flow channel to remove particles. Please note that the correlation curves do not consider the presence of an inlet filter.
- Please also note that the AMON sensor does not function in ultra-dry atmospheres.

APPENDIX 1 – How to set ESP 32 for Wi-Fi communication

These instructions aim to explain how to compile and upload the ESP32 sketch in the Arduino IDE, then verify data transmission (UART is always active).

1) Required hardware and software

- AMON board
- Specific AMON USB-to-UART cable (no 5V wire)
- Arduino application IDE (recommended version 1.8.19)
- ESP32_AMON.ino provided on-line at https://anemon-sensors.com/download

Use only the UART-USB C cable provided, a classical UART-USB-C cable shall not be used unless the red wire is disconnected!

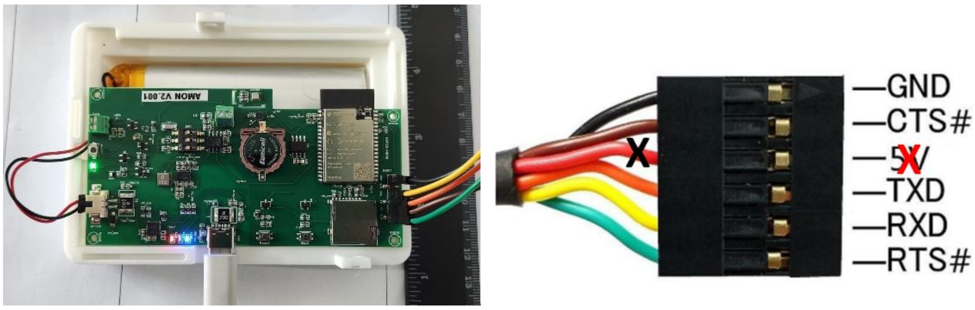


Figure 4 - AMON v2 UART connector settings: no 5V !

2) Install ESP32 support in the Arduino IDE (recommended version 1.8.19)

- Open File → Preferences.
- In “Additional Boards Manager URLs”, add:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- Go to Tools → Board → Boards Manager..., search for “esp32” (Espressif Systems) and install.
- Then select Tools → Board → ESP32 Arduino → esp32 wrover kit.

3) Library to install

- ArduinoWebsockets (author: Gil Maimon). IDE: Sketch → Include Library → Manage Libraries..., search for “ArduinoWebsockets” and install (≥ 0.5).
- Others (WiFi/WiFiUdp/Wire, PCNT, esp_timer, FreeRTOS) are provided by the ESP32 core.s

4) Open and prepare the sketch

- Open the .ino file (e.g. ESP32_AMON.ino provided on-line at https://anemon-sensors.com/download)
- Set at the top of the file:
    - Line 32: #define BROKER_MODE 0
        - 0 = WebSocket to a PC with static IP → adjust SERVER_IP, WS_PORT, WS_PATH.

    - 1 = UDP discovery + TCP client (no hardcoded IP).
  - Your Wi-Fi credentials:
    - Line 192: const char* STA_SSID = "YourSSID";
    - Line 193: const char* STA_PASS = "YourPassword";
- Save.

5) SAMPLING FREQUENCY

- The counting period is set by GATE_US (default 1,000,000 µs = 1 s).
- To change the pace:
- constexpr uint32_t GATE_US = 5000000; // 5 s

5) SAMPLING FREQUENCY

- The counting period is set by GATE_US (default 1,000,000 µs = 1 s).
- To change the pace:
- constexpr uint32_t GATE_US = 5000000; // 5 s

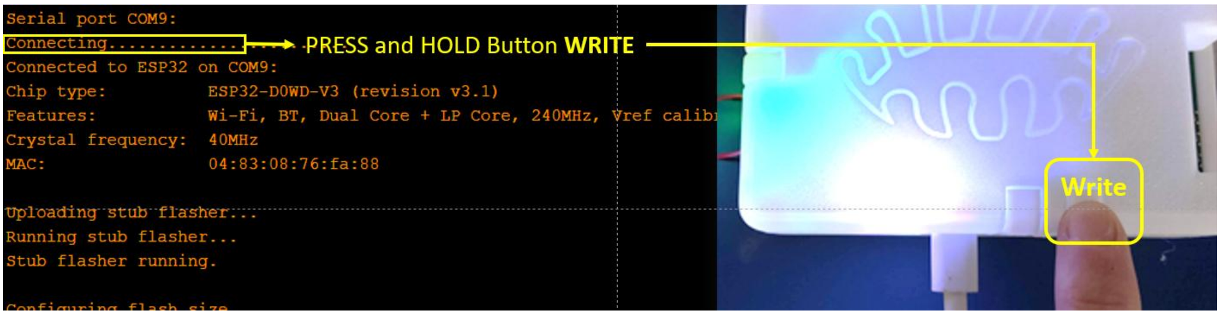
6) Compilation options

- Tools → Board: esp32 wrover kit
- Tools → Upload Speed: 921600 (or 460800 or less if unstable)
- Tools → Port: select the ESP32 COM port
- Leave partitions and PSRAM as default (except WROVER → PSRAM Enabled)

7) Upload

- Click "→" (Upload). The first time may take longer (dependency download).

⚠ After compiling, while the message "connecting" appears in Arduino Console, press and hold the button WRITE on AMON Sensor till the end of writing process like shown on next figure:



```
Serial port COM9:
Connecting..............
Connected to ESP32 on COM9:
Chip type:          ESP32-D0WD-V3 (revision v3.1)
Features:           Wi-Fi, BT, Dual Core + LP Core, 240MHz, Vref calibr
Crystal frequency:  40MHz
MAC:                04:83:08:76:fa:88

Uploading stub flasher...
Running stub flasher...
Stub flasher running.

Configuring flash size
```

Figure 5 - Arduino sketch downloading instructions: press and load "write" button

8) Check with Serial Monitor (115200 baud)

- Tools → Serial Monitor → 115200 baud
- Expected messages (examples):
  - "--- Scan I2C ---" and detected addresses (or "No device found" if nothing connected)
  - "PCNT OK (…)"
  - "deviceId = ESP32_XXXXXXXXXXXX"

- “Wi-Fi connection to … OK” + “ESP32 IP: …”
- In mode 0: “WS connect → ws://<PC_IP>:<PORT>/ws”

The UART format is always active for serial communication
Every second (if valid frequency):

The firmware now embeds real-time signal correction and NH_3 estimation.

UART output format (115200 baud, 8N1)

```
NH3,<ts_ms>,<rtc_iso>,<r2_raw_ohm>,<r2_deriv_corr_ohm>,<r2_kalman_corr_ohm>,
<nh3_kal_ppb>,<nh3_deriv_ppb>,<freq_hz>,<kal_a0>,<kal_a1>,<t_c>,<rh_pct>,<p_hpa
>,<vbat>,<soc>,<deviceId>
```

Example:

```
NH3,123456,"2026-01-12T14:22:08",1520.3,1488.1,1495.6,85.2,92.4,123.45,-12.3,0.87,22.4,
45.3,1012.8,4.10,87.5,ESP32_12AB34CD5678
```

Field description

Field	Description
ts_ms	Internal ESP32 timestamp (ms)
rtc_iso	RTC time (DS3231 or fallback)
r2_raw_ohm	Raw equivalent resistance of NH_3 sensor
r2_deriv_corr_ohm	Resistance corrected using derivative-based method
r2_kalman_corr_ohm	Resistance corrected using Kalman filter
nh3_kal_ppb	Estimated NH_3 concentration (ppb) from Kalman-corrected signal
nh3_deriv_ppb	Estimated NH_3 concentration (ppb) from derivative correction
freq_hz	Measured oscillator frequency
kal_a0, kal_a1	Kalman regression coefficients (offset and RH slope)
t_c	Temperature (°C)
rh_pct	Relative humidity (%)
p_hpa	Pressure (hPa)
vbat	Battery voltage (V)
soc	Battery state of charge (%)
deviceId	Unique ESP32 identifier

9) COMMON MODIFICATIONS

- Change Wi-Fi SSID/password (STA_SSID / STA_PASS)
- Switch to MODE 0 if on an enterprise network without multicast/broadcast
- Adjust GATE_US for faster/slower frequency

Network notes (optional)

- Mode 0: WebSocket → point SERVER_IP/WS_PORT/WS_PATH to the “broker” machine
- Mode 1: the ESP32 sends HELLO_NH3:<deviceId> via UDP broadcast (DISC_PORT). The broker machine replies OK_NH3:<port> and the ESP32 opens a TCP client to that port.

9) Embedded signal correction and NH_3 estimation

The ESP32 firmware embeds two complementary correction approaches:

9.1 Derivative-based correction (Anemon Sensors©)

The raw resistance R2 measured by the 555 oscillator circuit drifts over time due to adsorption/desorption dynamics: when NH3 concentration changes, the sensor does not respond instantaneously but follows an exponential relaxation curve. This creates a lag artifact that makes the resistance appear artificially lower (during a rise) or higher (during a fall) than the true equilibrium value.

To compensate, a sliding-window derivative correction is applied at every sample (1 Hz). The mean slope dR_mean is used to correct the raw resistance:

➔ if the slope >0, R_corr = (1 + |dR_mean|)ˆα × R_now (with α = 0.55 for a sub-linear, bounded amplification).

➔ If the slope <0, R_corr = β × (1 – |dR_mean|ˆα) × R_now (with β = 0.97 to avoid over-correction during descent.

The corrected value r2_deriv_corr_ohm is then converted to NH_3 concentration in ppb via a linear gain/offset calibration.

NH3 Calibration: Gain and Offset

The final conversion from corrected resistance to NH3 concentration (in ppb) is a simple linear equation applied to both output channels:

- nh3_deriv_ppb = NH3_GAIN × r2_deriv_corr_ohm + NH3_OFFSET
- nh3_kal_ppb = NH3_GAIN × r2_kalman_corr_ohm + NH3_OFFSET

The two calibration constants NH3_GAIN (slope, in ppb/Ω) and NH3_OFFSET (intercept, in ppb) are defined as compile-time constants near the top of the source file, in the "Correction dérivée" section:

  - cppconst float NH3_GAIN = 1.0f; // ppb per ohm — to be determined experimentally
  - const float NH3_OFFSET = 0.0f; // ppb zero offset — to be determined experimentally

Both are currently set to their identity values (gain = 1, offset = 0), meaning the output is expressed directly in ohms until a proper calibration is performed against a reference NH_3 source. NH_3_GAIN and NH_3_OFFSET are given in the calibration table provided by ANEMON Sensors. Update the constants and recompile.

9.2 Kalman-based correction (recommended)

A 2-state Kalman filter performs a linear regression of the sensor resistance versus relative humidity:

$$R2\_corr = R2\_raw - (a0 + a1 \times RH)$$

This approach improves baseline stability and allows direct conversion into NH_3 concentration (ppb).

The Kalman coefficients (a0, a1) are transmitted in real time for diagnostic and post-processing purposes.

APPENDIX 2 – How to install and configure AMON v2 DASHBOARD (PC)

The executable dashboard AmonDashboard.exe is available on our website www.anemon-sensors.com\download

1) QUICK START

- Double-click AmonDashboard.exe.
- If Windows asks, ALLOW network access (private AND public).
- The browser opens at: http://127.0.0.1:8065

2) WHAT THE APPLICATION DOES

- Displays live measurements (plots).
- Continuously logs incoming data into /data (CSV per device).
- Can also save daily files into /records when pressing "Start / Stop" in the interface.

3) WRITTEN FILES

- Live per device: data/live_<deviceId>.csv
- Columns: t_pc,ts_ms,freq_hz,r2_ohm,t_c,rh_pct,p_hpa,vbat,soc,deviceId,src
- (t_pc = sensor RTC time if available, otherwise PC time. src = ws/tcp/uart)
- Manual recording (Start / Stop button):
- <records>/<deviceId>/<YYYYMMDD>_<deviceId>.csv
- Columns: t_pc,ts_ms,freq_hz,r2_ohm,t_c,rh_pct,p_hpa,vbat,soc,deviceId

4) PORTS & NETWORK MODES

Recommended mode = MODE 1 (UDP Discovery + TCP)

- UDP DISCOVERY: port 50000
- TCP DATA: port 7000
- Mode 0 (optional) = WebSocket with fixed IP
  - WS_PORT: 80 (address configured on ESP32 only)

5) WINDOWS FIREWALL

Allow AmonDashboard.exe on:

- UDP 50000 (discovery)
- TCP 7000 (data stream)

6) CHANGING PORTS (OPTIONAL)

The executable reads environment variables:

- DASH_PORT : web server port (default 8065)
- WS_PORT : WebSocket port (for MODE 0)
- TCP_PORT : TCP port (for MODE 1) — default 7000
- DISC_PORT : UDP discovery port (for MODE 1) — default 50000

Examples (PowerShell, before launching the exe):

- setx DASH_PORT 8065
- setx TCP_PORT 7001

- setx DISC_PORT 50001

Or via a .bat file:

```
@echo off
set DASH_PORT=8065
set TCP_PORT=7000
set DISC_PORT=50000
start "" "%~dp0AmonDashboard.exe"
```

7) QUICK UI USAGE

- “Refresh devices”: rescans the live_*.csv files.
- “Clear view”: only shows points received afterwards.
- “Reset view”: cancels the effect of “Clear view”.
- “Start / Stop” + path: logs into /records.
- “Download CSV (view)”: exports the currently displayed data.
- Live status at the top (“🟢/🟠/🔴”) is based on the last received data (PC side).

8) DATA FORMAT (FOR EXTERNAL TOOLS)

- Network (JSON per message — MODE 0 WebSocket or MODE 1 TCP):

```
{
"deviceId":"ESP32_XXXX",
"ts_ms": 123456789,
"freq_hz": 123.456789,
"r2_ohm":  1234.567890,
"t_c":     21.5000,
"rh_pct":  45.2500,
"p_hpa":   1013.25,
"vbat":    4.115,
"soc":     87.50
}
```

- UART serial (Always active – 115200 8N1):

```
NH3,<ts_ms>,<rtc_iso>,<r2_ohm>,<freq_hz>,<t_c>,<rh_pct>,<p_hpa>,<vbat>,<soc>,<deviceId>
```

Example:

```
NH3,123456,"2025-10-31T10:55:02",1543.220581,123.456879,22.4375,45.3125,1012.85,4.102,87.45,ESP32_12AB34CD5678
```

9) QUICK TROUBLESHOOTING

- No “Discovery OK” on ESP32: UDP 50000 blocked by firewall.
- No TCP data stream: TCP 7000 blocked, or PC/ESP32 not on the same network.
- UI opens but no device: check that data/live_*.csv files appear after reception; otherwise check network/firewall.