SIE & Durability Semester Project

Final Report

June 8, 2025

Professor: Athanasios Nenes
Supervisors: MAKE lab managers : Marc, Willow, Stephane, Yvan, Sylvain
Students: Lorraine Naux & Matheo Godenzi

Prototyping of a Low-Tech, Portable Gas Monitoring Station

Abstract

This report presents the design, development, and first partial testing of a low-tech, portable gas monitoring station intended for deployment aboard sailboats during scientific expeditions. Developed as part of two joint semester projects (SIE and Sustainability projects) in collaboration with Sailowtech, the station aims to monitor key atmospheric pollutants—ozone (O_3), carbon monoxide (CO), and particulate matter (PM)—as well as solar irradiance, while operating under the demanding conditions of marine environments. The system emphasizes modularity, affordability, and repairability, incorporating off-the-shelf sensors, a Teensy 4.0 microcontroller, and an embedded data acquisition and transmission setup. The station features a cylindrical casing built from recycled materials, a customized internal skeleton for sensor placement, and a hybrid power system using both a rechargeable battery and NMEA2000 marine interface. Extensive effort was invested in mechanical design, sensor calibration, and signal correction—especially for environmental cross-sensitivities and motion-induced artifacts. Despite challenges in mechanical fitting, electronics integration, and sensor reliability that require remediation, the prototype aims to offer an affordable, scalable, and open-source solution for real-time environmental monitoring in remote and low-infrastructure settings.

Contents

1 Introduction

Air pollution, driven by anthropogenic emissions of reactive gases such as nitrogen dioxide (NO_2), sulfur dioxide (SO_2), carbon monoxide (CO), ozone (O_3), and volatile organic compounds (VOCs), poses serious risks to both public health and ecosystems [1, 2]. Monitoring these pollutants in remote or understudied regions—such as open oceans, coastal zones, harbors, and island environments—remains a significant scientific challenge due to the logistical and technical constraints of existing monitoring infrastructure.

Scientific expeditions aboard sailboats present a unique opportunity to collect high-resolution environmental data in marine and coastal environments, often far from established monitoring stations. However, the variability of meteorological conditions, limited onboard power availability, and space constraints necessitate the use of lightweight, low-power, and compact instrumentation.

This report presents the development of a prototype gas monitoring station designed primarily for deployment on sailboats during scientific expeditions. The system is intended to identify and quantify key atmospheric gases in real time and record data over long periods of time, with emphasis on portability, autonomy, and repairability. By integrating compact gas sensors, embedded computing, and data logging capabilities, the prototype aims to support in-situ measurements in remote environments and contribute to research in marine atmospheric chemistry, pollutant transport, and climate interactions [3, 4].

1.1 Context

This project was proposed by the *Sailowtech* [5] association and is realized by Lorraine Naux and Matheo Godenzi, as part of their sustainability semester project and SIE project respectively. Careful attention has been paid to minimizing the prototype's impact on both human society and the environment. The system is designed to be fully repairable and includes only essential electronic components, thereby reducing its embedded environmental footprint. In alignment with an open-source philosophy, all schematics, designs, and documentation will be posted in open source access on Sailowtech's github [6] to encourage transparency, reuse, and collaborative improvement.

1.2 State of the art

Air quality monitoring technologies have evolved significantly in recent years, driven by the increasing need for portable, real-time, and multi-gas sensing systems applicable across urban, industrial, and remote field environments. Modern gas monitoring devices range from compact handheld detectors to advanced multi-sensor platforms with integrated data logging and wireless capabilities.

Recent developments emphasize sensor modularity, low power consumption, and ease of maintenance. While high-end instruments offer robust performance and Bluetooth connectivity for data synchronization, cost-effective models often trade off features such as real-time communication and open repairability for affordability and simplicity [3, 7].

Device Name	**Price (CHF)**	**Battery**	**Repairability**	**Remote Access**	**Sensors**	**Features**	**Source**
Aeroqual Series 500	1'365	Not specified	Tool-less sensor swaps	Not available	O_3, CO, PM, VOCs, NO_2, Temp, RH	Data logging, 0–5V output	Aeroqual
Aeroqual Series 300	975	Not specified	Tool-less sensor swaps	Not available	O_3, CO, PM, VOCs, NO_2, Temp, RH	Min/max/avg display	Aeroqual
Dräger Pac 8000	Not listed	2 years (24/7)	Sensor/filter replaceable with tools	Bluetooth	O_3, CO, NO_2, H_2S, Cl_2, NH_3, PH_3	IP68 rating, event logging	Dräger
GasDog GD200-O3	815	Up to 100 hours	Return to manufacturer	Not available	O_3	Alarms, optional data logging, USB	GasDog
K-600 Detector	240	Not specified	Return to manufacturer	Not available	Custom (O_3, CO, PM, VOCs)	Compact design, USB data transfer	Oxidation Tech
Aeroqual Ranger	2'500	>20 hours	Hot-swap with pre-calibrated sensors	Bluetooth	O_3, CO, PM, VOCs, NO_2, Temp, RH	App sync, advanced logging	Gas Sensing

Table 1: Comparison of commercial gas monitoring devices.

Table 1 presents a comparative overview of several commercially available gas monitoring devices. Key parameters considered include price, sensor types, battery life, repairability, remote access, and data handling capabilities.

The Aeroqual Series 500 and 300 offer modular sensor options and are designed for ease of use with tool-less swaps, but lack wireless connectivity. These are suitable for semi-static monitoring environments where mobility and ruggedness are not critical. The Aeroqual Ranger represents a more advanced model with Bluetooth synchronization and hot-swappable sensors, but at a significantly higher price point.

The Dräger Pac 8000 stands out for its IP68 ruggedization and extended sensor lifetime, making it viable for industrial or harsh environments. Its limited flexibility in sensor replacement, however, could be a constraint in long-term field deployments where adaptability is key.

The GasDog GD200-O3 is focused on ozone detection with limited remote capabilities but boasts a relatively long battery life. The K-600 Detector offers customizable sensing but requires factory servicing, limiting its maintainability in remote or expedition contexts.

In the context of mobile scientific expeditions—especially maritime or sailboat-based platforms—key challenges include power autonomy, physical durability, and environmental resistance. The purpose of prototyping a new gas monitoring station is to find a different balance between remote accessibility, repairability and measurement precision. According to Balz Maag et al. the sensors are more error-prone than high-end sensing infrastructures[8], constituting a real engineering challenge to ensure data accuracy. Sensor calibration has proven to be effective to improve the data quality of low-cost sensors and maintain the reliability of long-term, distributed sensor deployments [8].

1.3 Original prototype

The initial concept for the gas monitoring station was developed during the previous semester. It was originally meant to be a weather station. Its outer casing was designed to be cylindrical in shape to ensure maximum compactness and save space on the boat. The electronic components would be housed inside this main cylindrical structure.

To allow ambient air to circulate while preventing water and salt from entering, a curved chimney system was designed for the top of the structure. This system consists of a smaller cylinder mounted on top of the main body and covered by an overlapping cap. This design allows airflow while effectively shielding the internal components from humidity and salt infiltration.

To support the structure and secure the first weather station prototype to the boat, two end caps would be positioned at the top and bottom of the cylinder. Five threaded rods would run through the end caps to hold everything firmly in place. Initially, the plan was to mount the weather station at the stern of the boat. The overall external structure is represented in figure 1.

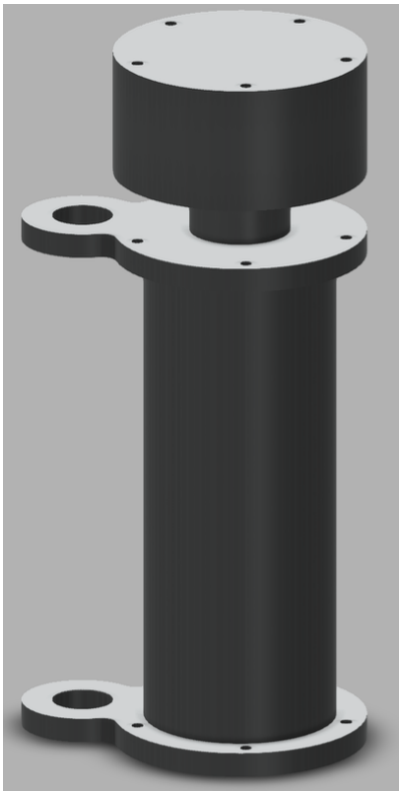


Figure 1: External casing of the first prototype imagined.

For the electronics, the initial plan included integrating a pressure, humidity, and temperature sensor (BME280), a pyranometer, a gyroscope, an SD card module, a NMEA2000 interface for accessing boat data, and a micro-

controller (Teensy 4.0). The complete circuit diagram incorporating all these components is shown in Figure 2.

To power the weather station in the absence of a NMEA connection, a 4S-2P rechargeable battery was assembled using refurbished cells. This battery configuration—four cells in series and two in parallel—provides a voltage range of approximately 13.7–16.8 V and a capacity of 1500 mAh. The output voltage of around 16 V is well-suited for powering the pyranometer.

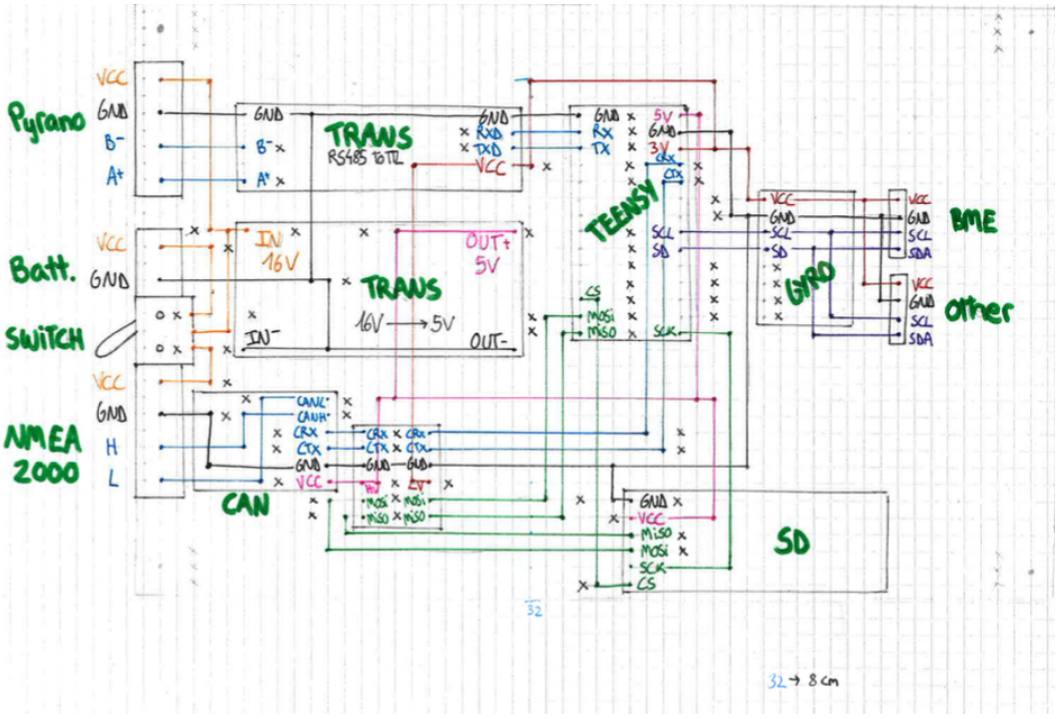


Figure 2: Scheme of the first strip board.

When the project resumed, the circuit was successfully tested on a strip board, but the external casing remained unbuilt.

2 Current prototype

The objective is to build a portable **gas monitoring station** capable of withstanding marine environmental conditions (mainly wind and salted water) while providing useful gas concentration and sunlight irradiance data for scientific studies and life aboard sailboats.

After identifying several challenges, the station's purpose was revised to emphasize the measurement of gas concentrations (CO and O_3), fine particles, and solar irradiance. It will therefore collect the following environmental data:

- Total solar radiation power [W/m2]
- Partial pressure of atmospheric pollutants and fine particles (CO and O3) [ppm].

The system is also designed to retrieve data from the boat itself, including:

- Wind direction and apparent force [kts]
- Boat speed [kts]
- GPS position [lat, long]

To ensure consistent and continuous data collection, the gas monitoring station has been designed to operate without interruption. During navigation, it can be powered directly via the boat's NMEA cable. When docked or in port, the integrated battery enables the station to operate independently of the NMEA connection. The battery makes it possible to collect environmental data in coastal or port areas, where pollution levels may be higher. The collected data can be accessed via a web interface and downloaded in CSV format, as well as being stored locally on an SD card. Figure 3 shows the overall operation of the station and the process of data collection (see Section 4 for further information).

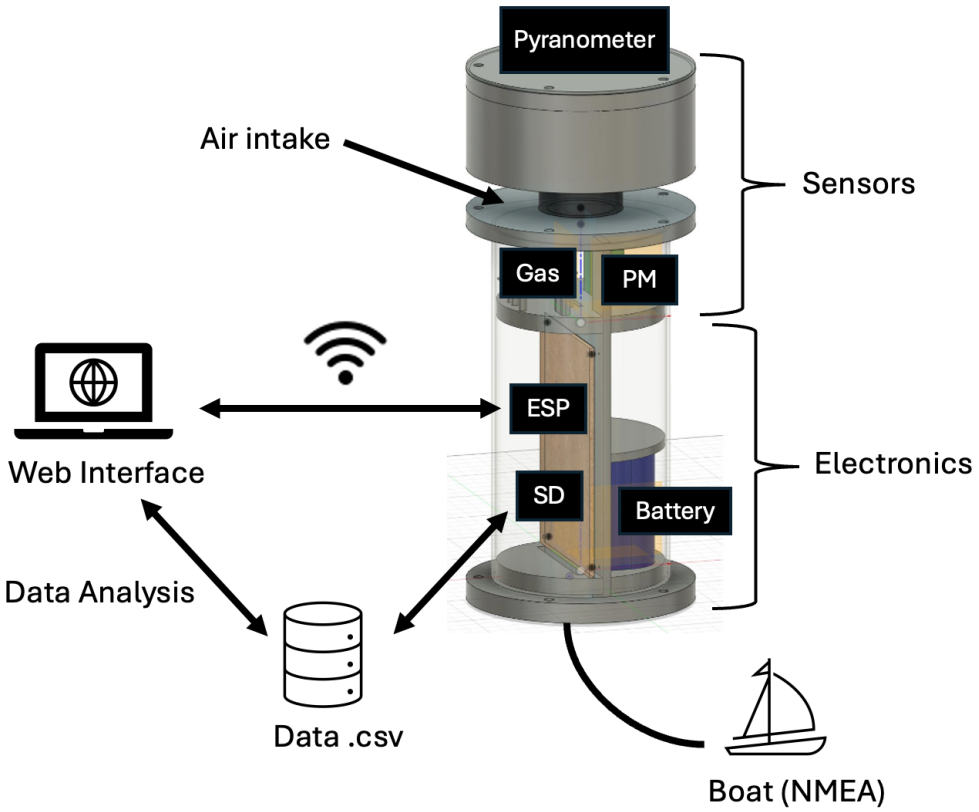


Figure 3: General operation of the gas monitoring station.

2.1 Casing

2.1.1 External case

The original outer casing design was retained to allow testing of this first version of the weather station, which had not yet been physically built. The only modification was the simplification of the upper and lower covers into a circular shape. Since there is no universally optimal location on a boat to mount the station, the design was made intentionally adaptable to accommodate various installation points.

Figure 4 shows the external casing, which was designed using Fusion 360 software. It consists of several components with different diameters, detailed in Table 2.

The main body of the station is a cylindrical shell made from recycled PVC, which houses and protects all electronic components. At each end of the cylinder, polycarbonate covers enclose the system and maintain structural stability through five metal threaded rods. As the design of an internal skeleton had to be imagined, the design of the lower and upper covers had to be modified. These pieces are represented in figure 5. A smaller PVC cylinder is mounted on the upper cover to allow airflow into the station. Above it, a cap is installed to enable ventilation while preventing water and moisture ingress. The cap is also held in place by five metal rods, which allow its height to be easily adjusted.

The cylindrical parts were fabricated from recycled PVC sourced from the SKIL, while the end covers were made of polycarbonate for their robustness, UV resistance, and thermal stability. As suitable cylindrical components were not readily available for the cap, it was custom-designed and 3D-printed using PETG filament, chosen for its durability, UV resistance, and suitability for outdoor environments.

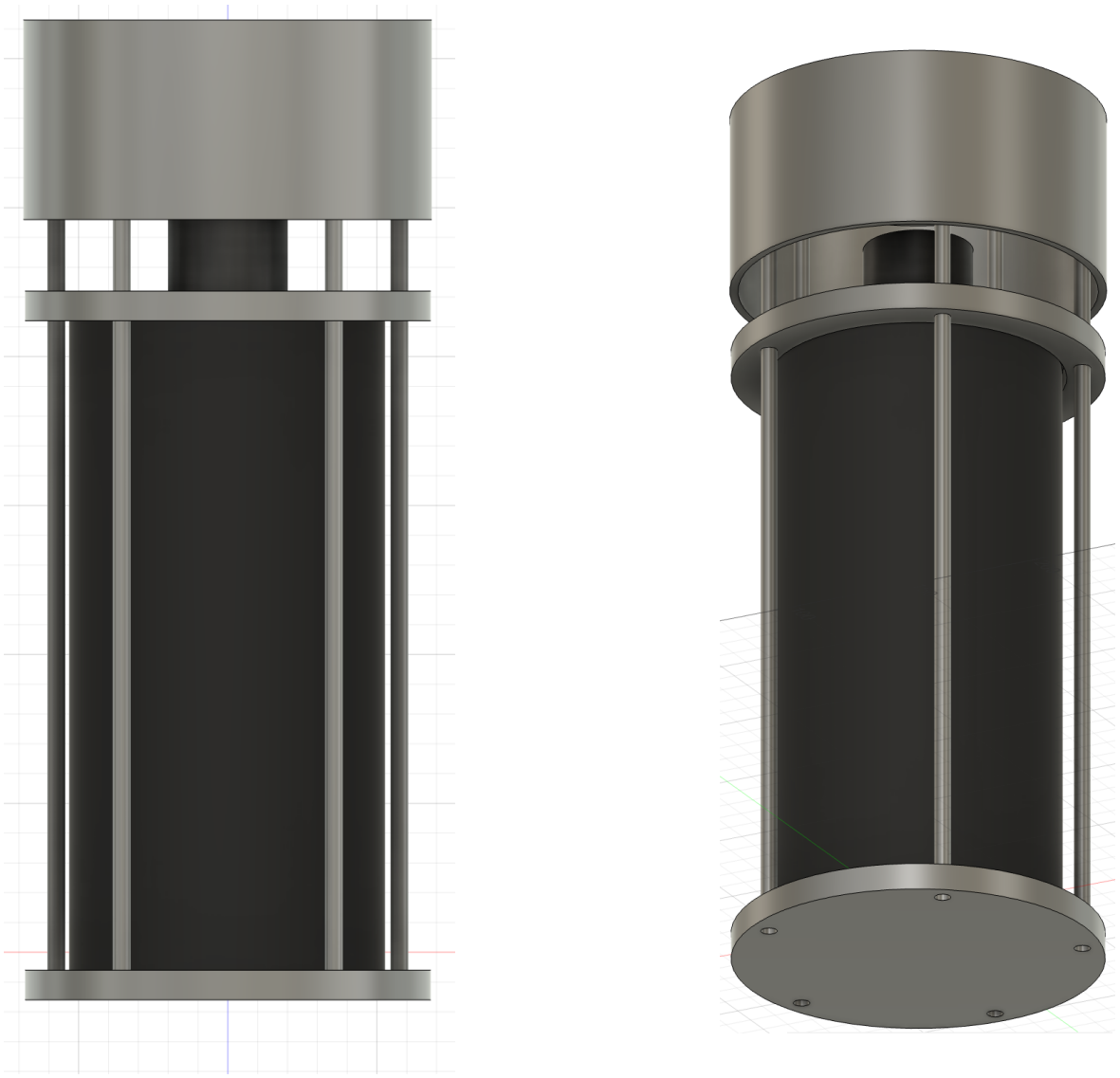


Figure 4: Front and and transversal view of the external case.

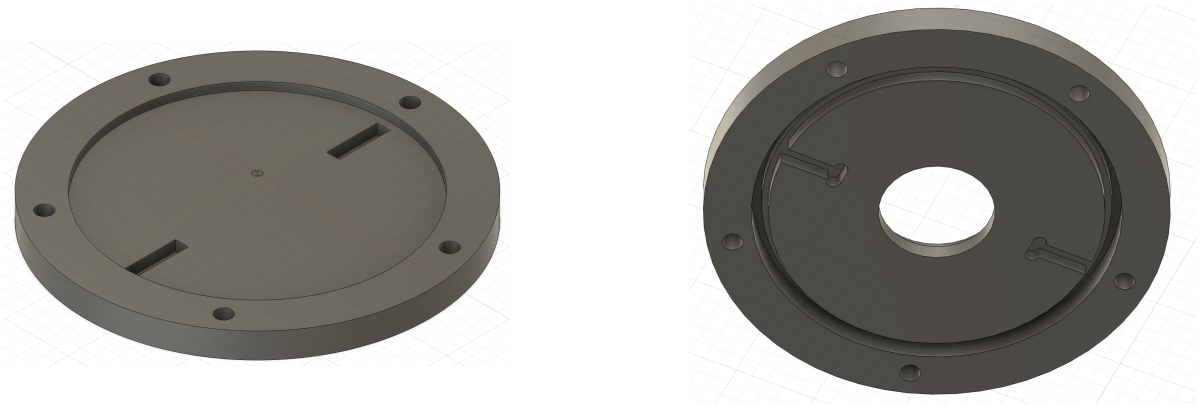


Figure 5: Transversal views of the upper and lower covers.

	Outer Diameter (mm)	Inner Diameter (mm)	Height (mm)	Material	Construction
Main Cylinder	106	102	222	PVC	sawn only
Small Cylinder	40	34	47	PVC	sawn only
Cap	-	-	67	PETG	3D-printed
Upper & lower Cover	136	106	10	Polycarbonate	CNC

Table 2: Dimensions of the materials used for the external structure of the box.

2.1.2 Internal skeleton

An internal support structure was necessary to securely hold the electronic components, battery and various sensors. This internal skeleton consists of two circular plates (one on the top and one on the bottom) and a central vertical panel that is used to mount the electronic board. The first version of the skeleton was made from recycled PVC using a CNC cutter. However, to simplify testing of sensor mounts, the final prototype was manufactured from MDF using a laser cutter, which is a faster and more accessible technique than CNC machining.

In order to accommodate the battery within this structure, the central panel was redesigned to allow for better integration. As shown in Figure 6, a rectangular cut-out matching the battery's dimensions was added to the centre, along with top and bottom brackets to hold the battery vertically. To provide horizontal stability, side notches allow a strap to be used to secure the battery firmly in place. An additional protective plate is inserted between the battery and its PCB to safeguard both components.

Figure 6: Transversal views of the two sides of the internal skeleton.

2.1.3 Component placement

Figure 7 shows the layout of the various electronic components attached to the internal skeleton.

First, the pyranometer (1) is placed on top of the cap and secured with screws. All the other sensors are placed at the top of the upper circle, just below the air intake through the small cylinder. The ozone and carbon monoxide sensors (3) and (4) are positioned using brass collars and screws inserted into the circle. The fine-particle sensor (2) is placed on the circle thanks to a notch integrated into its design. Finally, the BME sensor (temperature, humidity and pressure) is placed in this chamber and attached to the central part with screws. The PCB (6) is attached to the central part of the skeleton using colonettes and screws. The battery (5) is attached using the fastening system explained above.

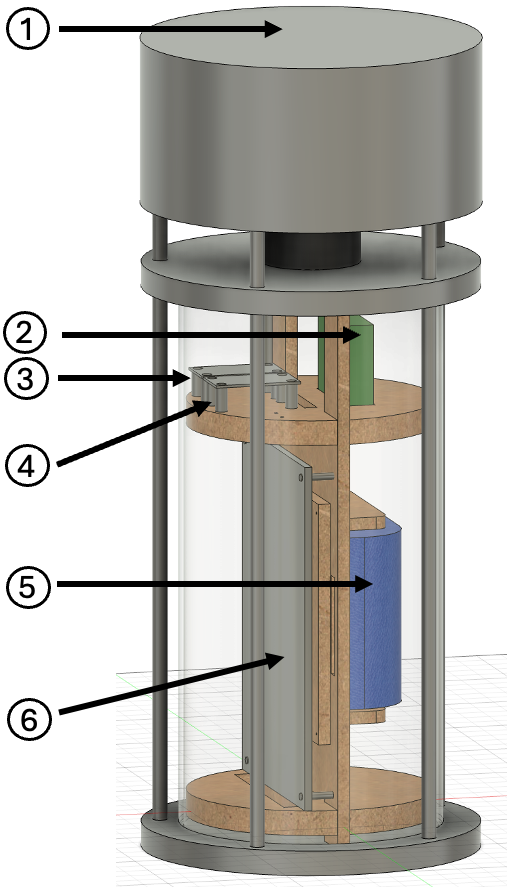


Figure 7: Transversal view of the gas monitoring station with the electronic components : (1) Pyranometer placed on the top, (2) PM sensor, (3) CO sensor, (4) O3 sensor, (5) battery, (6) PCB.

2.1.4 Case insulation

One of the biggest challenges of this case is protecting it from water and salt infiltration as much as possible. To address this issue, a number of measures have been implemented:

- o-rings have been added to the covers to improve sealing.
- insulation gaskets have been added at the sensor cable entry points to prevent infiltration.
- the cylinder has been drilled in several places to control the internal airflow and prevent the accumulation of air and moisture in one area.
- application of white paint to the main cylinder to promote thermal reflection and better air circulation.

Once the watertightness of this case has been tested, one of the possible evolutions, if the infiltration of water and salt is too great, is the redesing of the small cylinder through which the air passes. Indeed, it could have a shape inspired by tesla valves [9], but only in the blocking sense of their use. In this way, air could pass through unhindered, while water would be trapped in the corners. This might solve the problem, but would make the part more complex to design.

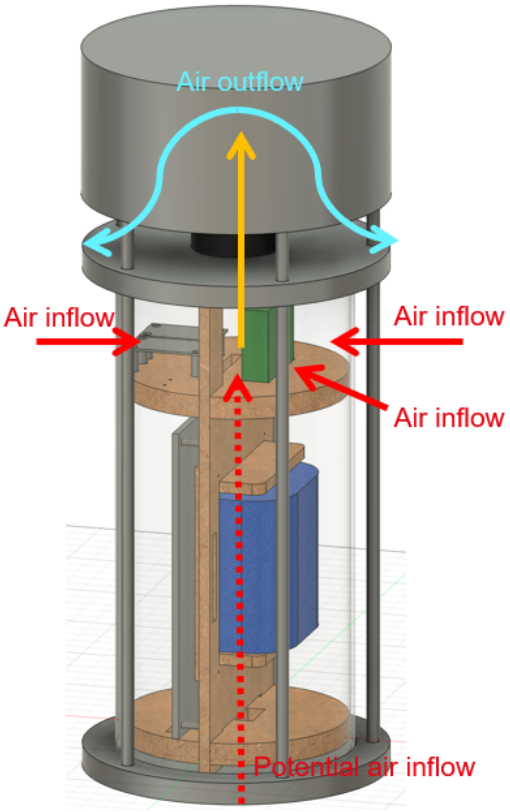
2.1.5 Air circulation

Air circulation is needed to bring about potential excesses in pollution and to ensure that the station keeps monitoring the ambient air nearby. To facilitate airflow, holes were created on the sides of the casing, at the sensors' level. To protect the sensors from any potential water, they were dug upwards, diagonally. Thus, rain or even splashes of water would have to go back up to enter the station.

As shown in Figure 8a, the lower section of the casing was painted white, while the upper cap was left black. This design promotes a passive convective airflow, often referred to as the *chimney effect*, by creating a controlled temperature gradient between the two sections. White surfaces reflect a large portion of incident solar radiation, particularly in the visible spectrum, whereas black surfaces absorb most of it [10, 11]. Due to its higher absorptivity (close to 1), the black cap undergoes a more significant radiative energy gain under sunlight, converting solar radiation into thermal energy. This results in an increase in surface temperature, which then heats the surrounding air through conduction and re-radiation in the infrared spectrum [12]. In contrast, the white-painted lower section reflects much of the incident light, leading to relatively less heat absorption and thus cooler internal air. The resulting temperature difference between the upper and lower sections causes the warmer, less dense air to rise and exit through vents in the black cap. As illustrated in Figure 8b, this upward movement induces a pressure differential that draws cooler ambient air into the enclosure through the openings near the base, maintaining passive airflow within the sensor chamber.

Additionally, as the boat will be constantly moving, air will be pushed into and out of the station mechanically as well.

(a) External aspect of the station, white below and black above



(b) air circulation scheme

Figure 8: Physical principles enabling passive ventilation.

2.2 Electronics

Now that the outer casing is fully described, let's dive into the electronics of the gas monitoring station prototype. Here's a summary of all electronic elements that make it up:

Component	Model / Details	Notes / Interface	Reference
Micro-Controller	Teensy 4.0	3V/5V -	2.2.1
Pyranometer	SN-300AL-GH-N01	7V to 30V/Modbus	2.2.2
Carbon monoxide (CO) Sensor	MQ-7	5V/Analog	2.2.3
Ozone (O3) Sensor	MQ-131	5V/Analog	2.2.4
Particulate Matter Sensor	Sensirion SPS30	5V/UART	2.2.5
Temperature, Pressure, Humidity Sensor	BME280	3V I2C	2.2.6
Gyroscope - Accelerometer	MPU6050	3V I2C	2.2.7
WiFi Module	ESP8266 Wemos D1 mini	5V/UART	2.2.8
SD Card Reader	Purecrea SD	5V/SPI	2.2.9
Logic Level Converter I	Purecrea 5V-3V	3V/5V	2.2.10
Logic Level Converter II	LM2596 DC-DC	16V/5V	2.2.10
RS485 to TTL Converter	RS485 to TTL	3V UART	2.2.11
CANBUS Reader	MCP2551	5V/CAN BUS	2.2.12
Powering/Data Interface	NMEA 2000	CAN-Bus	2.2.13
Switch	TL36	-	2.2.14
Battery	4S-2P	16V	2.2.15

Table 3: Hardware Components **Specifications**. Note: The red components are included in the designs but have not been implemented on the physical prototype yet.

Three categories of modules are distinguished :

- **Environmental sensors:** Which include the **MQ-7** (CO), the **MQ-131** (O_3), the **SPS30** (Particulate matter) -capable of monitoring multiple PM sizes- and the **pyranometer** (sunlight irradiance). These sensors vary in reliability depending on sensor type and environmental conditions. O_3 sensors, typically electrochemical, may suffer from cross-sensitivity to NO_2 and requires calibration, yet it returns accurate results with regard to observable concentration variations[7]. PM sensors, often based on optical scattering, provide good correlation with reference instruments under controlled conditions. Yet, they may be affected by humidity, particle composition. Additional airflow control improves their accuracy [13]. CO sensors are generally stable, especially electrochemical types, and can be reliable for trend monitoring when regularly calibrated [14, 3]. The pyranometer was added as it was available. Yet, the model at hand is highly unreliable and other models should be considered to ultimately replace it. Although a CO_2 sensor was first considered, it was then removed from the list as CO_2 variations are smaller than the error margin of the low-cost sensor envisaged for this project. Other low-cost sensors for gases such as SO_2, NO_x, and VOCs where not included in this prototype as they suffer from limitations in accuracy, selectivity, and stability. Electrochemical sensors for SO_2 and NO_2 are prone to cross-sensitivities and require frequent calibration to remain reliable [14, 7]. Metal-oxide VOC sensors, while affordable, generally lack chemical specificity and are highly sensitive to humidity and temperature changes [15].

- **Calibrating sensors:** Which include the **BME280** (temperature, humidity and pressure) and the **MPU6050** (accelerometer and gyroscope). Originally meant to monitor environmental states, the BME280 was kept in the prototype to calibrate and correct the Environmental sensors measurements. Being inside the same closed volume as the gas sensors that produce significant amounts of heat when functioning would bias all environmental temperature and relative humidity measurements. The MPU6050 is also used as a correcting device together with the pyranometer. On a rocking boat, the station will constantly be moving inducing unwanted variations in the solar irradiance readings. The MPU6050 is then used to measure the deviation between the vertical axis and the orientation of the station, thus allowing to correct solar irradiance measurements.

- **Electronic modules:** Which include the **Teensy 4.0** (micro-controller), the **ESP8266 Wemos D1 Mini** (WiFi), the Purecrea SD (SD-card reader), the **Purecrea 5V-3V and LM2596 DC-DC** (logicl level converters), the **RS485 to TTL converter** (protocol converter), the **switch** and the **NMEA2000** (boat powering). The micro-controller is the brain of the prototype and manages all sources of information at once to direct them to the WiFi module and to the SD-card. The SD-card reader allows to store timeseries of the data for further analysis. The WiFi module is necessary to get an instantaneous view of the data being recorded while the station is measuring in a remote place as well as to create a remote server on which timeseries can be retrieved from the SC-card. Logic level converters are needed to ensure that the right voltages and communication protocols are used with each sensor. Since two sources of power are planned, i.e. the external battery and the NMEA connection, a switch is required to select which one to use when. The battery is required for remote use on day-trip excursions, while the NMEA200 connection can be used for long term sailing trips when the station is fixed on the boat. The NMEA200 can also provide wind and gps data to the teensy when available on the boat considered.

More details are provided in the Sections 2.2.1 to 2.2.15.

Figure 9 shows a schematic of connections between all elements of the electronic circuit designed on KiCad9.0 [16]. It illustrates which pins of the Teensy 4.0 must be connected to which pins of which module, depending on voltage and communication protocols.

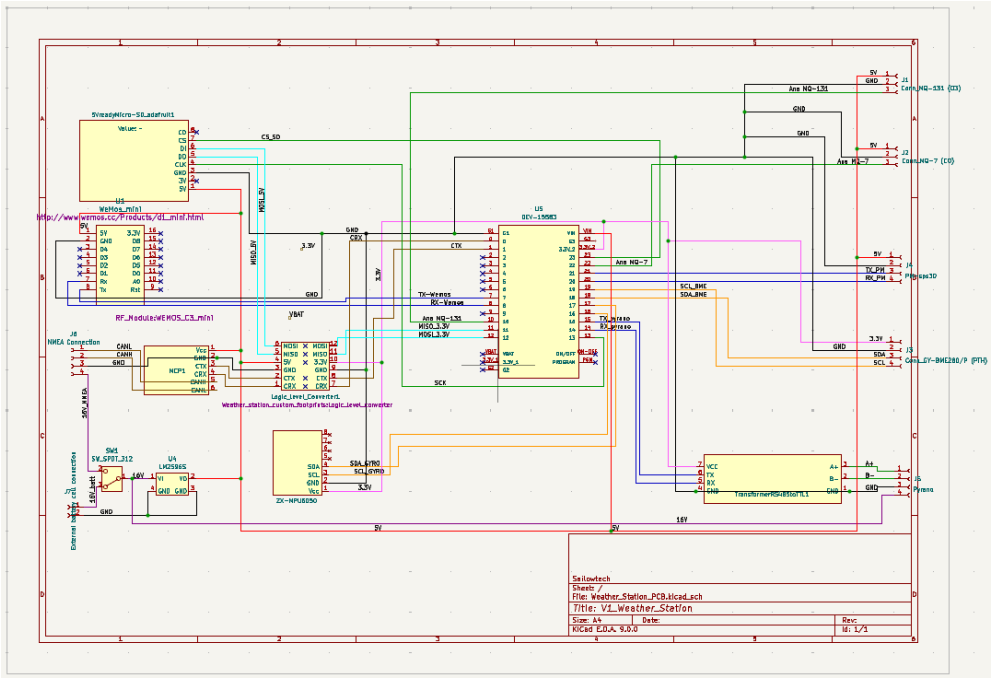


Figure 9: **Electronic circuit schematic.** Black : ground (GND) lines, pink : low voltage (3.3V) lines, red : middle voltage (5V) lines, purple : high voltage (16V) line, orange : I2C porotocol, dark blue : UART protocol, brown : CAN bus protocol, green analogical signal, select or clock. The green dots indicate connections between lines and the crosses indicate unavailability (the up-to-date schematic is on Sailowtech's github [17]).

The associated footprint is shown in Figure 10 which is derived from the schematic in Figure 9. It organizes the elements in space, and thus models the printed circuit board (PCB).

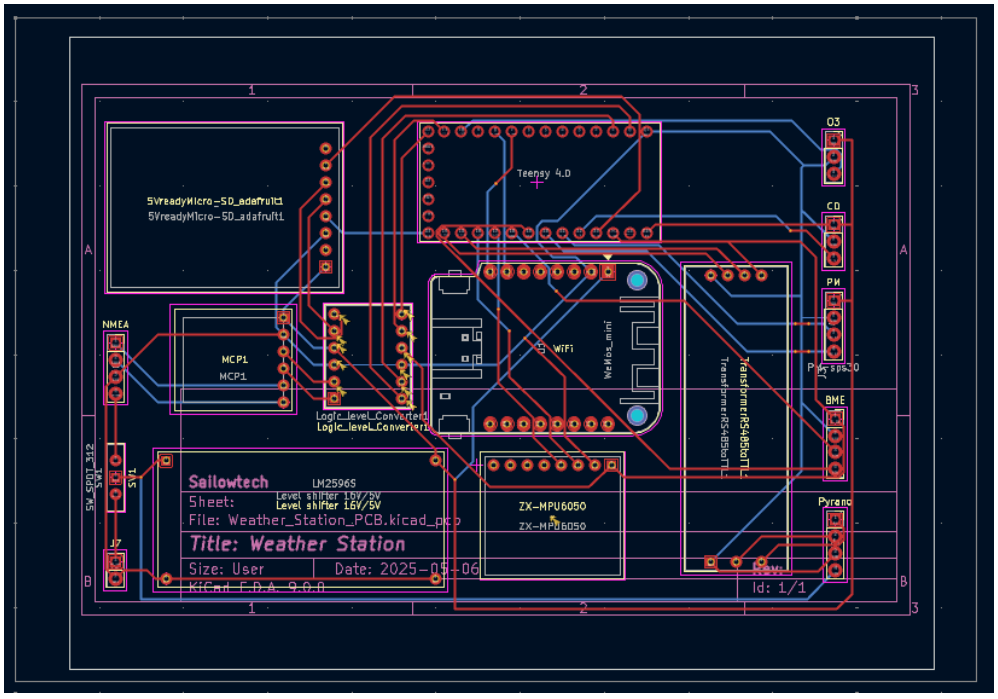


Figure 10: **PCB footprint with all modules**. The red and blue lines are copper tracks on two different levels. the red circles are through-holes.

2.2.1 Microcontroller

The Teensy 4.0 is a powerful, compact microcontroller board designed by PJRC[18] featuring an ARM Cortex-M7 processor running at 600 MHz, offering high-speed performance, extensive I/O capabilities, and compatibility with Arduino IDE for easy programming. It is used in this project as the main microcontroller and manages simultaneously all the electronic components listed in Table 3 of Section 2.2. The selection of a Teensy 4.0 microcontroller over a standard Arduino board (such as the Uno or Mega) is primarily justified by the Teensy's superior computational capabilities and expanded hardware resources. Powered by an ARM Cortex-M7 processor running at 600 MHz, the Teensy 4.0 significantly outperforms typical 8-bit Arduino boards, enabling real-time processing of multiple high-frequency sensor inputs without latency or data loss [19]. This is particularly critical for environmental monitoring systems where multiple gas sensors may be polled simultaneously, and timestamped data logging or wireless transmission must occur without bottlenecks. Additionally, the Teensy offers more I/O pins, multiple hardware serial ports, and native USB support with high-speed data transfer—features that enhance compatibility with digital sensors and external storage or communication modules [20]. Its compact size and low power consumption further make it an ideal choice for mobile or battery-operated deployments, such as those on scientific expeditions aboard sailboats.

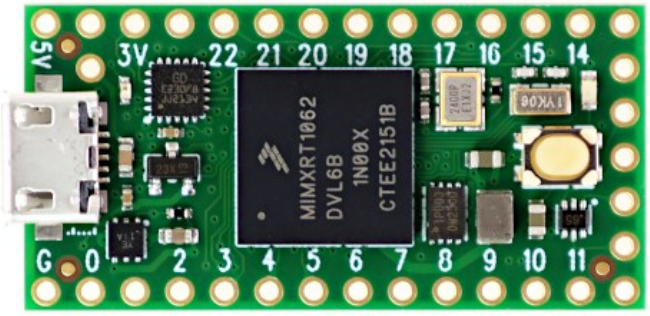


Figure 11: Teensy 4.0 board

The Teensy pinout scheme as well as all specifications can be found on the PJRC website[18].

2.2.2 Pyranometer

A pyranometer is a sensor used to measure solar irradiance. At the pyranometer's core is a black-coated thermopile sensor, made up of thermocouples that generate a voltage in response to a temperature difference caused by absorbed sunlight[21]—an effect known as the Seebeck effect[22]. The black coating ensures uniform absorption across all wavelengths of solar radiation. Pyranometers primarily measure Global Horizontal Irradiance (GHI), which includes both direct sunlight and diffuse sky radiation. The device outputs a low-voltage signal proportional to the solar irradiance, typically converted into watts per square meter (W/m^2) through calibration. It has a spectral range of 400 to 700 nm, a measurement range from 0 to 870 W/m^2, and can operate in a temperature range going from -30℃ to 75℃[23].

Figure 12: Pyranometer sensor : RS485

Figure 12 shows the model used in the gas monitoring station prototype : the RS485 [24] which had already been ordered at the start of the project.

2.2.3 Carbon monoxide sensor

Carbon monoxide is measured with a MQ-7 sensor shown in Figure 13.

The MQ-7 is a gas sensor used to detect carbon monoxide (CO) concentrations in the air. It operates based on a metal-oxide semiconductor (MOS) principle, using a heated tin dioxide (SnO_2) sensing layer[25]. In clean

air, the sensor's resistance is high, but when CO is present, it reacts with adsorbed oxygen ions on the sensor surface, reducing them and thereby lowering the sensor's resistance. This change in resistance is proportional to the CO concentration. The MQ-7 uses a cyclical heating mechanism, alternating between a high temperature (for sensing) and a low temperature (for recovery and baseline stability), requiring careful timing in signal processing. It outputs an analog voltage that can be read by a microcontroller. It has a measurement range of 20 – 2000 ppm, and can operate in a temperature range going from -10°C to +50°C [26].

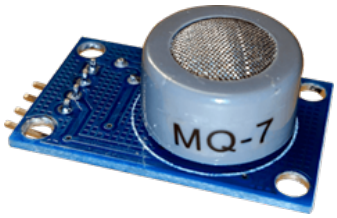


Figure 13: MQ-7 sensor

2.2.4 Ozone sensor

Ozone is monitored with an MQ-131 sensor displayed in Figure 14

The MQ-131 ozone sensor operates on a reverse chemiresistive principle using tungsten trioxide (WO_3) as its sensing material : it shows increasing resistance R_s with rising gas concentration, as ozone is a strong oxidizing gas. Unlike reducing gases (e.g., CO or CH) that donate electrons and decrease the sensor's resistance, ozone extracts electrons from the sensor's tungsten trioxide (WO_3) surface, reducing the number of free charge carriers. This deepens the electron depletion layer in the semiconductor, thereby increasing its resistance. This behavior is especially prominent in the low-concentration version of the MQ-131, which is tuned for sensitivity to small amounts of ozone. The sensor includes a built-in heater to maintain optimal temperature (≈300°C) for this reaction, and its response is calibrated using the Rs/R_0 ratio (resistance in gas vs. clean air). It requires 24 to 48 hours of preheating to operate effectively and provides an analog voltage output.[27, 28]. Precise calibration instructions are given in section 3.2. It has a measurement range of 10ppb - 2ppm, and can operate in a temperature range going from : -20°C to +50°C [27].

Figure 14: MQ-131 sensor

2.2.5 Particulate matter sensor

The particulate matter sensor used is called Sensirion sps30 and it is displayed in Figure 15.

The Sensirion SPS30 is an optical particulate matter (PM) sensor that operates on the principle of laser scattering. In this method, a laser beam illuminates airborne particles as they pass through the sensor's detection chamber. Ambient air is drawn passively into the sensor through small openings, the particles scatter the laser light, and the intensity and angle of the scattered light are measured by a photodetector. These measurements allow the sensor to determine the concentration and size distribution of particles in the air [29, 30]. Independent evaluations have demonstrated that the SPS30 provides reliable measurements of PM concentrations across various size fractions, making it suitable for air quality monitoring applications [31]. Its mass concentration range goes from 0 to 1000 $\mu g/m^3$, its particle size detection range goes from 0.3 – 10 μm, and its operating temperature range goes from -10°C to +60°C [32].

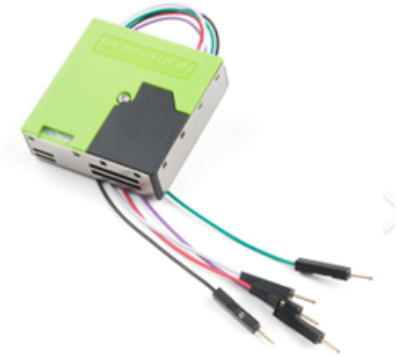


Figure 15: Sensirion sps30 sensor

2.2.6 Pressure, humidity, and temperature sensor

Pressure humidity and temperature are monitored with a GY-BME280 sensor shown in Figure 16.

The GY-BME280 is a sensor module based on the Bosch BME280, measuring temperature, humidity, and atmospheric pressure[33]. Temperature is measured using a bandgap sensor that detects voltage changes across semiconductor junctions affected by temperature. Humidity is sensed by a capacitive sensor where a polymer dielectric layer absorbs moisture, changing the capacitance proportional to relative humidity. Pressure is measured with a MEMS piezo-resistive sensor, where atmospheric pressure deforms a diaphragm, altering resistance in piezo-resistive elements[34]. The sensor system offers a wide range of environmental measurements. It can measure temperatures from -40°C to +85°C, with a typical accuracy of ±1.0°C. Humidity measurements span from 0% to 100% relative humidity, with a typical accuracy of ±3% RH. For atmospheric pressure, the sensor operates within a range of 300 to 1100 hPa, maintaining an accuracy of approximately ±1 hPa [35].

Figure 16: BME sensor

2.2.7 Gyroscope - accelerometer

The Gyroscope and Accelerometer are contained in a single device called MPU6050, illustrated in Figure 17.

The MPU6050 is a motion-tracking sensor that combines a 3-axis accelerometer and a 3-axis gyroscope on a single chip. "The accelerometer works on the principle of piezo-electric effect, the ability of certain materials to generate an electric charge in response to applied mechanical stress"[36]. It measures acceleration along three axes using capacitive MEMS technology: when the device moves, tiny proof masses shift, causing changes in capacitance that are proportional to the acceleration. The gyroscope measures angular velocity based on the Coriolis effect—when the sensor rotates, vibrating MEMS structures experience a Coriolis acceleration[36] that induces measurable displacement, which is converted into an electrical signal through the piezo-electric effect as well. The accelerometer supports measurement ranges of ±2g, ±4g, ±8g, and ±16g. The gyroscope can measure angular velocities within the ranges of ±250, ±500, ±1000, and ±2000 degrees per second (°/s). Finally it is designed to function within an operating temperature range of -40°C to +85°C[37].

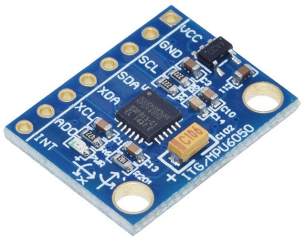


Figure 17: MPU6050 sensor

2.2.8 WiFi module

The WiFi module used is called Wemos D1 mini (Wemos) (c.f. Figure 18). It is a compact development board based on the ESP8266 microcontroller. Physically, it features a USB port for power and programming, a voltage regulator to supply 3.3V to the ESP8266 chip, and a set of GPIO pins for connecting sensors and actuators. These GPIO pins are not used in this project. Instead, it is operating in Soft Access Point (SoftAP) mode, where it creates its own Wi-Fi hotspot. This allows a computer to connect directly to the Wemos D1 mini without needing an external router. The code provided in Appendix A.4 was flashed to the Wemos through USB to set a custom network name (SSID) and password, turning the board into a standalone wireless hub.

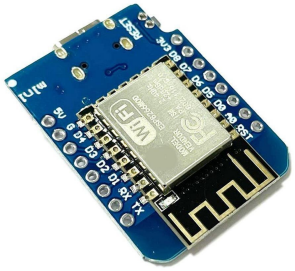


Figure 18: Wemos D1 mini

2.2.9 SD-card module

The SD-card reader module allows to physically connect an micro-sd card to the station, to let it store the data for a prolonged period of time. It communicates with the Teensy using the SPI (Serial Peripheral Interface) protocol, which enables high-speed, synchronous data transfer. The SPI bus includes four key lines: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), and CS (Chip Select). In this setup, the microcontroller acts as the master, sending commands and data to the SD card via MOSI, while receiving data back from the card on the MISO line. The clock signal, provided on the SCK line, synchronizes communication, and the CS line activates the SD card for communication. This protocol allows efficient read/write operations to the SD card, making it well-suited for data logging applications in embedded systems.

Figure 19: SD-card module

2.2.10 Logic level converters

Logic level converters are used to safely interface electronic components that operate at different voltage levels. For example, a microcontroller running at 3.3 V may need to communicate with a sensor or module that uses 5 V

logic. Directly connecting these devices can damage components or result in unreliable communication. Logic level converters shift voltage levels up or down to ensure compatibility between devices, preserving signal integrity and protecting hardware.

(a) LM2596 DC-DC 16V to 5V cnverter (b) 3.3 V to 5 V logic level converter

Figure 20: Voltage and logic level converters used in the monitoring system.

Figure 20 displays that used in the circuit: the 16V to 5V converter allows the xteranl battery to power the Teensy which requires 5V direct current. The 3.3V to 5V converter switches voltage for SPI instructions coming from the the NMEA2000.

2.2.11 RS485 to TTL converter

This module allows to convert the UART encoded information receivable and sendable by the Teensy into RS485-adapted instructions.

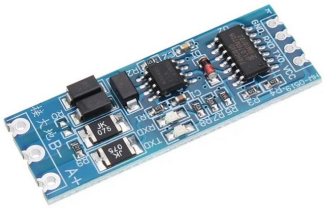


Figure 21: RS485 to TTL logic level shifter to connect to the pyranometer in RS485

2.2.12 CANBUS Reader

The MCP2551 is a device that allows to connet the NMEA2000 system to the teensy by converting CANBUS protocole to SPI.

Figure 22: CANBUS reader for NMEA2000 connexion

2.2.13 NMEA2000

NMEA 2000 is a marine communication protocol based on the CAN (Controller Area Network) bus standard, designed to allow robust, plug-and-play networking of marine instruments and sensors. It enables multiple devices—such as GPS receivers, wind sensors, and autopilots—to exchange data over a shared bus using standardized message formats called PGNs (Parameter Group Numbers)[38]. To interface a Teensy microcontroller with an NMEA 2000 network, a CAN transceiver (e.g., MCP2562) and appropriate software libraries (such as the open-source `NMEA2000` library for Arduino/Teensy[39]) are used to decode messages directly on the CAN bus. This setup allows the Teensy to parse and utilize marine data such as wind speed, wind direction, and GPS position, provided by other devices on the NMEA 2000 network [40, 41].

2.2.14 Switch

The switch effectively chooses the source of power between the NMEA2000 which gets its power from the boat's battery and the stations embedded battery.

2.2.15 Battery

A 4S-2P rechargeable battery was constructed out of refurbished cells by the students who initiated the project. According to their assessment : "These cells are characterized by a voltage and a current of 3.7-4.2V and 1500mAh, respectively. Connecting batteries in series increases the voltage, while connecting batteries in parallel increases the current. Due to the high voltage requirements from the pyranometer, optimally working at 12V, the battery needed to have at least 4 cells in series, resulting in a total voltage of around 16V [42]".

The total energy stored in the battery pack is given by:

$$E = V \times C = 16\,\mathrm{V} \times 1.5\,\mathrm{A\,h} = 24\,\mathrm{W\,h}$$

Based on the tests carried out, the gas monitoring station continuously requires 2.9 W. Which grants the following total runtime if ran continuously :

$$\text{Runtime} = \frac{E}{P} = \frac{24\,\mathrm{W\,h}}{2.9\,\mathrm{W}} \approx 8.28\,\mathrm{h}$$

The gas monitoring station will continuously run for approximately **8.3 hours** using 4 batteries of 4.2 V and 1500 mA h connected in series.

In our first conception, we imagined turning the station on and off manually, 15 minutes per day, which would lead to a full month of daily measurements without charge. Yet, continuous measurement is more appropriate for these sensors due to their important noise and drift upon starting and stopping [8].

2.3 Overall cost

based on a macro (over-)estimation, this gas monitoring station prototype has cost around CHF 400.-, including parts that needed to be re-bought.

3 Calibration & correction

All chosen modules must be calibrated to be properly used. Up to now, the MPU6050 (gyroscope and accelerometer) the MQ7 (CO) and the MQ131 (O_3) have been calibrated. The pyranometer suddenly stopped functioning for a reason yet to be determined and the SPS30 (PM) has never started working, making their calibration impossible. The BME280 (Pressure, Humidity, Temperature) is factory calibrated [43] and deemed precise enough for this first prototype.

3.1 Gyroscope - Accelerometer

The **MPU6050's gyroscope** is calibrated by removing the bias for all three axes. At start-up, sample of 500 measurements is averaged for each axis, constituting the average bias. These biases are then removed from all subsequent measurements. Calibration is automatically performed every time the station is turned on.

The **MPU6050's accelerometer** must be manually calibrated as it can suffer from:

- Offset error (bias): Readings are non-zero when the device is stationary.
- Scale factor error: Readings do not match real-world values exactly (e.g., gravity $\neq$ 9.81 m/s^2).
- Axis misalignment or cross-axis sensitivity.

There are two types of calibration: static calibration (bias and scale correction) and dynamic calibration (involving an external reference). However, the latter is more complex and usually not necessary for basic applications.Static calibration is preferred in this case and is performed as follows : The MPU6050 is placed flat on a surface in six known orientations (one face up at a time). In each position, only one axis should measure $\pm$1g (gravity), and the other two $\approx$0g. The raw values from each axis are recorded. An offset and scale factor can be computed from the $\pm$1g readings along all axes:

- Offset (bias): The average of the two values ($\pm$1g).
- Scale factor: The half-difference from the expected $\pm$1g values.

All subsequent measured accelerations are then corrected according to the following formula :

$$Ac_{cor} = \frac{Ac_{raw} - Offset}{Scalefactor} \tag{1}$$

3.2 MQ Sensors

MQ sensors produce analog voltage outputs that vary with the concentration of the target gas (see Sections 2.2.3 and 2.2.4). Calibration involves determining the sensor's baseline resistance (R_0), which represents its resistance in clean air. Ideally, this process requires exposing the sensor to gas samples with known concentrations to measure the corresponding sensor resistance (R_S) at each level. A calibration curve can then be established by fitting an appropriate model to the data. In the absence of reference gas samples, standard characteristic curves provided in the manufacturer's datasheet can be used. For both the MQ-7 and MQ-131 sensors, the relationship between the resistance ratio $\frac{R_S}{R_0}$ and gas concentration (in ppm) has been shown to follow a power-law behavior (c.f. Appendix A.2 and A.1), expressed in the following equation:

$$\log_{10}\left(\frac{R_s}{R_0}\right) = a \cdot \log_{10}(\text{ppm}) + b \Leftrightarrow ppm = 10^{\frac{1}{a}\cdot\log_{10}\left(\frac{R_s}{R_0}\right) - \frac{b}{a}} \tag{2}$$

Where:

- R_s is the sensor resistance during measurement,
- R_0 is the sensor resistance in clean air,
- a and b are parameters to be determined.

To get R_0, 48 hours of continuous powering are required to allow the sensing material to stabilize. Once the sensor is stabilized in a clean air environment, its baseline resistance R_0 is measured by first converting the raw ADC reading into a voltage:

$$V_{\text{out}} = \frac{\text{ADC\_value}}{1023} \times V_{\text{cc}}, \tag{3}$$

Where:

- V_{cc} is the Analog pin output voltage equal to 5V,
- ADC_{value} is the Analog-to-Digital Converter value ranging from 0 to 1023,
- V_{out} is the corresponding voltage.

Then the sensor resistance is computed as follows:

$$R_s = R_L \cdot \left(\frac{V_{\text{cc}} - V_{\text{out}}}{V_{\text{out}}} \right). \tag{4}$$

Where R_L is the load resistor of the module on which the MQ sensor sits, assumed to equal 10000 Ω.

For each MQ sensor, the calculation is done ten times and the average value is assigned as R_0. During normal operation, R_s is calculated in real-time and the ozone concentration is estimated using the empirical logarithmic formula displayed in Equation 2. The calibrated R_0 is stored in the code. Calibration should be repeated periodically, as R_0 may drift over time due to aging or environmental changes.

3.2.1 Ozone sensor (MQ-131)

R_0 was measured to be $R_0 \approx 5172.13\ \Omega$ for O_3, taking the values of $a \approx 0.45$ and $b \approx 0.90$ respectively, according to the MQ131 datasheet curve [27] which can be found in Appendix A.1.

3.2.2 Carbon monoxide sensor (MQ-7)

For CO, R_0 was measured to be $R_0 \approx 9500\ \Omega$, taking the values of $a \approx -0.7$ and $b \approx 1.4$. This was deduced from the MQ7 datasheet [26] on the power law curve that can be found in Appendix A.2.

3.3 Instrument-based corrections

Measurement correction can be achieved by cross-validating sensor outputs with data from complementary sensors, further improving measurement accuracy and precision.

3.3.1 Solar irradiance correction

To estimate the tilt of the monitoring station with respect to the vertical, an MPU6050 inertial measurement unit (IMU) was used to combine data from a 3-axis accelerometer and a 3-axis gyroscope. The sensor manufacturer chose a non-conventional way to set its axes [37]. When mounted their orientation with respect to the motherboard is the following: the x-axis points toward the Teensy microcontroller, the y-axis toward the battery pack, and the z-axis into the base of the unit. Knowing that the motherboard will be vertically set, it results that when the boat is not rocking, the x-axis will be parallel to the water - likely perpendicular to the boat's sides, the y-axis will be perpendicular to the water, and the z-axis will also be parallel to the water but also parallel to the boat's sides.

Sensor Fusion Approach

A complementary filter was implemented to fuse accelerometer and gyroscope data for estimating pitch and roll. This technique balances the high-frequency responsiveness of gyroscope data with the low-frequency stability of accelerometer readings, thus minimizing drift and reducing noise [44].

The fusion equations are:

$$\hat{\phi}_{n+1} = \alpha \cdot \phi_{\text{acc},n} + (1 - \alpha) \cdot \left(\hat{\phi}_n + \Delta t \cdot \dot{\phi}_{\text{gyr},n} \right) \tag{5}$$

$$\hat{\theta}_{n+1} = \alpha \cdot \theta_{\text{acc},n} + (1 - \alpha) \cdot \left(\hat{\theta}_n + \Delta t \cdot \dot{\theta}_{\text{gyr},n} \right) \tag{6}$$

where:

- $\hat{\phi}$ is the estimated pitch angle (rotation around the y-axis),
- $\hat{\theta}$ is the estimated roll angle (rotation around the x-axis),
- α is the complementary filter coefficient (set to 0.98, thus heavily favoring the accelerometer data, as the latter was deemed rather noisy),
- Δt is the sampling time,
- $\dot{\phi}_{\text{gyr}}$ and $\dot{\theta}_{\text{gyr}}$ are angular velocity readings from the gyroscope (in °/s),
- ϕ_{acc} and θ_{acc} are instantaneous angle estimates from the accelerometer.

Accelerometer Angle Calculation

Given the sensor's rotated frame, the accelerometer-based pitch and roll are computed as:

$$\phi_{\text{acc}} = \tan^{-1} \left(\frac{-a_z}{\sqrt{a_y^2 + a_z^2}} \right) \tag{7}$$

$$\theta_{\text{acc}} = \tan^{-1} \left(\frac{a_x}{a_y} \right) \tag{8}$$

These equations were chosen based on the physical orientation of the axes, where a_x, a_y, and a_z represent acceleration readings in g.

Tilt Angle and Application
To estimate the sensor's tilt from horizontal, the Euclidean norm of the pitch and roll estimates is used:

$$\theta_{\text{tilt}} = \sqrt{\hat{\phi}^2 + \hat{\theta}^2} \tag{9}$$

This is a valid approximation for small angles and when solar angle data is not available. The resulting tilt angle is used to correct solar irradiance measured by a horizontal pyranometer:

To estimate the global horizontal irradiance (GHI) from a pyranometer mounted on a tilted or rocking platform, the measured irradiance must be divided by the cosine of the tilt angle. Assuming the sun is approximately overhead and the tilt is the dominant source of deviation from horizontal, the correction is applied as:

$$I_{\text{horizontal}} = \frac{I_{\text{measured}}}{\cos\left(\frac{\theta_{\text{tilt}} \cdot \pi}{180}\right)} \quad [\text{W/m}^2] \tag{10}$$

This compensates for the reduced or increased effective area of the pyranometer exposed to incoming solar radiation due to tilt.

3.3.2 Gas measurement correction
The MQ-7 and MQ-131 datasheets provide insight into the sensors' reaction with respect to humidity and temperature [26, 27]. These relationships can be found in Appendix A.3. To increase measurement reliability, further tests and corrections are required to include these environmental effects in the measurements.

4 Use
4.1 Software
The software was developed using the Arduino framework for the Teensy 4.0 microcontroller. It handles sensor initialization, calibration, data acquisition, processing, and logging. This following section describe the main software components and concepts implemented in the final version.

Serial communication and baud rates The system uses multiple serial interfaces:

- `Serial2` used for communication with the ESP8266 module at a Baud rate of 115200.
- `Serial3` is configured at 9600 baud to communicate with the pyranometer using the MODBUS protocol.
- `Serial` is used for communication on the terminal at a Baud rate of 74880.

Correct baud rate configuration ensures reliable and synchronized data exchange between devices. The baud rate defines the number of bits transmitted per second during serial communication; both communicating devices must use the same rate to exchange data reliably.

Sensor setup and calibration For the MPU6050 gyroscope, a calibration function is called during initialization. It takes several consecutive readings (500 samples) while the device is stationary, then calculates an average for each axis. These biases are then subtracted from all subsequent measurements, to correct deviations due to the gyroscope's natural drift. For the gas sensors (MQ-7 and MQ-131), calibration is based on a reference value (`R0`), corresponding to the sensor resistance in a reference environment (clean air). During runtime, the software reads the sensor output voltage, calculates the instantaneous resistance (`Rs`) and then evaluates the gas concentration using a ratio `Rs/R0`. These formulas are directly encoded in the `readMQ7()` and `readMQ131()` functions, and are adapted according to the logarithmic curves taken from the sensor data sheets.

Data logging and timing All processed data is logged to an SD card in CSV format, which includes : timestamp, temperature, humidity, pressure, pitch, roll, inclination angle, light (raw and corrected), CO and O_3 concentrations. If the file is newly created, a header row is written.

The system captures a full set of measurements every 5 seconds. This interval is currently implemented using a simple delay mechanism and is sufficient for typical environmental monitoring applications. The sampling rate can be adjusted in future versions. In addition, a timing mechanism is implemented to calculate the time elapsed between two loop iterations. This is done using the `millis()` function to compute a time difference (`deltaTime`) in seconds. This delta time is essential for accurate integration of gyroscope data when estimating the orientation of the system, particularly for computing pitch, roll, and inclination angle.

4.2 Web interface and system architecture

As described in previous sections, the gas monitoring station collects environmental data from multiple sensors and stores it in a CSV file format. This data can be accessed through two primary methods:

- Direct access via the SD card.
- Wireless download through a web interface, accessible via Wi-Fi.

To achieve this, several hardware components are integrated and communicate with each other as shown in Figure 23.

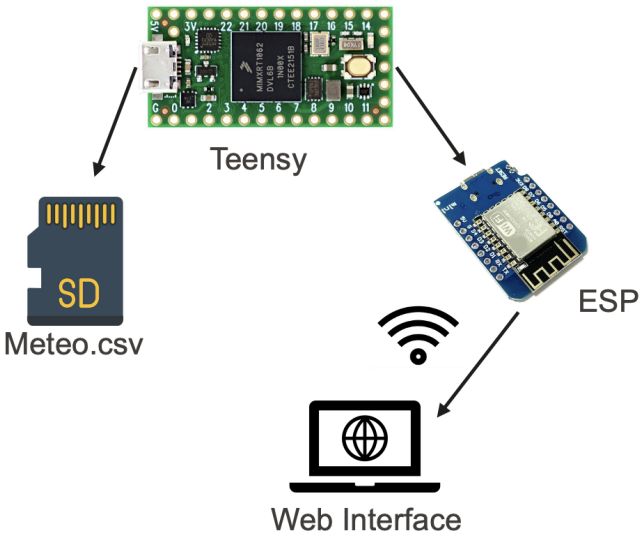


Figure 23: Simplified diagram of the system architecture.

System components and interactions

- Teensy (micro-controller) : acts as the core processor. It interfaces with various sensors using UART and I^2C protocols, collects data, and logs it to a CSV file on the SD card.
- SD Card module: stores all sensor measurements in CSV format, which can be retrieved either directly or wirelessly.
- ESP8255 (Wi-Fi module): connected to the Teensy via UART, it handles wireless communication. Upon receiving the read measurements command, it requests the CSV file from the Teensy and transmits the contents to the user's device.
- Web Interface (User's device): connects to the ESP's Wi-Fi hotspot (SSID: TeensyBridge), and opens a basic web page hosted by the ESP (e.g., http://192.168.4.1). This page allows users to download the most recent CSV file with a single click.

Communication with Teensy: A dedicated script, named 'final-code.ino' (see in Appendix A.5), is deployed to the Teensy via USB. This code performs the following key functions:

1. Initializes sensors: Gyroscope (MPU6050), Pyranometer, BME280, MQ-7, MQ-131
2. Performs calibration for the gyroscope, MQ-7, and MQ-131
3. Responds to CSV file requests from the ESP module
4. Runs a continuous loop to:
   (a) Open the CSV file on the SD card for writing
   (b) Read light intensity from the pyranometer
   (c) Read gyroscopic data to calculate pitch and roll, then compute tilt angle (θ) to correct irradiance measurements
   (d) Read environmental data from the BME280 (temperature, humidity, pressure)
   (e) Read gas concentrations from MQ-7 (CO) and MQ-131 (O_3)
   (f) Log all sensor values to the CSV file
   (g) Close the CSV file to preserve data integrity

This firmware must be flashed onto the Teensy once via USB (must select the Board Teensy on Arduino), after which the system runs autonomously.

Creation of the Wi-Fi and Web interface: To enable wireless accees to the data stored on the SD card, the ESP8266 module is used to both create a Wi-Fi hostpost and host a simple web interface. This is achieved by using the 'ESP-web-server.cpp' code (see Appendix A.4) and the ESP8266Wifi and ESP8266WebServer libraries.

This code must be flashed onto the ESP via USB once (select the 'NodeMCU 1.0' or 'Generic ESP8266' board on Arduino or PlatformIO).

- Wi-Fi Hotspot creation: In the setup() function, the ESP8266 starts in Access Point (AP). It creates a Wi-Fi network named 'TeensyBridge' protected with the password '12345678".
- Hosting a Web Server: Once the AP is active, the ESP8266 launches a web server on port 80, the standard port used for HTTP traffic. It means users can simply open a browser and go to http://192.168.4.1 without neededing to specify a port. Two HTTP routes are handled:
  1. /(root) : displays a welcome page with a button
  2. /download : triggers the CSV download

  The root handler (handleRoot) sends a user-friendly web page with a 'Download Data' button that links to /download.
- Serving the CSV file : when the user clicks the button, the browser request the /download route, which triggers the function handleDownload() . This function does :
  1. Sends a serial command to the Teensy to request the CSV data
  2. Wait 2 sec to receive the response from the Teensy via UART
  3. As data arrives, it is accumulated into a String
  4. Once complete, the server responds with proper HTTP headers to initiate a file download.
- Optional TCP communication : in addition to the web interface, the ESP8266 also runs a TCP server on port 23 which is traditionally used for Telner-style communication. This TCP connection provide a real-time between the ESP and the Teensy. When a terminal client connects to port 23: any data sent by Teensy over UART can be visualised directly and inversely for commands from the terminal to the Teensy.

5 Discussion

5.1 Challenges

Concept

The first challenge was conceptual. Initially, it was unclear whether the station should be a multipurpose system—monitoring both environmental variables and pollutant gases—or a more specialized unit. As development progressed, it became increasingly evident that the presence of gas sensors would irreversibly bias the environmental measurements, with no straightforward method of correction. Consequently, it was decided to retain the temperature, humidity, and pressure sensor (BME280) solely for the purpose of actively compensating the gas sensor data.

Mechanics

The first major task involved learning and using Fusion 360 software to model the entire station, a process that took approximately eight weeks, as we had never used it before.

During development, several mechanical parts had to be redesigned and remade multiple times due to unforeseen material constraints that only became apparent after the theoretical design phase had been completed. For instance, there was a significant issue with the battery not fitting into the outer cylinder, which required a redesign of the central structure to accommodate it properly.

Electronics

The electronics development phase encountered several notable challenges. Initially, significant time —approximately eight weeks— was invested in learning and utilizing the KiCad9.0 software for printed circuit board (PCB) modeling as we had no background in electronics. During prototyping, numerous PCB design issues emerged, including mirrored sensor placements, the unintended removal of circuit components, and unreliable electrical connections. Due to the mirroring of a few footprints, manual corrections were required once the PCB was printed (c.f. Figure 24) which required precise manipulation of welding material, including soldering, welding and blazing techniques.

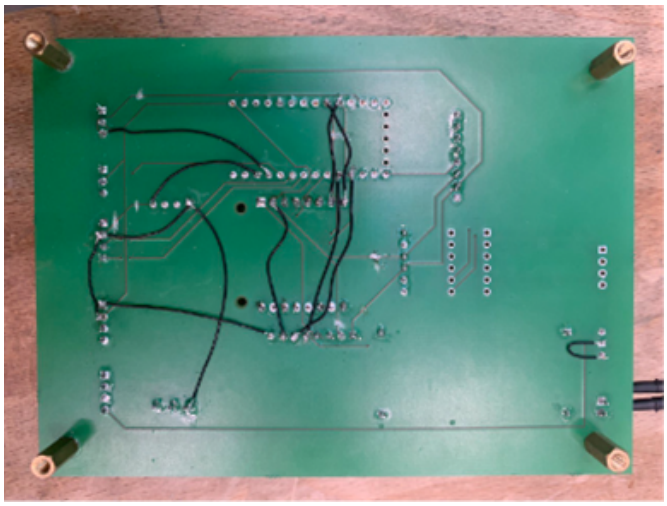


Figure 24: PCB correction

Further complications arose with sensor functionality: the pyrometer experienced unexplained malfunctions likely stemming from internal defects, despite extensive troubleshooting. The issue with the pyranometer that we inherited from our predecessors is that the only available documentation on it is in Chinese.

Additionally, the fine particle sensor (Sensirion sps30) failed to operate correctly, primarily due to an inappropriate communication protocol. It is supposed to work with both UART or I^2C protocols. In order to combine all sensors using solely one Teensy 4.0, the UART protocol was selected first as it allowed using different pins for all sensors. Despite repeated efforts to make it work, the sps30 never provided any response.

As explained in the instrument-based corrections section (c.f. Section 3.3), the axis of the MPU6050 are defined in an non-conventional way [37]. This required adaptation of the corrective formulas.

Software

The software creation and design also brought its fair share of challenges and issues. The sensor's baud rates had to be adapted to make sure that the data could be read and interpreted correctly. Delays between measurements had to be increased when merging the codes so that data could be gathered before trying to retrieve a new piece of information. Finally, depending on the module used, more or less online resources and documentation were available to help code the appropriate functions to enable the devices.

5.2 Improvements & Next Steps

Key areas were identified where improvements are necessary :

Mechanical

- Improve imperviousness to water by adding O-rings and isolating all cable holes
- Protect the inflow holes with mechanical structures
- Screw the pyranometer on top of the station and find a way to route its wire inside
- Further protect the aeration holes using elbow pipes to reduce water ingress while allowing airflow

Electrical

- Redesign RS485 to TTL footprint
- Flip ESP8266 and CAN BUS footprints
- Change wiring to use I^2C for SPS30
- Use wider pin connectors for battery and pyranometer
- Replace existing connectors with more practical sensor interfaces
- Add a switch between NMEA and external battery; remove existing soldered wire below

Conceptual

- Apply correction of gas sensor readings using temperature and humidity data
- Evaluate power-saving strategies that minimize impact on sensor accuracy
- Convert O_3 concentrations directly to ppb in software

User Access

- Conduct field testing in varied environmental conditions
- Develop a user-friendly interface for configuration and data access
- Avoid loading the entire CSV file into ESP memory before sending it – use e line-by-line or streaming approach to supper larger files

5.3 Low-tech considerations

In developing this environmental monitoring station, particular attention was given to aligning the project with low-tech values. This approach prioritizes simplicity, sustainability, autonomy, and meaningful impact over technical sophistication for its own sake. Figure 25 highlights the topics tackled by the low-tech approach to make sure the weather station is useful, accessible, and sustainable.

- **Located** : the use of the station will answer located problems but the manufacturing process still depends on global production chain (mainly the electronic parts).
- **Objective:** by focusing on what is essenital (measuring gas concentrations, solar irradiance in a continuous manner) for our station, no high-resolution data sensors or external infrastructure (screen) were implemented. This design allo to saves energy, reduces complexity and focuses on sufficiency i.e collecting enough data for meaningful analysis.
- **Psychologically transforming:** the station is designed to reveal invisible phenomena, such as ozone and carbon monoxide levels, by making them visible in real time. Whether data are collected in seemingly pristine, remote areas or more polluted environments such as ports, it can provoke reflection and even discomfort. This contrast challenges users' perceptions of air quality and environmental impact, fostering a deeper, more personal awareness of the atmosphere they inhabit.
- **Empowering:** this station empowers individuals to take direct, local, and autonomous measurements, giving them greater control over their environmental awareness and actions.
- **De-automated**: the station intentionally avoids unnecessary automation, such as mechanical ventilation or complex control systems. Instead, it relies on simple, transparent mechanisms that users can understand, maintain, and adapt. This approach enhances reliability and gives users greater control over how the station operates, reinforcing autonomy and trust in the system.
- **Radically useful :** it is radically useful in its intended purpose: providing accessible, durable environmental measurements as a compact onboard weather station
- **Technically sustainable**: the system is built with carefully selected sensors for precision and reliability, emphasizes repairability, and runs on simple, well-documented code based entirely on open-source libraries.

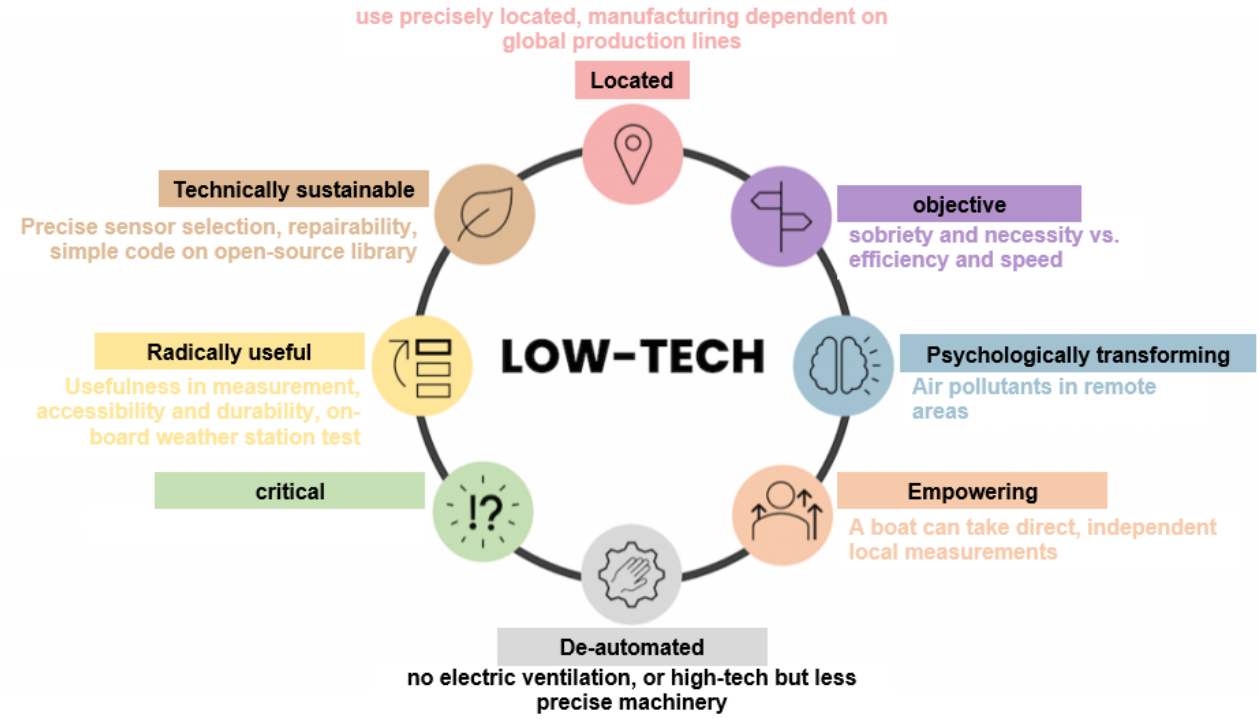


Figure 25: Low-tech principles to follow

6 Conclusion & outlook

The development of the Sailowtech gas monitoring station strives for a successful balance between affordability, functionality, and sustainability in the context of marine environmental monitoring. Designed to operate under the constraints of sailboat expeditions, the station aims to offer a compact, low-cost, and repairable solution for tracking key atmospheric pollutants—ozone (O_3), carbon monoxide (CO), and particulate matter (PM)—along with solar irradiance.

The tradeoff achieved in this project results in a device that costs less than CHF 500, provides remotely accessible data via WiFi, and remains adaptable and repairable even by users with minimal technical background. This combination makes the station particularly suitable for low-infrastructure environments and educational or grassroots scientific initiatives.

Despite a limited autonomous battery life of approximately 8 hours, the inclusion of an external power interface via NMEA2000 allows for continuous operation when mounted on sailboats, while still enabling standalone measurements during shorter, battery-powered excursions. This dual-power approach ensures flexibility and reliability across different deployment scenarios.

Throughout the prototyping process, significant challenges in mechanical design, electronics integration, and sensor calibration were addressed, leading to a functional and open-source system that prioritizes accessibility and environmental awareness. While further refinement is needed to improve sensor accuracy and system robustness, the station already offers a valuable prototype for marine atmospheric research—and a proof of concept for sustainable, low-tech innovation in environmental monitoring.

6.1 Acknowledgment

We want to thank Sailowtech for the unique opportunity to work on a highly stimulating project that allowed us to combine our will to provide useful, accessible, and sustainable solutions to the academic world and our ingenious and imaginative minds. We also want to thank the Skil and Spot's lab managers, Willow, Stephane, Marc, Sylvain, and Yvan, who provided their insights into prototyping and design. We would like to thank Professor Athanasios Nenes for his insights on air pollution and what sensors may be deemed of interest. Finally, we acknowledge that artificial intelligence (AI) played a part in helping to write this report, in particular chatGPT.

Contribution to the report:

- Matheo Godenzi : state of the art, electronics, calibration, discussion
- Lorraine Naux : original prototype, casing, software and web interface, discussion

References

[1] Frank J Kelly and Julia C Fussell. "Air pollution and public health: emerging hazards and improved understanding of risk". In: *Environmental Geochemistry and Health* 37.4 (2015), pp. 631–649. DOI: 10.1007/s10653-015-9720-1.

[2] World Health Organization. "Health impacts of air pollution". In: (2024). Accessed: 2025-06-07. URL: https://www.who.int/teams/environment-climate-change-and-health/air-quality-energy-and-health/health-impacts.

[3] Nuria Castell et al. "Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?" In: *Environment International* 99 (2017), pp. 293–302. DOI: 10.1016/j.envint.2016.12.007.

[4] M Van Damme et al. "Long-range transport and evolution of marine boundary layer air pollution over the central Pacific Ocean". In: *Atmospheric Environment* 177 (2018), pp. 143–153. DOI: 10.1016/j.atmosenv.2018.01.035.

[5] Sailowtech. *Sailowtech - Site officiel*. Accessed: 2025-05-24. 2025. URL: https://sailowtech.ch/fr/.

[6] Sailowtech. *Sailowtech on GitHub*. https://github.com/Sailowtech. Accessed: 2025-06-07. 2025.

[7] L. Spinelle et al. "Field calibration of a cluster of low-cost available sensors for air quality monitoring. Part A: Ozone and nitrogen dioxide". In: *Sensors and Actuators B: Chemical* 215 (2015), pp. 249–257. DOI: 10.1016/j.snb.2015.03.031.

[8] Balz Maag, Zimu Zhou, and Lothar Thiele. "A Survey on Sensor Calibration in Air Pollution Monitoring Deployments". In: *IEEE Internet of Things Journal* 5.6 (Dec. 2018), pp. 4857–4870. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2853660. URL: https://ieeexplore.ieee.org/document/8405565 (visited on June 7, 2025).

[9] Wenming Li et al. "Tesla valves and capillary structures-activated thermal regulator". In: *Nature Communications* 14.1 (July 6, 2023). Publisher: Nature Publishing Group, p. 3996. ISSN: 2041-1723. DOI: 10.1038/s41467-023-39289-5. URL: https://www.nature.com/articles/s41467-023-39289-5 (visited on June 5, 2025).

[10] Michael F. Modest. *Radiative Heat Transfer*. 3rd. Academic Press, 2013.

[11] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer*. 4th. Taylor & Francis, 2002.

[12] Frank P. Incropera et al. *Fundamentals of Heat and Mass Transfer*. 7th. John Wiley & Sons, 2011.

[13] ML Zamora et al. "Correlation of low-cost sensor readings with co-located particulate matter reference monitors in field deployments". In: *Atmospheric Measurement Techniques* 12.3 (2019), pp. 1335–1347. DOI: 10.5194/amt-12-1335-2019.

[14] Nicolas Masson, Ricardo Piedrahita, and Michael Hannigan. "Low-cost sensors for the measurement of atmospheric composition: overview of topic and future applications". In: *Atmospheric Measurement Techniques* 8.8 (2015), pp. 3219–3246. DOI: 10.5194/amt-8-3219-2015.

[15] Aaron D. Smith, Chithra Karunakaran, and Wook Lee. "A review of chemiresistive gas sensors based on metal oxide nanostructures for detection of volatile organic compounds". In: *Sensors and Actuators B: Chemical* 166 (2012), pp. 354–368. DOI: 10.1016/j.snb.2012.01.074.

[16] KiCad Developers. *KiCad EDA - Schematic Capture PCB Design Software*. Accessed: 2025-05-24. 2025. URL: https://www.kicad.org/.

[17] Sailowtech. *Sailowtech on GitHub*. Accessed: 2025-05-24. 2025. URL: https://github.com/Sailowtech.

[18] Paul Stoffregen. *Teensy 4.0 Development Board*. Accessed: 2025-05-24. 2025. URL: https://www.pjrc.com/store/teensy40.html.

[19] PJRC. *Teensy 4.0 Technical Specifications*. Accessed: 2025-06-07. 2025. URL: https://www.pjrc.com/store/teensy40.html.

[20] ARM Ltd. "The ARM Cortex-M7 Processor: Enabling High-Performance Embedded Systems". In: *ARM Technical White Paper* (2016). Accessed: 2025-06-07. URL: https://developer.arm.com/documentation/whitepapers/cortex-m7.

[21] ESS Earth Sciences. *Pyranometers – Solar Irradiance Sensors*. Accessed: 2025-05-24. 2025. URL: https://www.essearth.com/pyranometer-solar-irradiance-sensors/.

[22] Electrical4U. *Seebeck Effect and Seebeck Coefficient*. Accessed: 2025-05-24. 2023. URL: https://www.electrical4u.com/seebeck-effect-and-seebeck-coefficient/.

[23] *Photosynthetic Plant Sensor - RS485 - Micro Robotics*. URL: https://www.robotics.org.za/PHOTO-RS485 (visited on June 8, 2025).

[24] AliExpress. *Pyranometer Sensor Module for Arduino*. Product page accessed on AliExpress. 2025. URL: https://fr.aliexpress.com/item/1005004266688857.html (visited on May 24, 2025).

[25] Nisal Kobbekaduwa, W. R. de Mel, and Pahan Oruthota. "Calibration and Implementation of Heat Cycle Requirement of MQ-7 Semiconductor Sensor for Detection of Carbon Monoxide Concentrations". In: *Advances in Technology* 1.2 (2021). Accessed: 2025-05-24, pp. 377–392. DOI: 10.31357/ait.v1i2.5068. URL: https://www.researchgate.net/publication/356979922_Calibration_and_Implementation_of_Heat_Cycle_Requirement_of_MQ-7_Semiconductor_Sensor_for_Detection_of_Carbon_Monoxide_Concentrations.

[26] Hanwei Electronics Co., Ltd. *MQ-7 Semiconductor Sensor for Carbon Monoxide*. https://cdn.sparkfun.com/assets/b/b/b/3/4/MQ-7.pdf. Accessed: 2025-06-05. Zhengzhou, China, 2007. URL: https://cdn.sparkfun.com/assets/b/b/b/3/4/MQ-7.pdf.

[27] Winsen Electronics. *MQ131 Gas Sensor (Low Concentration) Datasheet*. Accessed: 2025-05-25. 2017. URL: https://cdn.sparkfun.com/assets/9/9/6/e/4/mq131-datasheet-low.pdf.

[28] The Engineering Projects. *Introduction to MQ131 Ozone Gas Sensor*. Accessed: 2025-05-25. 2024. URL: https://www.theengineeringprojects.com/2024/03/mq131-ozone-gas-sensor.html.

[29] Sensirion AG. *SPS30 Particulate Matter Sensor Datasheet*. https://cdn.soselectronic.com/productdata/0c/0f/243ffc55/sps30-2.pdf. Accessed: 2025-05-25. 2023.

[30] Sensirion AG. *Particulate Matter Sensing for Air Quality Measurements*. Accessed: 2025-05-25. 2021. URL: https://sensirion.com/products/product-insights/specialist-articles/particulate-matter-sensing-for-air-quality-measurements.

[31] Jessica Tryner and John Volckens. "Laboratory Comparison of Low-Cost Particulate Matter Sensors to Measure Transient Events of Pollution - Part B - Particle Number Concentrations". In: *Sensors* 23.17 (2023), p. 7657. DOI: 10.3390/s23177657. URL: https://www.mdpi.com/1424-8220/23/17/7657.

[32] Sensirion AG. *SPS30 Particulate Matter Sensor Datasheet*. Version 1.0, Document Number: SPS30-Datasheet. July 2019. URL: https://sensirion.com/media/documents/8600FF88/64A3B8D6/Sensirion_PM_Sensors_Datasheet_SPS30.pdf.

[33] Bosch Sensortec. *BME280 Environmental Sensor*. Accessed: 2025-05-24. 2025. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/.

[34] SunFounder. *BMP280 Sensor — Ultimate Sensor Kit Documentation*. Accessed: 2025-05-24. 2025. URL: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/14-component_bmp280.html.

[35] Bosch Sensortec GmbH. *BME280 Combined Humidity and Pressure Sensor Datasheet*. Revision 2.0, Document Number BST-BME280-DS002. May 2018. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf.

[36] SunFounder. *Accelerometer & Gyroscope Module (MPU6050) — Ultimate Sensor Kit Documentation*. Accessed: 2025-05-24. 2025. URL: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/05-component_mpu6050.html.

[37] TDK InvenSense. *MPU-6000 and MPU-6050 Product Specification*. Accessed: 2025-06-07. TDK Corporation. 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[38] *NMEA 2000*. https://en.wikipedia.org/wiki/NMEA_2000. Accessed: 2025-06-07. 2025.

[39] Timo Lappalainen Terry. *NMEA2000 Arduino Library GitHub Repository*. Accessed: 2025-06-07. 2025. URL: https://github.com/ttlappalainen/NMEA2000.

[40] National Marine Electronics Association. *NMEA 2000 Standard Overview*. Accessed: 2025-06-07. 2019. URL: https://www.nmea.org/nmea-2000.html.

[41] Wilfried Voss. *Technical Report: Teensy 4.0 NMEA 2000 Simulator*. Accessed: 2025-06-07. 2025. URL: https://jcom1939.com/technical-report-teensy-4-0-nmea-2000-simulator/.

[42] Fulvia Malvido Montandon Nils Manny. "PENS-490, ENAC Project, low-tech weather station". In: (2025).

[43] Bosch Sensortec. *BME280 Combined Humidity and Pressure Sensor*. Datasheet Revision 1.8. 2018. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (visited on June 4, 2025).

[44] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. "Nonlinear complementary filters on the special orthogonal group". In: *IEEE Transactions on Automatic Control* 53.5 (2008), pp. 1203–1218. DOI: 10.1109/TAC.2008.923738.

[45] Farwah Nawazi. *MQ7 Carbon Monoxide (CO) Gas Sensor Module*. Accessed: 2025-05-24. 2021. URL: https://www.circuits-diy.com/mq7-carbon-monoxide-co-gas-sensor-module/.

[46] Bastian Maag, Zimu Zhou, and Lothar Thiele. "A survey on sensor calibration in air pollution monitoring deployments". In: *IEEE Internet of Things Journal* 5.6 (2018), pp. 4857–4870. DOI: 10.1109/JIOT.2018.2866743.

[47] Microchip Technology Inc. *Using CAN Bus in Marine Applications*. Accessed: 2025-06-07. 2020. URL: https://www.microchip.com/en-us/solutions/automotive/networking/can-bus.

[48] Angelo M. Sabatini. "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing". In: *IEEE Transactions on Biomedical Engineering* 53.7 (2006), pp. 1346–1356. DOI: 10.1109/TBME.2006.875664.

[49] *What's my Right to Repair?* Right to Repair Europe. URL: https://repair.eu/whats-my-right-to-repair/ (visited on June 2, 2025).

[50] *Ambient (outdoor) air pollution*. URL: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health (visited on June 2, 2025).

A Appendix

A.1 MQ131 calibration curve

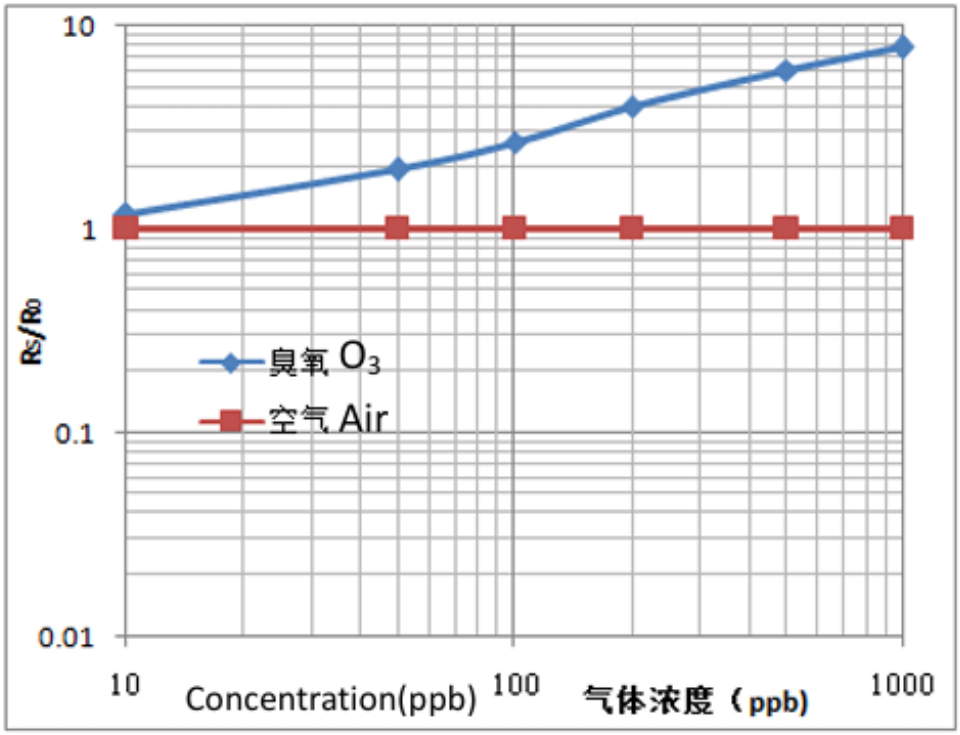


Figure 26: mq131 calibration curve found in Source [27]

Based on Figure 26, concentrations in ppm are deduced according to the relationship ppb = 1000 x ppm. The blue curve seems to correspond to a power law $y = B \cdot x^a$. Its coefficient can be deduced by applying a log transform to both sides of the equation: $\log_{10}(y) = b + a \log_{10}(x)$ with $b = \log_{10}(B)$. Solving a system of two equations using 0.01 ppm and 1 ppm as x values and approximating the corresponding y values to 1 and 8 respectively: $a \approx 0.45$ and $b \approx 0.90$.

A.2 MQ7 calibration curve

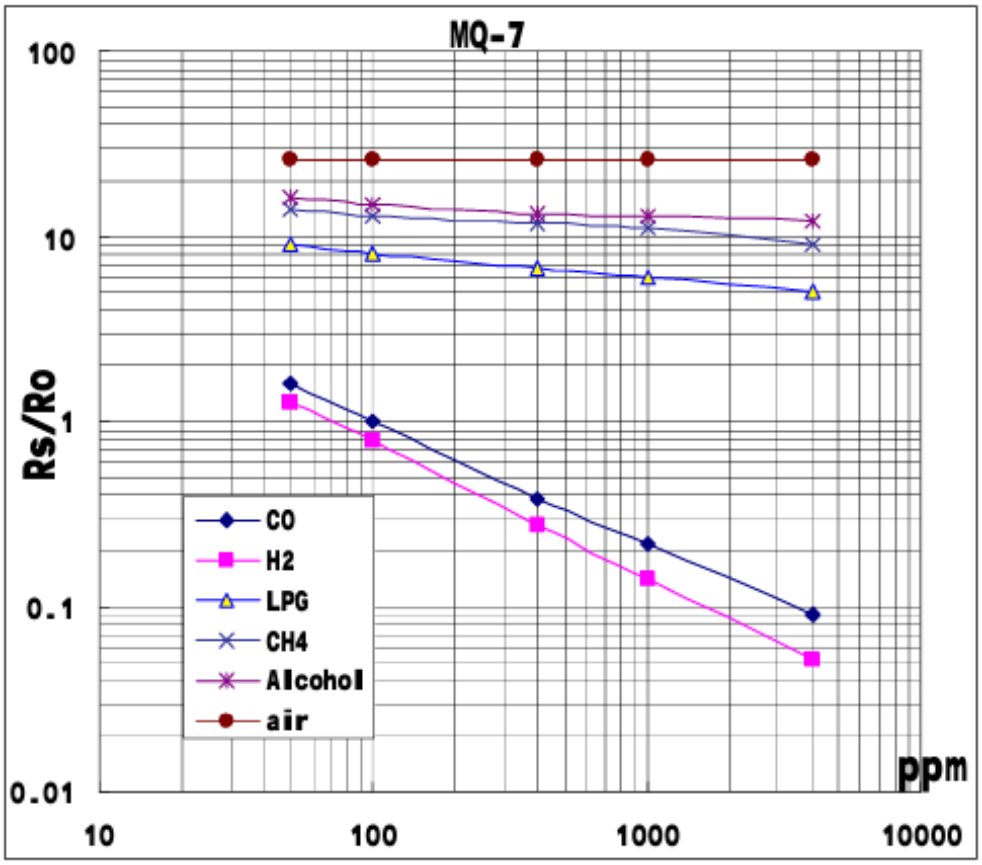


Figure 27: MQ7 calibration curve found in Source [26]. Only the CO curve (dark blue square curve) is relevant to the calibration. The other curves indicate that the sensor can react to other chemical species.

Based on Figure 27. The blue curve seems to correspond to a power law $y = B \cdot x^a$. Its coefficient can be deduced by applying a log transform to both sides of the equation: $\log_{10}(y) = b + a \log_{10}(x)$ with $b = \log_{10}(B)$. Solving a system of two equations using 100 ppm and 1000 ppm as x values and approximating the corresponding y values to 1 and 0.2 respectively: $a \approx -0.7$ and $b \approx 1.4$.

A.3 MQ temperature & humidity corrections

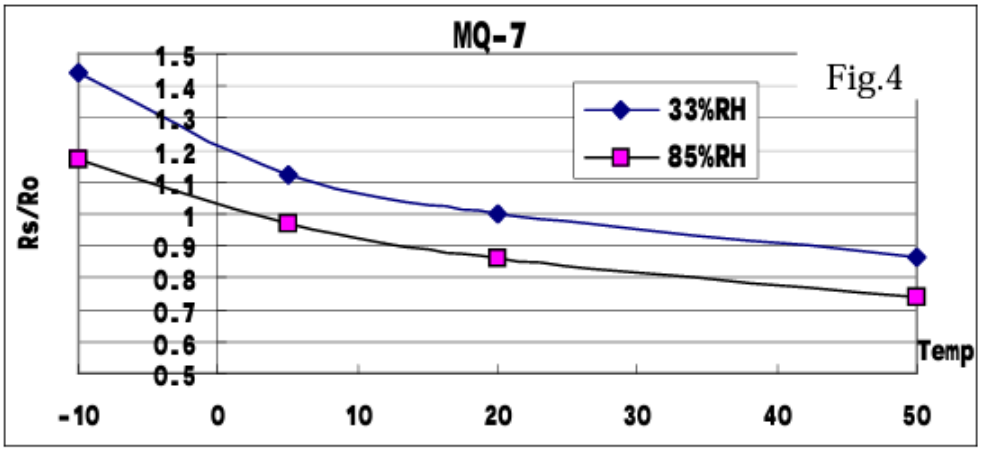


Figure 28: MQ-7 R_S/R_0 variation with temperature and relative humidity for a fixed concentration of CO.

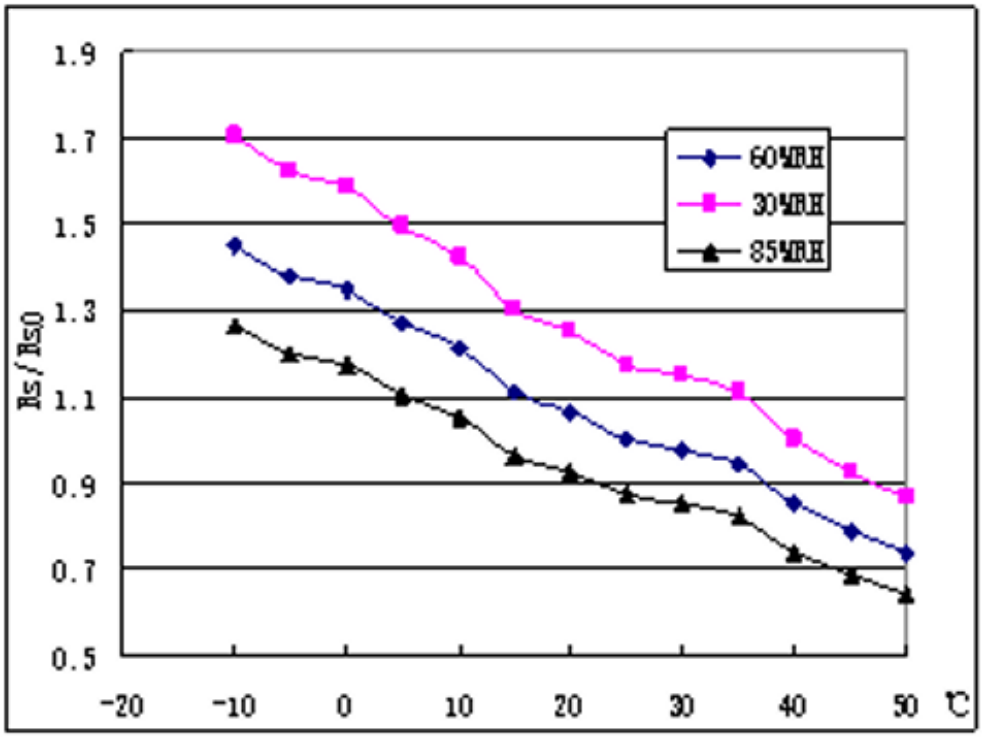


Figure 29: MQ-131 R_S/R_0 variation with temperature and relative humidity for a fixed concentration of O_3.

A.4 WiFi code: 'ESP-web-server.cpp'

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

WiFiServer tcpServer(23);          // Serveur TCP pour la communication série
WiFiClient tcpClient;

ESP8266WebServer webServer(80);    // Serveur HTTP

void handleDownload() {
  Serial.write("read measurements\n");
  delay(100);

  String file = "";
  unsigned long timeout = millis() + 2000;

  while (millis() < timeout) {
    while (Serial.available()) {
      char c = Serial.read();
      file += c;

      // Relancer le timeout si des données arrivent encore
      timeout = millis() + 100;
    }
  }

  // Important : forcer le bon en-tête HTTP + nom du fichier
  webServer.sendHeader("Content-Type", "text/csv");
  webServer.sendHeader("Content-Disposition", "attachment; filename=\"meteo.csv\"");
  webServer.send(200, "text/csv", file);
}


void handleRoot() {
  String page = "<!DOCTYPE html><html><head><meta charset='UTF-8'>";
  page += "<title>Station Météo</title>";
  page += "<style>";
  page += "body { font-family: Arial, sans-serif; background-color: #f0f4f8; color: #333; text-align: center; paddin
  page += "h1 { color: #2c3e50; }";
  page += "p { font-size: 18px; margin-bottom: 30px; }";
  page += "button { background-color: #3498db; color: white; border: none; padding: 15px 25px; font-size: 16px; bord
```

```
  page += "button:hover { background-color: #2980b9; }";
  page += "</style></head><body>";
  page += "<h1>Bienvenue sur la station météo</h1>";
  page += "<p>Station connectée et serveur web actif.</p>";
  page += "<a href='/download'><button>télécharger données</button></a>";
  page += "</body></html>";
  webServer.send(200, "text/html", page);
}

void setup() {
  Serial.begin(115200);

  // Création du point d'accès WiFi avec mot de passe
  WiFi.softAP("TeensyBridge", "12345678");

  // Démarrage serveur TCP
  tcpServer.begin();
  tcpServer.setNoDelay(true);

  // Démarrage serveur web HTTP
  webServer.on("/", handleRoot);  // Définit la page à servir sur "/"
  webServer.on("/download", handleDownload);
  webServer.begin();

  // Serial.println("WiFi AP started. Waiting for connections...");
  // Serial.print("IP address: ");
  // Serial.println(WiFi.softAPIP());  // Affiche l'IP, normalement 192.168.4.1
}

void loop() {
  // Gérer les connexions serveur web
  webServer.handleClient();

  // Gérer la connexion TCP
  if (!tcpClient || !tcpClient.connected()) {
    WiFiClient newClient = tcpServer.available();
    if (newClient) {
      tcpClient = newClient;
      // Serial.println("TCP Client connected");
    }
  }

  if (tcpClient && tcpClient.connected()) {
    // Du WiFi vers Teensy
    //while (tcpClient.available()) {
    //  char c = tcpClient.read();
    //  Serial.write(c);
    //}

    // Du Teensy vers WiFi
    while (Serial.available()) {
      char c = Serial.read();
      tcpClient.write(c);
    }
  }
}
```

A.5 Teensy Code : 'final-code.ino'

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Adafruit_MPU6050.h>
#include <math.h>
#include <SPI.h>
#include <SD.h>


// SD Card Setup
File fichier;
const int chipSelect = 10;

// === Sensors ===
```

```
15
16  // Accelerometor and gyroscope
17  Adafruit_MPU6050 mpu;
18  float gyroBiasX = 0, gyroBiasY = 0, gyroBiasZ = 0;
19
20
21  // Temperature, Pressure and Humidity
22  Adafruit_BME280 bme;
23
24  // Pyranometer (MODBUS frame on Serial3)
25  byte message[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x01, 0x84, 0x0A};
26  float light = 0;
27  float correctedLight = 0;
28
29  // MQ-7
30  const int mq7Pin = A8; // Pin 22 on teensy 4.0
31  const float mq7_RL = 10000.0; // ohms to be confirmed
32  const float mq7_Vcc = 5.0; // calibrated
33  float mq7_R0 = 10000.0; // ohms To be calibrated
34
35  // MQ-131
36  const int mq131Pin = A9; // Pin 23 on teensy 4.0
37  const float mq131_RL = 10000.0; // supposed to be the correct one
38  const float mq131_Vcc = 5; //make sur it is the correct voltage to use
39  float R0 = 5172.13; // Fisrt indoors calibrated value 5172.13 when using a Vcc of 5V, 18.41 when using a Vcc of 3.3V
40
41  // MPU6050 vars
42  float rollEst = 0, pitchEst = 0;
43  float rollAcc, pitchAcc;
44  float alpha = 0.80; // 80% percent given to the accelerometer, 20% to the gyro
45
46  // Sensor flags
47  bool mpuOK = false;
48  bool bmeOK = false;
49  bool pyranoOK = true;
50
51  void calibrateGyro() {
52    Serial2.println("Calibrating gyro... Keep device still");
53    const int samples = 500;
54    float sumX = 0, sumY = 0, sumZ = 0;
55
56    for (int i = 0; i < samples; i++) {
57      sensors_event_t a, g, t;
58      mpu.getEvent(&a, &g, &t);
59      sumX += g.gyro.x;
60      sumY += g.gyro.y;
61      sumZ += g.gyro.z;
62      delay(2);
63    }
64
65    gyroBiasX = sumX / samples;
66    gyroBiasY = sumY / samples;
67    gyroBiasZ = sumZ / samples;
68
69    Serial2.println("Gyro bias calibrated.");
70  }
71
72  void setup() {
73    Serial2.begin(115200); //74880 pour affichage sur termainal arduino
74    Serial3.begin(9600); // Pyranometer MODBUS baud
75    delay(2000);
76
77    Serial2.println("=== SD Card initialization ===");
78    if (!SD.begin(chipSelect)) {
79      Serial2.println("Erreur : carte SD non détectée !");
80      return;
81    }
82
83    Serial2.println("=== Sensor initialization ===");
84
85    // I2C
86    Wire.begin();       // BME280
87    Wire1.begin();      // MPU6050
```

```
88
89    // MPU6050
90    Serial2.println("→ MPU6050...");
91
92    if (mpu.begin(MPU6050_I2CADDR_DEFAULT, &Wire1)) {
93      Serial2.println("   MPU6050 detected.");
94      mpuOK = true;
95      mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
96      mpu.setGyroRange(MPU6050_RANGE_250_DEG);
97      mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
98    } else {
99      Serial2.println("   MPU6050 not detected!");
100   }
101
102   calibrateGyro(); // Calibrate before loop starts
103
104   // BME280
105   Serial2.println("→ BME280...");
106   if (bme.begin(0x76)) {
107     Serial2.println("   BME280 detected.");
108     bmeOK = true;
109   } else {
110     Serial2.println("   BME280 not detected!");
111   }
112
113   Serial2.println("→ Pyranometer ready on Serial3 (MODBUS 9600).\n");
114
115 }
116
117 void readPyranometer() {
118   Serial3.write(message, sizeof(message));
119   delay(1000);
120   if (Serial3.available() >= 7) {
121     Serial3.read(); Serial3.read(); Serial3.read(); // Header
122     light = (256 * Serial3.read()) + Serial3.read(); // Data
123     light *= 0.207; // Convert to W/m²
124     Serial3.read(); Serial3.read(); // CRC
125   } else {
126     Serial2.println("    Invalid or missing pyranometer response.");
127     pyranoOK = false;
128   }
129 }
130
131 float readMQ7() {
132   int rawValue = analogRead(mq7Pin);
133   float voltage = rawValue * (3.3 / 1023.0);  // Teensy reads in 3.3V, so scale to 3.3V
134
135   // Calculate sensor resistance Rs
136   float Rs = mq7_RL * ((mq7_Vcc - voltage) / voltage); // Powered by 5V
137
138   // Calculate Rs/R0 ratio
139   float ratio = Rs / mq7_R0;
140
141   // Estimate CO ppm using log formula
142   // You may need to adjust a and b based on your specific MQ-7 version and datasheet graph
143   float a = -0.7; // TO BE UPDATED based on your sensor's datasheet
144   float b = 1.4;  // TO BE UPDATED based on your sensor's datasheet
145   float mq7_ppm = pow(10, (1/a * log10(ratio) - b/a));
146
147   Serial2.print("MQ-7 Raw ADC: "); Serial2.print(rawValue);
148   Serial2.print(" | Voltage: "); Serial2.print(voltage, 2); Serial2.print(" V");
149   Serial2.print(" | Rs: "); Serial2.print(Rs, 2);
150   Serial2.print(" | Ratio (Rs/R0): "); Serial2.print(ratio, 2);
151   Serial2.print(" | Estimated CO: "); Serial2.print(mq7_ppm, 2); Serial2.println(" ppm");
152
153   return mq7_ppm;
154 }
155
156 float readMQ131() {
157   int rawValue = analogRead(mq131Pin);
158   float voltage = rawValue * (3.3 / 1023.0);  // Teensy reads in 3.3V, so scale to 3.3V
159
160   // Sensor resistance
```

```
  float Rs = mq131_RL * ((mq131_Vcc - voltage) / voltage);

  // Ratio Rs/R0
  float ratio = Rs / R0;

  // Using an approximate logarithmic formula (from datasheet curve)
  // log10(ppm) = a * log10(Rs/R0) + b
  float a = 0.45; //Approximated from datasheet
  float b = 0.90; // Approximated from datasheet
  float mq131_ppm = pow(10, (1/a * log10(ratio) - b/a));

  Serial2.print("MQ-131 Raw ADC: "); Serial2.print(rawValue);
  Serial2.print(" | Voltage: "); Serial2.print(voltage, 2); Serial2.print(" V");
  Serial2.print(" | Rs: "); Serial2.print(Rs, 2);
  Serial2.print(" | Ratio (Rs/R0): "); Serial2.print(ratio, 2);
  Serial2.print(" | Estimated O3: "); Serial2.print(mq131_ppm, 2); Serial2.println(" ppm");

  return mq131_ppm;
}

void loop() {
  static unsigned long prevTime = millis();
  unsigned long currTime = millis();
  float deltaTime = (currTime - prevTime) / 1000.0;
  prevTime = currTime;

  // Declaration variables
  float temp = NAN, hum = NAN, pres = NAN;
  float pitch = 0.0, roll = 0.0, theta = 0.0;
  float correctedIrradiance = 0.0;
  float mq7_ppm=0.0;
  float mq131_ppm = 0.0;

 // respond to CSV file requests from the ESP module
if (Serial2.available() > 0) {
  String cmd = Serial2.readStringUntil('\n');
  cmd.trim();

  if (cmd == "read measurements") {
    Serial2.println("Received read request");

    fichier = SD.open("meteo.csv");
    if (fichier) {
      while (fichier.available()) {
        String line = fichier.readStringUntil('\n');  // Lit une vraie ligne
        Serial2.println(line);  // Envoie la ligne avec saut de ligne
      }
      fichier.close();
    } else {
      Serial2.println("Erreur : fichier introuvable");
    }

    delay(1000);
    return;
  }
}

  // Open the SD card for writting
  fichier = SD.open("meteo.csv", FILE_WRITE);
   Serial2.println("\n============ SENSOR DATA ============");

  if (fichier.size() == 0) {
    fichier.println("Time(s),Temperature(°C),Humidity(%),Pressure(hPa),Pitch(°),Roll(°),Theta(°),Light(W/m²),Correct
  }


  // === Step 1: Read pyranometer ===
  readPyranometer();  // updates `light`

  // === Step 2: Get MPU6050 data and compute tilt ===
  if (mpuOK) {
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
```

```
234
235       float ax = a.acceleration.x;
236       float ay = a.acceleration.y;
237       float az = a.acceleration.z;
238       float gx = (g.gyro.x - gyroBiasX) * 180.0 / PI;
239       float gy = (g.gyro.y - gyroBiasY) * 180.0 / PI;
240
241       // Accelerometer-based pitch and roll
242       pitchAcc = atan2(-az, sqrt(ay * ay + az * az)) * 180.0 / PI;
243       rollAcc = atan2(ax, ay) * 180.0 / PI; // True formula is atan2(ay/az) but the axes are defined differently with
244
245
246       pitchEst = alpha * pitchAcc + (1 - alpha) * (pitchEst + gx * deltaTime);
247       rollEst = alpha * rollAcc + (1 - alpha) * (rollEst + gy * deltaTime);
248
249       pitch = pitchEst;
250       roll = rollEst;
251
252       theta = sqrt(pitchEst * pitchEst + rollEst * rollEst);
253       correctedIrradiance = light / cos(theta * PI / 180.0);
254
255       Serial2.print("Inclination | Pitch: "); Serial2.print(pitchEst);
256       Serial2.print("°, Roll: "); Serial2.print(rollEst);
257       Serial2.print("°, Theta: "); Serial2.println(theta);
258
259       //Serial2.print("Light (raw): "); Serial2.print(light); Serial2.println(" W/m²");
260       Serial2.print("Corrected Light: "); Serial2.print(correctedIrradiance); Serial2.println(" W/m²");
261     } else {
262       Serial2.println(" MPU6050 data not available.");
263     }
264
265     // === Step 3: Read BME280 ===
266     if (bmeOK) {
267       temp = bme.readTemperature();
268       hum = bme.readHumidity();
269       pres = bme.readPressure() / 100.0F;
270
271       Serial2.print("Temperature: "); Serial2.print(temp); Serial2.println(" °C");
272       Serial2.print("Humidity: "); Serial2.print(hum); Serial2.println(" %");
273       Serial2.print("Pressure: "); Serial2.print(pres); Serial2.println(" hPa");
274       //fichier.println(String(currTime)+","+String(temp)+","+String(hum)+","+String(pres));
275     } else {
276       Serial2.println(" BME280 data not available.");
277     }
278
279     // === Step 4: Read Gas sensors ===
280     mq131_ppm = readMQ131();
281     mq7_ppm = readMQ7();
282
283     // === Step 5: Write data to CSV ===
284     fichier.println(
285       String(deltaTime, 3) + "," +
286       String(temp, 2) + "," +
287       String(hum, 2) + "," +
288       String(pres, 2) + "," +
289       String(pitch, 2) + "," +
290       String(roll, 2) + "," +
291       String(theta, 2) + "," +
292       String(light, 2) + "," +
293       String(correctedIrradiance, 2) + "," +
294       String(mq7_ppm, 2) + "," +    // Valeur placeholder pour CO (ppm)
295       String(mq131_ppm, 2)            // Valeur placeholder pour O3 (ppm)
296     );
297
298     fichier.close();
299     Serial2.println("========================================");
300     delay(5000);
301   }
302
303
304
```