Universidade do Minho
Escola de Engenharia

Rui Manuel Pereira da Costa

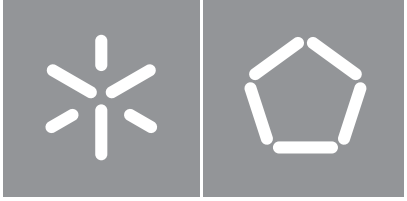
Hardware design and integration of BME688 Sensor into CPS devices for gas monitoring applications

dezembro de 2022

Hardware design and integration of BME688 Sensor into CPS devices for gas monitoring applications

Rui Costa

UMinho | 2022



Universidade do Minho
Escola de Engenharia

Rui Manuel Pereira da Costa

Hardware design and integration of BME688 Sensor into CPS devices for gas monitoring applications

Dissertação de Mestrado
Mestrado em Engenharia Eletrónica Industrial e Computadores
Sistemas Embebidos e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Jorge Cabral

February, 2023

Agradecimentos

Agradeço primeiramente ao Professor Doutor Jorge Cabral pela confiança depositada no meu trabalho, pela opurtunidade dada para realização do projeto e por todo o conhecimento transmitido ao longo dos anos.

Deixo um agradecimento a todas as pessoas do laboratório do ESRG, em especial ao professor João Carvalho por toda a ajuda prestada na elaboração da dissertação. À professora Sofia Paiva e ao professor Rui Machado por todo o acompanhamento e ao Rui Teixeira por todo o conhecimento transmitido.

A todos os amigos que me acompanharam, um obrigado, em particular ao parti all naite por todo o apoio ao longo dos anos e pela amizade incondicional e de longa duração. Agradeço ainda aos amigos que fiz durante este percurso por tornarem este caminho mais emocionante e desafiador.

Por fim, o maior agradecimento vai para toda a minha família pela motivação, carinho e investimento durante todos estes anos. Às minhas irmãs, e aos meus pais por toda o amor, incentivo e atenção. À minha madrinha que desde cedo apoiou o meu percurso escolar. Á tia do Carrinho pelas tardes de leitura e conversas infindáveis. Á tia Tónia por toda a amabilidade e ajuda prestada. Um especial obrigado a minha namorada Eva por toda a ajuda, paciência e apoio incondicional em todos os momentos.

Assim, agradeço profundamente a todos os que direta ou indiretamente participaram neste percurso e que nunca deixaram de acreditar em mim. Recordarei todos com carinho.

Project "(Link4S)ustainability - A new generation connectivity system for creation and integration of networks of objects for new sustainability paradigms [POCI-01-0247-FEDER-046122 | LISBOA-01-0247-FEDER-046122]" is financed by the Operational Competitiveness and Internationalization Programmes COMPETE 2020 and LISBOA 2020 under the PORTUGAL 2020 Partnership Agreement, and through the European Structural and Investment Funds in the FEDER component.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Resumo

Desenvolvimento de *Hardware* e integração do sensor BME688 em dispositivos CPS para aplicações de monitorização de gás

Os efeitos de diferentes gases podem ser prejudiciais tanto para o ambiente como para os seres humanos. Como tal a monitorização e deteção de gases é importante já que pode contribuir para um aumento da segurança em diversas aplicações e áreas. A indústria mineira, a automóvel, áreas médicas e áreas especializadas na monitorização da qualidade de ar em diversos locais têm contribuído para um aumento no estudo da deteção de gases.

O objetivo da dissertação é o desenvolvimento de *hardware* para implementar o sensor de deteção de gás BME688, e implementação de algoritmos para deteção dos gases CO_2 e H_2. Há muitos locais, onde não é possível ter uma fonte de alimentação direta a fornecer energia aos circuitos eletrónicos. Nesta dissertação serão utilizadas baterias, e como forma de reduzir custos de manutenção na troca de baterias e de deslocações ao local a monitorizar, a placa de circuito impresso a desenvolver será de baixo consumo energético.

Foi realizado o estudo do sensor BME688, visando o seu funcionamento e a análise do seu comportamento relativamente á deteção dos gases CO_2 e H_2. Para desenvolver este estudo foi utilizado um kit de desenvolvimento comercial que integrava estes sensores. Recorrendo ao kit do BME688 foi possível recolher um dataset dos dois gases alvo e posteriormente foram desenvolvidos os circuitos electrónicos (PCB), e implementados os algoritmos de AI para deteção de CO_2 e H_2.

Relativamente ao *hardware*, a placa foi desenvolvida para optimização do seu consumo energético. Todos os circuitos eletrónicos foram dimensionados com esse propósito. Foram analisados os consumos energéticos e comparados com o kit existente no mercado. Relativamente ao software, foram implementados algoritmos de AI. Foram implementadas soluções baseadas em *decision tree*, *support vector machines* e rede neuronal. Foram analisados os consumos de todos os algoritmos, os seus resultados e o tempo de execução dos mesmos. Foi ainda desenvolvido um estudo para estimar a duração de uma bateria.

Foi possível, com todos os resultados, avaliar o sistema desenvolvido e apresentar sugestões para trabalho futuro.

Palavras-chave: deteção de gás, sensor BME688, baixo consumo, *machine learning*

Abstract

Hardware design and integration of BME688 sensor into CPS devices for gas monitoring applications

The effects of several gases can be harmful to both the environment and humans. Therefore, gas monitoring and detection devices are vital as they increase safety in various applications and areas. The mining, automotive, medical, and other industries have contributed to an increase in the study of gas detection and monitoring air quality in various locations.

Thus, the dissertation's objective is to develop hardware to implement the BME688 gas detection sensor and the corresponding algorithms for CO_2 and H_2 gas detection. In several places, it is not possible to have a direct power source to supply energy to electronic circuits. In this dissertation, batteries will be used. As a way to reduce maintenance costs in changing batteries and trips to the site to be monitored, the printed circuit board to be developed will be of low energy consumption.

The BME688 sensor was studied, aiming at its operation and behavior regarding the gases CO_2 and H_2. To develop this study it was used a commercial development kit (reference board) that integrated these sensors. Using the BME688 kit it was possible to collect a dataset of the two target gases. Afterward, the electronic circuits (PCB) were developed, and the AI algorithms for CO_2 H_2 detection were implemented.

Regarding *hardware*, the board was developed for optimization of its energy consumption. All the electronic circuits were sized for that purpose. The energy consumption was analyzed and compared with the existing kit in the market. Regarding software, AI algorithms were implemented. Solutions based on *decision tree*, *support vector machines*, and neural networks were implemented. The consumption of all algorithms, their results, and their execution time were analyzed. A study was also developed to estimate the lifetime of a battery.

With all the results, it was possible to evaluate the developed system and make suggestions for future work.

Keywords: gas detection, BME688 sensor, low-power, machine learning

Contents

List of Figures

List of Tables

Listings

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
ASIC	Application-specific Integrated Circuit
BJT	Bipolar junction transistor
BoM	Bill Of Materials
CE	Counter Electrode
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
ESD	Electrostatic discharge
ESL	Equivalent Series Inductance
ESR	Equivalent Series Resistance
GPIO	General Purpose Input/Output
HW	Hardware
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IR	Infrared Radiation
JATG	Joint Test Action Group
LDO	Low Dropout Output

LUT	Lookup Table
MCU	Microcontroller
MEMS	Microelectromechanical systems
MOS	Metal Oxide Semiconductor
NDIR	Nondispersive Infrared sensor
NN	Neural Network
PCB	Printed Circuit Board
RBF	Radial Basis Function
RE	Reference Eletrode
RF	Radio Frequency
RTC	Real Time Clock
SoC	System on Chip
SPI	Serial Peripheral Interface
SVM	Support-Vector Machine
SW	Software
SWD	serial wire debug
TDLAS	Tunable diode laser absorption spectroscopy
TVS	Transient Voltage Suppressors
UART	Universal asynchronous receiver/transmitter
VOCs	Volatile Organic Compounds
WE	Working Electrode

1 Introduction

This chapter presents the contextualization, the motivation, the objectives, and the structure of the dissertation. In the contextualization, the general theme and the perspective of the dissertation frame will be introduced. Next, the motivation for developing the dissertation study and its objectives are presented and finally, the structure followed for the development of the dissertation will be described.

1.1 Contextualization and Motivation

Gas leakage detection methods became a concern after the effects of harmful gases on human health were discovered. Sometimes early detection of gases can solve many problems associated with their negative effects. Before the advent of modern electronic sensors, the detection was done using rudimentary methods. Throughout the 19th and early 20th centuries, coal miners would bring canaries into tunnels to detect noxious gases such as carbon dioxide, carbon monoxide, and methane. In the presence of these gases, the canary would stop singing or even die, being a warning to the miners [21]. Due to demands from industry, gas detection sensors have been evolving over time, and gas sensors have gained high importance [23].

Environmental gas detection has become essential in diverse fields and applications, aiming to develop a sensor capable of collecting information about numerous gases. That information can prevent accidents, avoid equipment malfunctions, provide warnings about air pollution, or even help hospital patients.

However, in the different gas detection scenarios, the sensors have shortcomings. These include cross-sensitivity, and low selectivity [20]. Software techniques are used to increase sensitivity and efficiency without additional costs or hardware modifications. These bring advantages for gas detection by overcoming hardware deficiencies. These techniques include, for example, artificial intelligence algorithms [20]. However these algorithms can be very complex and challenging to implement on microcontrollers which requires an efficient implementation.

The power consumption of these sensors can be high since many technologies require the sensor material to be heated to high temperatures. If the application of these sensors is such that there is direct access to a power source, these characteristics are less irrelevant. However, for many applications, the sensors do not have a power supply nearby and need to be powered by a battery. Long battery life is

important to reduce maintenance costs in these cases. If the sensor consumes a lot of power, this increases these costs. Also, the board where the sensor is implemented needs to be low power, and the software developed needs features aimed at low power management as well.

1.2 Main Goal

The main goal of this dissertation is the development of a board that implements the BME688 gas detection sensor and the implementation of gas detection algorithms. The board will have to be low-power to guarantee a large lifetime of the device. To achieve this, the following objectives were set:

- **Explore DevKit Board [61]**

  The goal is to explore an existing board that integrates the BME688 sensor (reference board). This board has eight BME688 sensors. It intends to explore how the BME688 sensor works and the hardware developed on this board.

- **Explore the detection of harmful gases using the BME688**

  The goal is to explore the behavior of the sensor for different gases. In this dissertation, hydrogen and carbon dioxide will be explored. The behavior of the sensor for different concentrations of these gases will be explored. It is entended to acquire a valid sensor dataset. The goal is to take a valid dataset from the sensor when exposed to a certain concentration of gas so that gas detection algorithms can be developed in the future.

- **Hardware Development**

  The goal is to develop a new board that implements the interface to BME688 sensor and the processing capabilities to implement the detection algorithms. This board will have to be low-power with as few BME688 as possible. It has to include a low-power MCU and a modem in order to be able to send gathered gas data to the cloud.

- **Implementation of artificial intelligence algorithms**

  The goal is the implementation of artificial intelligence algorithms for gas detection in a microcontroller. These algorithms were obtained on the data set taken for the target gases. The implemented algorithms will be analyzed for their execution time.

- **Consumption Analysis**

  The goal is to analyze the consumption of the BME688 sensor. In addition to this, it is intended to analyze the consumption of the implemented algorithms. Finally, it is intended to analyze the energy consumption in a real context, with the sensor and the chosen gas detection algorithm, to study battery life.

1.3 Structure

The document is structured in six chapters. The first chapter is the current one, in which contextualization and motivation were made, and the objectives of the dissertation were presented.

The second chapter presents the literature review, providing a theoretical introduction to the different gas detection techniques. It then presents the possible applications of each of the techniques. Market research for different types of existing sensors was made. An extensive review of the BME688 sensor was carried out. Also, a review of different machine-learning techniques is made. Finally, a study of *hardware* design and the most important concepts in this area is made.

The third chapter presents the system specifications. This chapter is divided into two parts, *hardware*, and *software*. Initially, the requirements for solving the problem are seen followed by a justification of the hardware and software choices made to solve the problem. The power, MCU, modem, and sensor circuits are presented.

The fourth chapter presents the implementation. The chapter is also subdivided into hardware and software. In the hardware section, the circuits are implemented, considering the choices made in the previous chapter. Finally, the layout of the developed board is described. In the software section, the most important implementations are described. The implementation of the communication between the sensor and the MCU is detailed. The data sets for the target gases CO_2 and H_2 are also studied and analyzed. Finally, the implementations of the algorithms are seen, as well as the implementation of the real context example for analyzing the life of a battery.

The fifth chapter presents the tests and results. This chapter will observe the developed board and the tests performed on it. The tests performed on the sensor and the tests performed on the implemented algorithms will be analyzed.

The last chapter presents the conclusions of the dissertation. This chapter focus on the conclusions obtained during the development of this project. Finally, suggestions for future work are exposed, aiming at the project's development and growth.

2 State of the art

This chapter addresses the main concepts of gas detection technologies, starting by introducing some essential concepts on the subject, followed by a study of each of the most known technologies. After that study, the different applications of each gas detection technology are seen, and the challenges that these sensors present are analyzed. A study of the market for different sensors and types was also made. Different artificial intelligence algorithms are studied, such as decision trees, support vector machines, and simple neural networks. Finally, the main concepts in hardware design are studied.

2.1 Gas detection and relevance

Gas detection has received increasing attention in both industrial and academic research. This relevance is due to the wide applications of gas detection sensors in different areas. Industrial production, for example, in the mining industry for methane detection, the automotive industry for detecting pollutant gases from vehicles, medical applications for simulation of the human olfactory system, air quality monitoring focusing on carbon monoxide detection, and environmental studies for greenhouse gas monitoring, are examples of main areas for which gas detection is of growing importance [40].

2.2 Gas detection technologies

Gas sensors rely on a physical or chemical reaction with the target gas [23]. This reaction generates a response related to the concentration of the gas. Other sensing methods explore the propagation characteristics of the gases to perform measurements and compare them with a reference to determine which gas is being sensed. Other methods are based on comparing physical properties such as wave propagation [23]. Gas sensors are typically classified into several types based on the sensing element with which it is constructed. Figure 1 summarises some of the different methods used in gas detection.

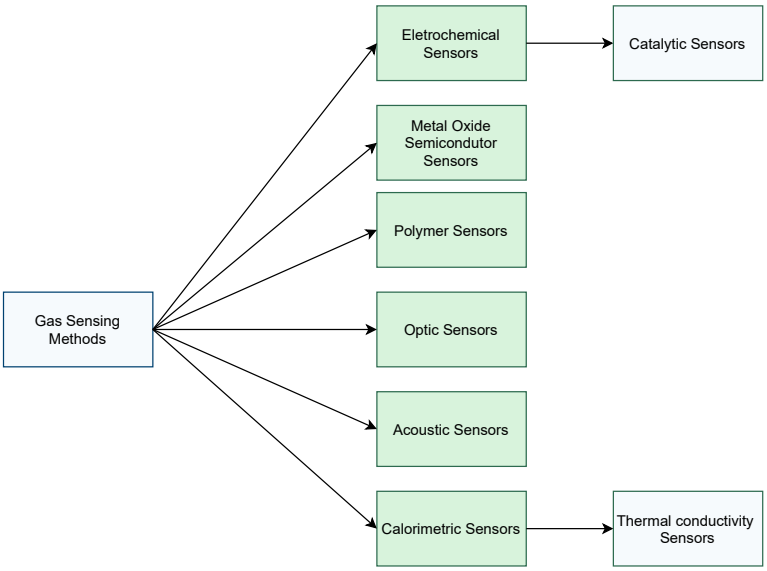


Figure 1: Gas sensing methods [40]

2.2.1 Fundamental concepts

Before explaining the different methods for gas detection, it is important to explain some theoretical fundamentals necessary for a better understanding of these methods. In this topic, some important definitions related to gas detection will be presented, such as the definition of sensitivity, selectivity, response time, recovery time, and working temperature [7].

- The sensitivity of a gas detection sensor is its ability to show a change in the measured signal per unit concentration of the analyte (analyte is a chemical substance or component in a sample that is the target of analysis in an experiment). A sensor with low sensitivity may not detect the presence of a low-concentration gas, while one with high sensitivity may do so.

- From a gas detection perspective, selectivity is the ability to classify different gas compositions with different gas combinations and concentrations. Thus, selectivity is the characteristic that determines whether a sensor responds selectively to a specific gas when it is in an environment with combinations of gases.

- The response time is the time required for the sensor to respond to a variation in the concentration of the target gases.

- The recovery time is the time required for the sensor signal to return to its initial value when the target gas concentration becomes zero.

- The working temperature is the temperature at which the sensor operates for maximum sensitivity. This temperature can vary from sensor to sensor or for detecting different gases.

2.2.2 Electrochemical sensors

The main function of these sensors lies in chemical redox reaction to detect the target gas. This chemical reaction will produce an electrical signal proportional to the concentration of the gas. These sensors require high operating temperatures based on the electrolyte chosen [81].

The electrolyte composition should facilitate the chemical reaction and efficiently carry the ionic charge across the electrodes. The electrolyte solution is mainly composed of acids and bases frequently in gel forms. It must form a stable reference potential with the reference electrode and be compatible with the materials used to construct the sensor. Evaporation of the electrolyte means deterioration of the sensor signal [23] [81].

These sensors are composed of a membrane that separates the environmental gases from the electrolyte solution (hydrophobic barrier), the electrodes, and a capillary-type opening. Figure 2 represents a schematic of the constitution of these sensors.

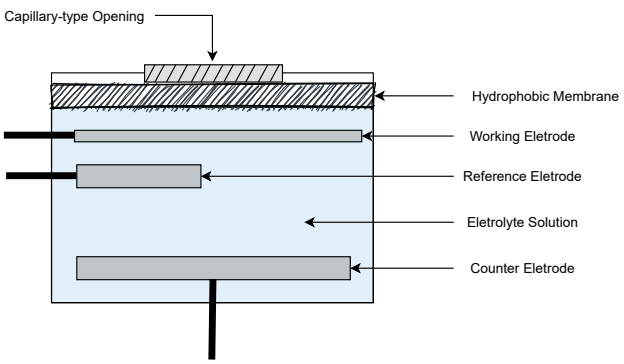


Figure 2: Electrochemical sensor scheme [81]

To allow the required amount of gas to enter the sensor, the gas needs to pass through a capillary-type opening and hydrophobic barrier. This barrier also prevents electrolyte leakage. After reaching the working electrode, the ambient gas causes a redox reaction. The electrode, specifically chosen for a given gas, catalyzes these reactions. By measuring the current between the working electrode (WE) and the counter electrode (CE), it is possible to deduce the amount of target gas present. The reference electrode (RE) is mostly used for controlling redox reactions to reduce the working electrode's potential difference due to its deterioration. In some sensors these electrodes may not be used [81] [13].

Different sensors may use different types of selective membranes, electrolytes, and working electrodes to improve the sensor's selectivity for a specific type of ambient gas. This selection may or may not lower the sensitivity of the sensor. The electrolyte composition and sensing electrode material are selected based on the chemical reactivity of the target gas. For a sensor where high sensitivity is desired for low gas concentration, a hydrophobic membrane of coarse porosity and a less restricted capillary is used. This allows more gas to pass through. However, there will be a greater chance of water molecules from the

electrolyte escaping to the environment. In other words, the lifetime of sensors with high sensitivity is reduced compared to sensors with less sensitivity [81].

A filter can be used between the membrane and the ambient gases to increase the sensor's accuracy. This filter will limit contact with unwanted gases [23]. Most of these sensors require a small amount of oxygen and humidity to function properly. Electrochemical transducers are easily miniaturized, reaching the order of a few millimeters.

Catalytic sensor

The most widely used electrochemical sensor is the catalytic one. This sensor uses the enthalpy of combustion to create temperature variations that are measured resistively.

For a combustible gas to burn, it needs to reach a high temperature, called the ignition temperature. However, using chemicals means the gas may burn or ignite at lower temperatures. This is known as the catalytic phenomenon. Therefore, this type of gas sensor is made on this principle and is called a catalytic sensor. A gas molecule oxidizes on the catalyzed surface of the sensor at a temperature lower than the ignition temperature. All conductive materials change their conductivity as the temperature increases [13].

In short, the gas detection process in these sensors consists of the chemical reaction between the catalytic element and the combustible gases, leading to a temperature rise in the sensor, and causing a variation in its resistance. For that, a wire coil coated with the catalyst is electrically heated to a temperature that allows it to burn the gas being monitored [13]. This heating process activates the reaction between the gas and the catalytic element. The measurement of this internal resistance variation will dictate the presence of combustible gases in the environment [23].

This type of sensor typically consists of two elements, the sensing element, and the compensating element. The first contains catalytic material and is sensitive to combustible gases. The compensating element is inert. Thus the combustible gases will react only with the sensing element, leading to increased temperature and resistance. The compensating element will not react with the combustible gases, so temperature and resistance will remain unchanged [49]. A schematic representing the sensor's principle of operation is shown in figure 3.

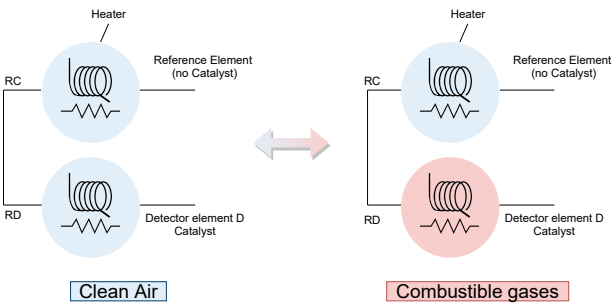


Figure 3: Catalytic sensor principle [49]

The resistance value of the detector element will be the only one to change in the presence of combustible gases. By measuring the output voltage, it is possible to evaluate the presence of the gases. Figure 4 shows a scheme for measuring the output voltage.

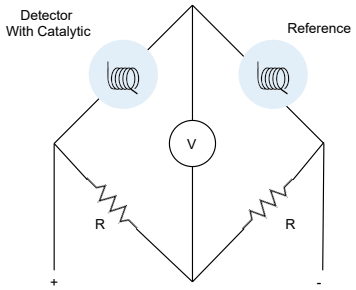


Figure 4: Catalytic sensor scheme [23]

Although these sensors present a low sensibility, not being easy task to precisely measure gas concentration, they present a long lifetime, which can reach up to 10 years [23].

2.2.3 Metal oxide semiconductors sensors

The use of metal oxide semiconductors as gas sensors for detecting poisonous, explosive, and polluting gases has received extensive research. Their compact size, great stability, low cost, and potential integration with silicon technology are all factors for their attractiveness [26].

Metal oxide semiconductor sensors detect gases by surface adsorption when the gas comes into contact with the sensor. They provide several advantages, such as low cost, high sensitivity, and easy manufacturing.

Most metal oxide semiconductors sensitive to gases are n-type. However, there are some p-type, such as NiO_x (usually doped with n-type semiconductors such as TiO_2), which could be used as gas sensing materials. The n-type has more electrons than protons, while the p-type has fewer electrons. The n-type sensor has an internal resistance decreased by reducing gases and increased by oxidizing gases. The p-type has the opposite operation. A significant difference between p-type and n-type is the sensitivity of the semiconductor depending on its temperature. The sensitivity increases with the temperature for the n-type, while for the p-type, it decreases. Therefore, p-type semiconductors have relatively lower operating temperatures than n-type semiconductors [23] [40].

Tin dioxide (SnO_2),cupric oxide (CuO), chromium oxide (Cr_2O_3), vanadium pentoxide (V_2O_5), tungsten trioxide (WO_3) and titanium dioxide (TiO_2) are some examples of metal oxides used for gas detection. They are used in sensors mainly based on changing resistance to target gases.

Tin dioxide (SnO_2) is a widely used material for gas detection. It is an n-type material whose electrical conductivity depends on the density of pre-adsorbed oxygen ions on its surface [40] [80]. Oxygen-related

gas detection involves the surface adsorption of oxygen on the surface of SnO_2. The surface then undergoes charge transfer during the reaction of chemisorbed oxygen with the target gas molecules, thereby changing the surface resistance of the sensor element. The charge carrier's surface is reduced by the electrons trapped by oxygen, which also creates a surface potential that might serve as a potential barrier to the flow of electrons. Current passes via the SnO_2 microcrystals conjunction regions (grain boundary) inside the sensor. The increase in resistance of the SnO_2 film is related to the adsorbed oxygen creating a potential barrier at the grain boundary that restricts the free flow of carriers. The adsorbed oxygen species react with any reducing gases that may be present in the air sample. This results in the release of free electrons back into the conduction band and a reduction in negatively charged oxygen surface density. As a result, both the grain boundary barrier height and the sensor resistance are reduced [11].

A simple example will be as follows. The atmosphere contains more oxygen than combustible gases. The oxygen particles attract the free electrons that are present in SnO_2 to the surface of it. No free electrons are available so the output current will be zero. When the sensor is placed in an environment with a particular gas, it will react with the absorbed oxygen particles and break the chemical bond between the oxygen and the free electrons. With the release of the free electrons, these are back to the initial position and can conduct current, which is proportional to the amount of free electrons available in SnO_2. The resistance of tin dioxide varies according to the variation of the gas concentration (this variation may be non-linear with the concentration) [40] [80] [11]. A possible example will be the resistivity value of tin dioxide which is typically around 50 kΩ in the air but can drop to around 3.5 kΩ in the presence of 1% methane [40].

Of all the metal oxide semiconductors, SnO_2 is among those with the highest sensitivity for the gases that are interested here (CO_2, H_2). The sensitivity of this sensor changes depending on the temperature, as explained above. This is due to the *reaction temperature* of $O-$. Increasing temperature increases the probability of adsorption of the gas molecule on the surface of the layer that would *consume* ions from the sensing materials, thus increasing the conductivity of this layer [40].

Figure 5 illustrate the typical constituents of a metal oxide gas sensor.

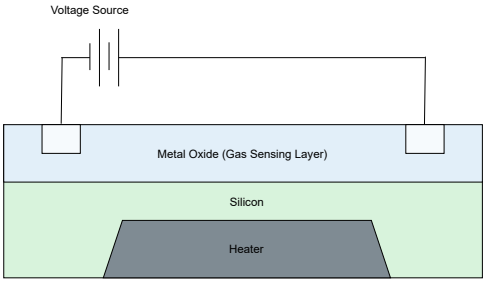


Figure 5: Typical metal oxide semiconductor gas sensor schematic [83]

The gas detection layer is the main component of the sensor that can be used to detect gas variation and generate a change in electrical resistance. As previously said, it can be SnO_2, CuO, Cr_2O_3 among

others.

The heating coil is intended to heat the sensing element to increase its sensitivity and efficiency. This constituent will need to have a high melting point, and it will need to remain heated without melting.

Typically these sensors have an electrode to transport efficiently the low current. Some sensors can have two electrodes. However, the most typical applications use just one [83] [80].

The working temperatures for SnO_2-based sensors are high, ranging from 250°C to 500°C. For each specific gas, there is an optimal temperature for detection. It is possible to distinguish between two different gases for a substantial temperature difference. For example, the optimum detection temperature of CH_4 is 400 °C, whereas that of CO is 90 °C. This requires thermostatically cycling the sensitive element so that the two gases can be detected by measuring the resistivity of the sensing element at each temperature value [40].

The most common method for increasing the sensor's sensitivity, using tin dioxide, is to increase the temperature. However, there are other methods, such as doping the surface of the sensing film with a suitable catalyst material or using sensor arrays [40].

In conclusion, these sensors, although widely used, have some disadvantages. High operating temperatures require large energy consumption and even more costs and complicated configurations compared to others that work at room temperature. These sensors also require a long recovery period after each gas exposure, which is not practical for some sensing devices and severely restricts their use in applications where gas concentrations can change rapidly [40] [23].

2.2.4 Polymers sensors

Some volatile organic compounds (VOCs) may cause adverse health effects, and below a threshold may not be detectable by metal oxide semiconductor sensors. Some of these VOCs can be inhaled by humans as they are constituents of many household products and industrial processes and usually are vaporized at room temperature. Some detection materials, such as polymers, are typically used for detecting VOCs or solvent vapors in the gas phase. They can also be used for detecting inorganic gases such as carbon dioxide [40].

When a polymer layer adsorbs a gas, if this layer has been exposed to the analyte vapor, it will see a change in its physical properties. These changes can be in its mass or dielectric properties. According to the changes in physical properties, polymers used for gas detection can be classified into conducting polymers, or non-conducting polymers [40].

The conductivity of polymers is low for them to function as gas-sensing materials. It has been found that their conductivity can be increased by doping processes and reducing reaction actions. After this doping process, which is reversible, the polymers become conductors or semiconductors. Thus, conducting

polymers can be seen as transducers, measuring the change in electrical properties. The electrical conductivity of conducting polymers is affected by exposure to various gases. Conductive polymers that can be used as gas sensing materials include poly-pyrrole (PPy), poly-aniline (PAni), poly-thiophene (PTh), and their derivatives [40] [5].

An example of a sensor using conductive polymers is the chemiresistor sensor. It consists of one or several pairs of electrodes and a layer of conducting polymer in contact with the electrodes. The electrical resistance of this sensor changes as gas is absorbed by the sensitive material. In comparison with electrochemical sensors, these do not require the electrolyte solution [77] [5].

Figure 6 represents the schematic of the conductive polymer sensor.

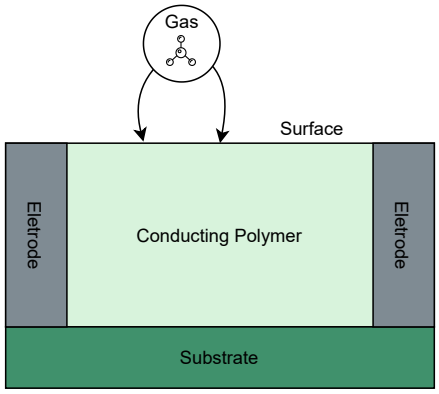


Figure 6: Polymer sensor schematic [77]

Non-conductive polymers have been used for coating the transducers. Polymers with different properties or physical adsorption mechanisms (physisorption) can be coated onto different sensing devices, being seen as a transducer altogether. The sensing devices could convert changes in the properties of the polymer being monitored into an electrical signal output. Although the principles of polymers in the presence of gases are easy to understand, their performance is not optimal [77].

These sensors have the advantage of operating at room temperature, high sensitivity, and short response times. They are sensors with low power consumption, cheap to produce, and with a simple and portable structure. However, they have some disadvantages, such as instability, low selectivity, and short lifetime [23] [40].

2.2.5 Optical sensors

Gas detection using optical methods is simple and can achieve higher sensitivity, selectivity, stability, and longer life than non-optical methods. The response time of these sensors is short, allowing real-time monitoring. Optical methods for gas detection are mainly based on spectroscopy. However, applications in gas sensors are few due to the high cost and difficult miniaturization process [40].

Specific gases absorb different wavelengths, each gas having a peculiar property of absorbing at different wavelengths. The *HITRAN* database provides the precise wavelength for each specific gas including CO_2 and H_2 [40].

Some techniques exploit optical characteristics as a way of measuring the concentration of gases in the environment. Others rely on adding materials that react with gases in the presence of light by reflecting, absorbing, or shifting the beam's wavelength into an optical fiber. There are several techniques for gas detection based on optical methods, such as tunable diode laser absorption spectroscopy (TDLAS), non-dispersive infrared, spectrophotometry, and photoacoustic spectroscopy, among others [23].

Infrared (IR) source gas sensors are based on optical detection and are widely used [23]. These sensors are based on basic absorption spectrometry, specifically on the principle of molecular absorption spectrometry.

Thus each gas has the property of absorbing infrared radiation with different wavelengths (infrared absorption fingerprint). So these sensors use the absorption of the radiation to determine which gas is present in the environment. Its concentration will be measured based on the attenuation level of the signal [8].

Commercial infrared sensors generally consist of an infrared source, a detector, a gas cell, the wavelength selection device, and some optical components such as lenses or mirrors to couple the radiation from the source through the gas cell to the detector [48] [42]. A schematic of a typical IR sensor is shown in figure 7.

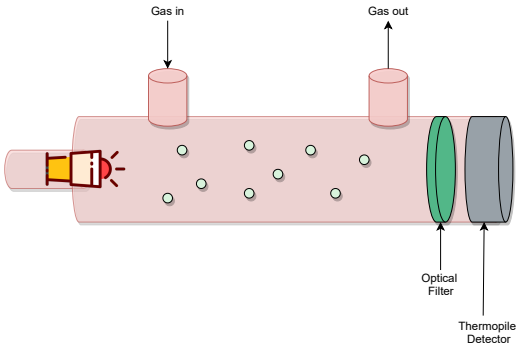


Figure 7: IR gas sensor schematic [42]

Frequently, the IR source emits broadband radiation, including the wavelength absorbed by the target gas. The target gas in the gas cell will absorb the radiation in its own peculiar way. The optical filter is adjusted based on the target gas and is used to filter out all radiation except the wavelength absorbed by the target gas. When there is no target gas, the photodetector receives the full wavelength because there is no absorption. If the gas is present, the signal is absorbed, and the photodetector does not receive the expected wavelength. The detected signal reduces as the concentration of the gas increases [48] [40] [50]. This system is also known as Non-Dispersive Infrared (NDIR) gas sensor.

NDIR optical sensors perform well and have been verified in areas such as air quality monitoring and gas leak detection systems. They are highly accurate and reliable [40].

2.2.6 Acoustic sensors

Acoustic sensors overcome some intrinsic weaknesses of sensors based on chemical principles, such as sensor lifetime. These sensors are divided into three main categories, sound velocity, attenuation, and acoustic impedance. This division is made according to the methods used. Sound propagates differently depending on the medium of propagation. In a gaseous environment, ultrasonic velocity is a function of temperature, pressure, humidity, and the properties of the gas mixture. With these characteristics, acoustic waves can be used to determine whether or not gas is present in the environment [23] [40].

The most widely studied and used technique is sound velocity. It is based on the difference in sonic velocity in different propagation media, using the propagation time over a fixed distance to determine the composition of a given gas mixture [40].

The following equation 1 represents the speed of sound concerning the medium in which it is presented. The variable c represents the speed of sound in meters per second, γ represents the specific heat ratio, R is the universal gas constant, T is the absolute temperature in kelvin, and MM is the molecular mass [23].

$$c = \sqrt{\frac{\gamma \cdot R \cdot T}{MM}} \tag{1}$$

This method is efficient when measuring concentrations in binary gases, i.e., it is effective when there is the possibility of two types of gases. When there are more gases, the system is susceptible to faults and may present false positives due to the presence of another gas in the proportions that would generate the same sound velocity in the propagation medium. The temperature also affects the detection, being necessary to proceed to a possible calibration of the sensor, considering the temperature and humidity of the system where the sensor will operate [40].

Figure 8 shows the schematic used in the acoustic gas-sensing approaches.

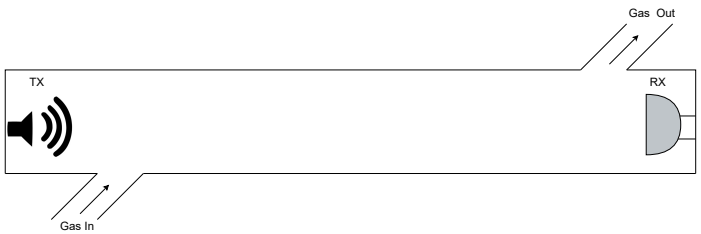


Figure 8: Acoustic gas sensing approach [23]

2.2.7 Calorimetic sensors

The most common methods for using calorimetric sensors are thermal conductivity. They use gas characteristics like thermal conductivity, creating temperature variations that will be measured resistively. These sensors generally have low selectivity, originating from their physical mechanisms. Many gases may have the same thermal conductivities. However, the use of these sensors is widespread, and with the necessary calibration of each sensor, it is possible to perform a safe and effective detection [40].

Thermal conductivity sensor

Thermally conductive sensors use the thermal conductivity of gases to detect the concentration of the gas in a given environment [40]. These sensors normally have two parallel tubes containing gas, and the sensing element is heated either by a metal wire or a thermistor to a specific temperature. One of the tubes contains the reference gas. The carrier gas flows into the other tube [54] [40]. Using this principle, the sensor detects changes in the gas's thermal conductivity by measuring the component's resistance in a column and comparing it to a reference flow. Helium or hydrogen is frequently used in the reference column. Most compounds have a lower thermal conductivity than these gases. Thus, the thermal conductivity is reduced whenever a different gas enters the column, and a detectable electrical signal is produced. The sensor is usually operated in the constant average temperature mode as part of a Wheatstone bridge circuit so that a temperature change in the sensor results in a signal. Helium has traditionally been the most commonly used gas, but the trend has been to change it due to increased scarcity [54].

The reference flow of circuit resistor 4 (figure 9) compensates for drift due to flow or temperature fluctuations. Changes in thermal conductivity due to different gases entering the tube will result in a temperature change, thus varying the value of resistor 3 (figure 9). This variation results in a signal that can be measured [40].

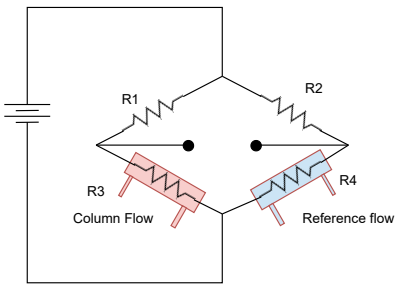


Figure 9: Thermal conductivity sensor scheme

The thermally conductive sensors have an extensive detection range, good stability, reliability, and simple equipment. However, their accuracy and sensitivity are the main disadvantages [40].

2.2.8 Sensors technologies comparison

After the short review on gas sensor technologies, table 1 summarizes the main characteristics of the sensors. Table 1 also present the advantages and disadvantages of each sensor described in this graph.

Despite the possibility of interference from other gases and environmental factors, such as temperature and humidity, sensors based on chemical reactions can increase their precision and accuracy with the employment of recalibration processes along with filters for other gases [23].

To improve the accuracy of some sensors, it is possible to employ two sensing technologies simultaneously, like acoustic and optic methods. In some cases, besides improving the gas measurement, this combination reduces the sensor's consumption, as two technologies that do not require the employment of heaters can be used [23].

Table 1: Table of sensing technologies [23]

Technology	Description	Strengths	Weaknesses
Electrochemical	In the presence of gas there is a chemical reaction between the gas and the sensing element, producing an electrical signal.	• Low Cost • Easily miniaturized • Filters can improve sensitivity • Long lifetime	• Low sensitivity and selectivity • Periodic calibration is necessary • High Response time • Interference from other gases and environmental factors
Metal Oxide Seminconductors (MOS)	It consists of a metal oxide semiconductor whose conductivity varies according to the presence of a certain gas. This results in a variation in the resistance value of the sensor. Temperature helps to increase the sensitivity of this sensor.	• Low Cost • Easily miniaturized • Filters can improve sensitivity • Short response time • Long liftime • Wide range of target gases	• Low sensitivity and selectivity • Need of heaters • Interference from other gases • Periodic calibration is necessary • High energy consumption
Polymers	They are divided into non-conductive and conductive. In conductive ones, the polymer reacts with the target gas, generating a variation of the output signal. The non-conductive transducers are coated with polymer to increase the detection efficiency.	• Low cost • High sensitivity • Short response time • Easily miniaturized • Low energy consumption	• Long-time instability • Long recovery times • Low lifetime • Periodical Calibration • Poor selectivity
Optic	It is based on the absorption spectra of each gas.	• High sensitivity and selecivity • Low response time • Long lifetime • Insensitive to environment change	• Difficulty in miniaturization • High cost
Acoustic	Through mathematical calculations it is possible to detect the target gas due to the speed at which sound propagates in the medium.	• Low response time • Long lifetime	• Reference gas needed • Difficulty in miniaturization • High cost • Low sensitivity
Thermal Conductivity	A reference gas is used to compare to a target gas. The presence of gas is detected by its conductivity. The gas enters a heated tube, leading to a change in conductivity which leads to a change in temperature, thus creating a detectable electrical signal.	• Low Cost • Long Lifetime • Easy fabrication • Low response time • Stable at ambient temperature	• Risk of catalytic poisoning and explosion • Intrinsic deficiencies in selectivity • Periodically calibration • Need of reference gases

2.3 Gas sensor applications

A gas sensor can be used for various applications as a means of monitoring the environment. They can be used to monitor an industrial process and/or assess air quality for safety at work. They can be used in homes to detect incidents such as fires, on oil platforms to control the gases released, in environments that need to control air quality (offices, car parks), and in mines for the concentration of harmful gases, among other applications.

Based on the technology used, each sensor can have a different application. Thus, based on the methods studied, the different applications of each sensor will be presented in table 2.

Table 2: Typical applications of sensors

Technology	Applications
Eletrochemical	• Industrial plant • Control of environmental emissions
Metal Oxide Semicondutor	• Industrial Applications and civil use • Environmental monitoring • Agriculture
Polymer	• Indoor Air Monitoring • Storage place of synthetic products such as paints, wax, or fuels • Chemical Industries
Optic	• Remote air quality monitoring • Gas leak detection systems with high • accuracy and safety
Acoustic	• Smart Metering of natural gas • Medical Applications • Early Warning Systems
Calorimetric	• Most combustible gases under industrial environment • Petrochemical plants • Mine tunnels • Kitchen

2.4 Challenges of gas sensors

Gas detection sensors use different methods and technologies to perform detection. Each method has advantages and disadvantages and there is no single most effective method or single most beneficial method. There is a best-suited method for each specific gas, so choosing a sensor is a trade-off [23] [40].

Ideally, one should be able to choose a sensor that can detect not just one type of gas but several. This feature is not always possible, as some methods are intended for a very limited set of gases. It would also be important for the sensor to be low-cost for numerous applications. Low-cost sensors tend to be less effective than the others, being unstable and often affected by atmospheric conditions [32]. A future challenge will be to lower the cost of these sensors using the most effective methods [40].

The energy consumption is another problem. Different technologies require high working temperatures to increase their sensitivity. This leads to high energy consumption.

The speed of detection for critical systems is important and one of the points to be improved for the different detection technologies. Sensors with fast detection could be used in several applications. Applications that require immediate detection would benefit the most from this feature.

Finally, durability would also be an essential aspect. This varies from sensor to sensor. Ideally, they should last as long as possible.

The aspects mentioned here present some of the challenges sensors go through, which are crucial in developing and evolving the technologies studied. In the future, it would be ideal if different technologies could implement some of these characteristics with an increasing evolution of sensors.

2.5 Market research

This section will present some sensors in the market that apply the technologies studied. Lastly, the BME688 sensor will be studied, and its technology will be verified.

2.5.1 Thermopile sensor EOC GDC TP 2C

The sensor EOC-GDC-TP-2C-CH_4 from Electro Optical Components is a MEMS thermopile element, dual channel TO-39, with a very high signal and high accuracy. This sensor uses the none dispersive infra-red (NDIR) technology, and the target gas for this sensor is methane CH_4. One of the applications of the sensor could be for industrial process control. Figure 10 represents an optical sensor EOC-GDC-TP-2C-CH_4.

Figure 10: Optical sensor EOC-GDC-TP-2C-CH_4 [67]

The sensor datasheet is available at [67]. The disadvantage of this sensor is that it can only detect methane, so it is necessary to obtain other sensors for other types of gases.

The performance specifications can be seen in table 3.

Table 3: Performance specifications of NDIR sensor [67]

Parameter	Typ	Unit	Conditions
Chip Size	1.8 x 1.8	mm^2	
Sensitive Area	1.35 x 1.35	mm^2	
Thermopile Resistance	43 ±10 %	$k\Omega$	Temp = 25 °C
Noise Voltage	27 ± 2	$nV/Hz^{1/2}$	Temp = 25 °C
NEP	0.62	$nW/Hz^{1/2}$	500K, 1 Hz
Responsivity	43	V/W	500K, $5.5\mu m$(Long pass)
Temp Coefficient of Resistance	0.1	%/°C	Temp = $25 \sim 50$°C
Time Constant	32	ms	
Specific Detectivity	$6.8e+07$	$cmHz^{1/2}/W$	500K, 1 Hz
Thermistor Resistance	$100 \pm 3\%$	$k\Omega$	Temp = 25 °C
Thermistor BETA-Value	$3950 \pm 1\%$	K	25 ° C/ 50°C

2.5.2 Figaro TGS 6812-D00 sensor

The TGS 6812-D00 sensor from the manufacturer "Figaro" is a sensor that uses catalytic technology. It is specifically designed to detect hydrogen, methane, and liquefied petroleum gas. This sensor features good durability, fast response, linearity, and high accuracy [76].

This sensor can detect both hydrogen and methane and can monitor gas leaks from stationary fuel cell systems that convert fuel gases into hydrogen. This sensor has an adsorbent inside its cap, which makes it more durable against silicone compounds in harsh environments [76]. It is represented in figure 11.

Figure 11: TGS 6812-D00 sensor [76]

The datasheet [76] suggests using a "Wheatstone bridge" for signal detection in the presence of gas. Combustible gases react with the catalytic solution by increasing temperature and thus increasing resistance. The resistance value will vary, and there will be an out-of-balance in the bridge.

The characteristics of the sensor are shown in the table 4.

Table 4: TGS 6812-D00 sensor specifications [76]

Model Number		TGS 6812-D00	
Sensing Element type		Catalytic	
Target gases		Hydrogen, methane, iso-butane	
Typical detection range		$0 \sim 100$ % LEL of each gas	
Standard Circuit Conditions	**Operating Voltage**	3.0 ± 0.1 V AC/DC	
Electrical Characteristics under standard test conditions	Heater Current	175 mA (typical)	
	Heater Power consumption	525 mW (typical)	
	Zero Offset	$-15 \sim +55$ mV	
	Output sensitivity ($\Delta Vout$)	hydrogen	$8 \sim 16mV$ in 4000ppm
		methane	$10 \sim 18mV$ in 5000ppm
		iso-butane	$5 \sim 11mV$ in 1800ppm
Standard Test Conditions	Test Gas Conditions	Hydrogen/methane/iso-butane in air at 20 ± 2°, 65 ± 5 % RH	
	Circuit Conditions	3.0 ± 0.05V AC/DC	
	Conditioning period before test	30 sec	
Operating conditions		-10°C$\sim +70$°C, $\leqslant 95$ %RH (w/o dew condensation)	
Storage conditions		-10°C$\sim +80$°C, $\leqslant 95$ %RH (w/o dew condensation)	

2.5.3 MQ sensor

There is a family of sensors designed to detect gases, which are given different names based on the target gases they contain. This family is known as the MQ-series from the manufacturer "Hanwei eletronics". The MQ-2 sensor will be presented, a sensor designed to detect methane, butane, and also smoke [44][71]. Figure 12 represents this sensor.

Figure 12: MQ-2 sensor [44]

The sensor datasheet also provides information about the sensor's materials, such as the electrodes, which are made of gold. The electrode line is made of platinum, and the aluminium oxide ceramic tube, among other components that compose the sensor. The necessary preheating time is 48 hours, and the sensor consumes about 800 mW of energy during this process. The operating current is about 160 mA [44][71].

2.5.4 NE4-CO sensor

The NE4-CO sensor from the manufacturer "Nemoto Sensor Engineering Company" is an electrochemical sensor. This sensor features three electrodes for the detection of carbon monoxide. It is a compact sensor with high reproducibility and linearity. It is also a very specific sensor, i.e., it only detects the presence of a particular gas, in this case, carbon monoxide [72] [60]. Electrochemical sensors would have to change the electrolyte liquid to detect different gases. Due to this fact, these sensors become very specific. The energy consumption of this sensor is reduced, as no heating processes are required. This sensor is represented in figure 13.

Figure 13: NE4-CO sensor [72]

A table with some characteristics is shown in the table 5.

Table 5: NE4-CO Sensor characteristics [72]

Items	Characteristics
Detected gas conc.	$CO\ 0 \sim 1000ppm$
Output Current	$70 \pm 15nA/ppm$
Repeatibility	Less than$\sim \pm 2$ %
Response Time	T90: less than 30 sec.
Zero Offset drift	Less Than $10ppm(-20 \sim 50°C)$
Temperature	$-20°C\sim +50°C$
Humidity	$15 \sim 90\%$ RH

2.5.5 VQ5 sensor

The VQ-5 sensor family from the manufacturer "SGX Sensortech" is based on thermal conductivity detection technology. The datasheet [62] of this sensor recommends using a Wheatstone bridge to monitor variations in the presence of target gases. The whetstone bridge is presented in figure 14. When a gas whose conductivity is higher than the reference gas (normal air in this sensor) is present, more heat is lost in this element than in the reference gas. This heat loss leads to a reduction of the resistance value, unbalancing the bridge. Thus, the higher the conductivity, the higher the heat loss, which lowers the resistance value.

This sensor can be operated up to the maximum recommended temperature, which is typically 500°C. The recommended bridge supply is around 2 volts, and the estimated maximum consumption is 0.48W.

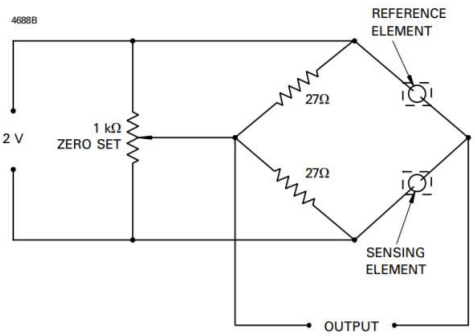


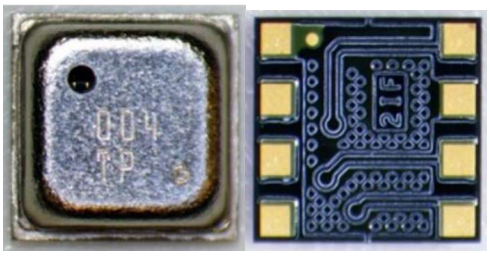
Figure 14: VQ5 Wheatstone bridge [62]

Figure 15 is a figure of a VQ5 sensor of the thermal conductivity type. This sensor is the VQ5MB, which means that it is a PCB-legged sensor with a sealed compensator can. The different variations of this sensor (VQ5, VQ5B, VQ5M, VQ5MB) can be found in its datasheet [62].

Figure 15: VQ5MB sensor [62]

2.5.6 BME688 sensor

The BME688, advertised as a 4-in-1 sensor, can measure temperature, pressure, humidity and gas concentration. It is housed in a small and compact package, 3.0 x 3.0 x 0.93mm, producing a package volume of 8.4mm. This sensor uses as digital interface (SPI and I2C) [6]. The packaging is 8-pin with a metal lid, and an opening in the top left-hand corner allowing this sensor to be exposed to the environment [35].



(a) BME688 Top (b) BME688 Bottom

Figure 16: BME688 Top and bottom package figures [35]

The upper and lower package photos are shown in figure 16. It is possible to see the opening mentioned in the pictures. The bottom surface has eight contact pads and metal traces for the package surface interconnection.

Photographs were taken of the inside of the sensor after removing its protective cover. That is shown in figure 17. It is possible to see the pressure, temperature, and humidity sensors embedded in a substrate on the left. On the right is the sensor for gas detection. The SoC or ASIC controller is under the left die [35].

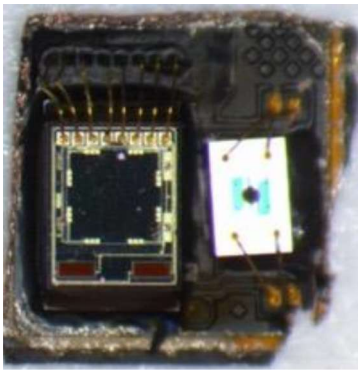


Figure 17: BME688 Without metal lid [35]

Figures were collected from the integrated MEMS mould [35]. The pressure, temperature, and humidity sensor are referenced in figure 18, and the location of each can be seen.

The pressure sensor will be a piezoelectric-based one [35]. In a piezoelectric sensor, an electrical charge is induced in the piezoelectric material when stress is applied. The ambient pressure makes a mechanical deflection which is measured by an electrical resistivity change [35] [30].

The temperature sensor could be a semiconductor diode [35]. The humidity sensor appears to be composed of two capacitor plates, which are probably a porous polymer-based dielectric [35].

In the figure, it is also possible to see the eight-wire connection that connects the integrated MEMS to the metal traces of the packaging substrate [35].

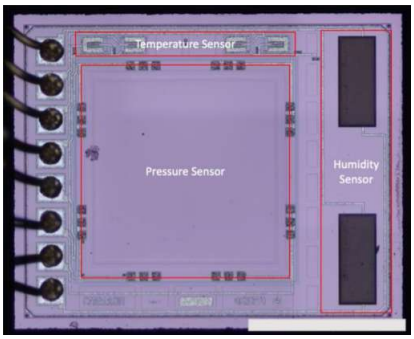


Figure 18: BME688 integrated MEMS [35]

Finally, the gas sensor remains to be discussed. As said before, this is next to the integrated MEMS and ASIC. There is also the figure 19 of this sensor. The gas-sensitive layer is located in the middle of the die (blue circle in the figure 19) above two electrodes [35]. Based on these figures and based on the datasheet provided, it is possible to conclude that this sensor is a metal oxide semiconductor sensor [35] [6]. This operating principle has also been studied. It is also possible to see in the figure the metallic oxide element. The most used in industry is tin dioxide. Based on the figure's visualization, evaluating the sensor's price, and checking its datasheet, it is a strong possibility that this sensor uses this element [35]. Thus, the sensor is a MOS and uses tin dioxide as the metal oxide.

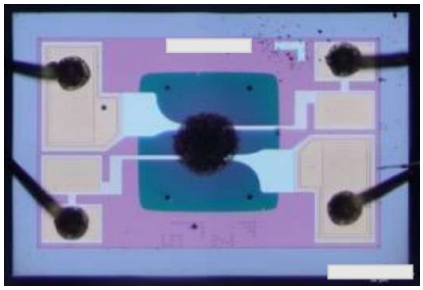


Figure 19: BME688 gas sensor [35]

A figure of the ASIC is presented in figure 20. A large logic block can be seen, represented in the image with its measurements, surrounded by analog and memory blocks [35].

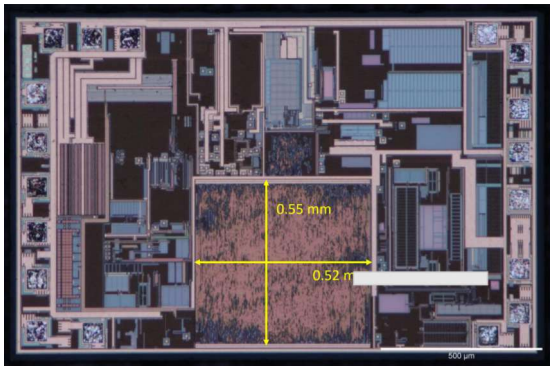


Figure 20: BME688 ASIC [35]

Since there is not enough information about the nature of the methods and technologies used for the different sensors composing the BME688, these practical studies were developed to obtain some conclusions about it. The datasheet, despite identifying the technology of some of them, is not very clear about it.

With these studies, it is possible to know how the sensor is developed internally and to draw conclusions about the sensors used. It is also important to establish a relationship between this sensor and the older sensor, the BME680. Studies have been developed comparing the BME680 with its successor, the BME688.

The size compared between this sensor and the previous one, including the circuits such as ASIC, MEMS integrated circuit, gas sensor, and logic ASIC circuit has been published [35]. The size comparision is summarized in table 6.

Table 6: BME688 and BME680 comparison [35]

Areas	BME688	BME680
ASIC DIE (mm^2)	1.62	1.62
ASIC Logic Circuit (mm^2)	0.32	0.32
Integrated MEMS (mm^2)	1.25	1.25
Gas Sensor (mm^2)	0.54	0.54

With this, it is concluded that the circuits all have the same measurement in both devices. However, based only on these measurements, it is not possible to conclude that the sensors are equal. It will be necessary to look at other blocks present in the ASIC and make a more extensive comparison. However, in physical terms, the two sensors are very similar [35].

The BME688 gas scanner detects gases, for example, volatile sulfur compounds, gas emissions from different surfaces, quantities of organic compounds due to food, exhaled breath, etc. It can be used in various applications [6].

With the external BME AI-studio tool, users can train the BME688 gas scanner for different applications and environments to be able to detect target gases. By placing the sensor in an isolated environment with the desired amount of a particular gas, it is possible to train this sensor so that it later detects the presence of this gas [6].

Electrical characteristics of this sensor can be seen in table 7.

Table 7: BME688 Electrical characteristics [6]

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage Internal Domains	V_{DD}	Ripple max. 50 mVpp	1.71	1.8	3.6	V
Supply Voltage I/O Domain	V_{DDIO}		1.2	1.6	3.6	V
Sleep Current	I_{DDSL}			0.15	1	μA
Standby current (inactive period of normal mode)	I_{DDSB}			0.29	0.8	μA
Current during humidity measurement	I_{DDH}	Max. Value at 85 °C		340	450	μA
Current during pressure measurement	I_{DDP}	Max. Value at -40 °C		714	849	μA
Current during temperature measurement	I_{DDT}	Max. Value at 85 °C		350		μA
Start-up Time	$t_{startup}$	Time to first communication after both V_{DD}> 1.58 V and V_{DDIO} < 0.65 v			2	ms
Power Supply rejection ratio (DC)	$PSRR$	full V_{DD}range			$\pm 0.01 \pm 5$	$\%r.H./V$ Pa/V
Standby Time accuracy	$\Delta t_{standby}$			± 5	± 25	$\%$

After the power-up sequence, the sensor automatically goes into sleep mode. This mode is the mode with the lowest power consumption. It presents a consumption between $0.15\mu A$ to a maximum of $1\mu A$.

2.6 Artificial intelligence

This chapter will present different artificial intelligence algorithms: decision trees, support vector machines, and neural networks. The idea will be to use low-power and low-cost sensors to make gas detection as efficient as possible. One way to make a more efficient detection comes with the use of artificial intelligence techniques [9]. These techniques may be able to overcome the weaknesses of the sensor (e.g. long time constant, hysteresis etc).

Artificial intelligence systems can be classified according to the type of supervision they receive during training. There are two main categories, unsupervised and supervised learning [24].

In supervised learning, the datasets are labeled, meaning that the training data which is fed into the algorithm includes the desired solution (label) [24].

In unsupervised learning, the algorithms used analyze unlabeled data sets. These algorithms discover hidden patterns in data without human intervention. Some examples of unsupervised learning algorithms are k-means, kernel PCA, and ECLAT [24].

There are two typical supervised learning tasks, classification and regression [24]. In the classification task, an algorithm is used to assign test data to specific categories. The regression task uses an algorithm to predict a numeric value given a set of features called predictors. Some examples of supervised learning algorithms are linear regression, support vector machines (SVMs), decision trees, random forests, and neural networks [24].

2.6.1 Decision tree

The decision tree is a supervised learning algorithm that can perform both classification and regression tasks.

The decision tree is structured as a tree, with the root node representing the entire population and samples, this node has a depth of 0. The following branches represent the decision rules, and each leaf sub-node represents the results. If the condition evaluated in each node is true, the next node to be analyzed should be the left one. If the condition is false, the next branch will be the right one. The tree is traversed until no children nodes are found.

An example of a decision tree is represented on figure 21.

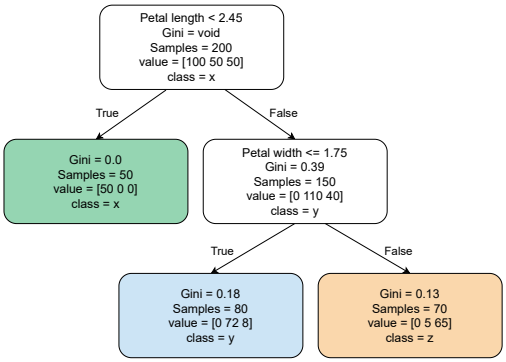


Figure 21: Decision tree example [24]

Each node represents some attributes. The samples attribute count how many training instances apply to that node. The value attribute represents how many instances of each class the node applies to, for example, for a value = [x,y], it is possible to say that there are x instances that belong to a class and y instances that belong to another class. The Gini attribute measures the impurity of each node. If a node represents a Gini of zero, it means that this node is "pure" and all training instances belong to the same class. The equation 2 represents the calculation of the impurity of each node [24].

$$G_i = 1 - \sum_{k=1}^{n} P_{i,k}^2 \tag{2}$$

Where,

- $P_{i,k}$ is the ratio of class K instances among the training instances in the i^{th} node

Using the example in figure 21 it is possible to clarify some of the concepts mentioned above. This example presents a classification for Iris database.

For the root node, it can be said that there are 200 samples, this being the total number of samples in this classifier. The root node condition is if the petal length is less than 2.45. Assuming that the iris being classified has a petal length of less than 2.45, the condition is true, and so the next node is on the left, being a leaf node. In the example, 50 samples belong to this condition. All these samples are of class x (iris types). This node has a Gini of 0. Then the iris being classified will be considered of type x. If the iris to be classified does not have a petal length of less than 2.45, the next node will be the one on the right. This node is considered a decision node. In this node, since it is not a leaf node, a question is asked: "is petal width less than 1.75?". If the iris to be classified has a petal width of less than 1.75, the next node will be the one on the left. If it is greater than 1.75, the next will be the right one. In the left node, the iris will be assigned as class y. In the right, it will be class z. Focusing on the left node, it is possible to conclude that for that condition, there are 72 samples of class y and 8 samples of class z. Therefore, the Gini cannot be 0. The value of the Gini will be equal to:

$$Gini = 1 - (\frac{72}{80})^2 + (\frac{8}{80})^2 \quad (3)$$

It is also possible to conclude that the decision tree has two features and a depth of 2.

2.6.2 Support-vector machines (SVM)

SVM is an artificial intelligence model that performs linear and non-linear classification and regression. Linear SVM comprises hard and soft margin classification. The hard margin classification is used when the dataset used is linearly separable, being more sensitive to outliers. If the dataset is separable, a *street* exists between the two classes, which is as wide as possible, with no instances inside the street. Figure 22 presents an example of a hard margin classifier. On the right it is possible to see the decision boundary of an SVM. The line separates the two classes and stays as far away from the closest training instances as possible.

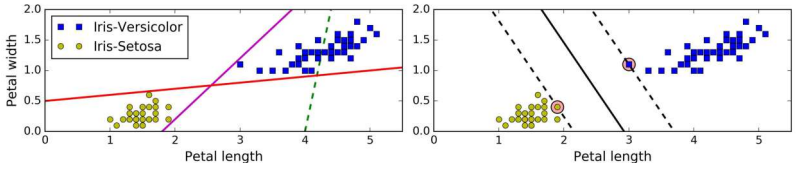


Figure 22: Hard Margin classification linear SVM [24]

The soft margin classification is a more flexible model, and there may be instances within the street. The goal is to try to keep the street as wide as possible without too many violations [24]. To control that can be used the C hyperparameter. A small value of C leads to a wider street but more margin violations. A high value leads to a small street and fewer violations. Figure 23 presents a soft margin classification and the difference between the C hyperparameter. The samples that are within the margin boundaries are called the support vectors [1].

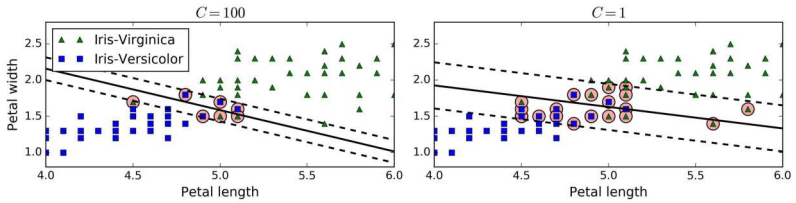


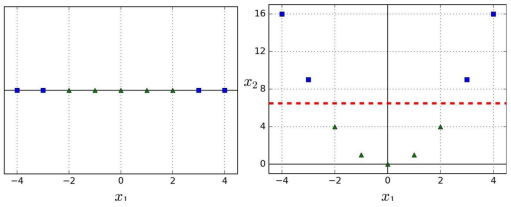
Figure 23: Soft margin classification linear SVM [24]

One technique used when datasets are not linearly separable is to add new features, and in some cases, these result in a linearly separable dataset.

An example is shown in figure 24. Figure 24a shows a simple dataset with only one feature x_1. As can be seen, the dataset is not linearly separable. However, if the polynomial feature from equation 4 is

added, the result is a two-dimensional linearly separable dataset. Figure 24b shows the dataset with the added feature that is now linearly separable.

$$x_2 = (x_1)^2 \tag{4}$$



(a) (b)

Figure 24: Non-linear dataset into a linear dataset adding features

Thus, by adding polynomial features, it is possible to make the datasets separable, and for very complex datasets, a higher degree polynomial is used. This also results in a slower model. Using a lower degree polynomial may be not effective for complex datasets [24].

Another option for a separable dataset are kernel functions. When datasets are not separable, SVMs provide the flexibility to map the input data to a higher dimensional feature space through the use of kernel functions [52].

Kernel functions gives the same result as if polynomial features were added without actually adding them. Thus, there is no combinatorial burst of features since they are not added [24].

In conclusion, a kernel function projects data from a low-dimensional space to a higher dimension space [17] [52] [22]. Although using these functions, there is no need to explicitly map the data to the higher dimensional spaces, which is typically an expensive computational task.

$$\langle x_1 | x_2 \rangle \leftarrow K(x_1, x_2) = \langle \Phi(x_1) | \Phi(x_2) \rangle \tag{5}$$

where, K is the kernel function and $\Phi(x)$ is a non-linear mapping function. K is the inner product of the feature space based on a mapping Φ [22].

The most commonly used kernel functions are [1]:

- Polynomial : $k(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + r)^d$ where d is specified by parameter degree, r by (coef0), γ is parameter C
- Radial Basis Function : $k(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2)$ where γ is a parameter that must be greater than zero

The polynomial kernel function is directional due to the dot product in the kernel. The output depends on the direction of the two vectors in low-dimensional space [22]. In figure 25 is possible to see an SVM

classifier using a polynomial kernel. The degree of the polynomial is represented by d, and the r variable is the zero order coefficient (coef0). This hyperparameter controls how much the model is influenced by high-degree polynomials versus low-degree polynomials [24].

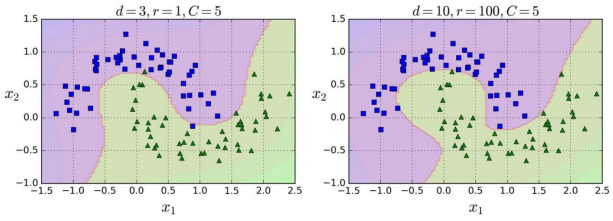
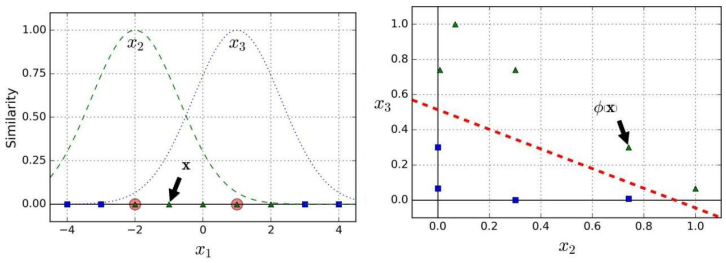


Figure 25: *SVM classification with polynomial kernel* [24]

In SVMs, radial functions define the radial Gaussian basis function (RBF). This is a similarity function that measures how much each instance resembles a particular landmark. An example is given below, where gamma equals 0.3, and two landmarks, x1=-2 and x1=1, are selected in figure 26a. Using the RBF kernel function, new features are calculated. As an example, is used the instance x1 = -1. It is at a distance from the first landmark of 1 and 2 from the second landmark. Thus, the new features will be $exp(-0.3 * 1^2) \approx 0.74$ and $exp(-0.3 * 2^2) \approx 0.30$. Figure 26b presents the transformed dataset [24] [22].



(a) (b)

Figure 26: Similarity function using RBF kernel function

In this function, the γ parameter influences the model by acting as a regularizer. If the model is overfitting, it will be necessary to reduce γ. If it is underfitting, it will be necessary to increase it. A high γ makes a tighter curve and the decision boundary more irregular. A γ makes the curve wider and the decision boundary smoother. Figure 27 shows an example of how this parameter varies [24].

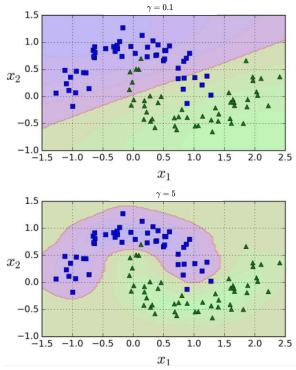


Figure 27: SVM classification with RBF kernel [24]

The selection of each kernel function is not fixed and depends on each application [22].

2.6.3 Neural networks (NN)

A neuronal network (NN) is a set of neurons connected to various points by the network. It consists of an input layer, one or more hidden layers, and an output layer. Figure 28 represents a typical feed-forward neural network. When a regression is performed, the output layer predicts a numeric value. When a classification is performed, the predicted value will be between 0 and 1, with zero being one class and 1 another. The predicted value will always be a probability. When there are more than two classes, there will be more output neurons, with each neuron representing each class and the output value representing the probability assigned by the network [25].

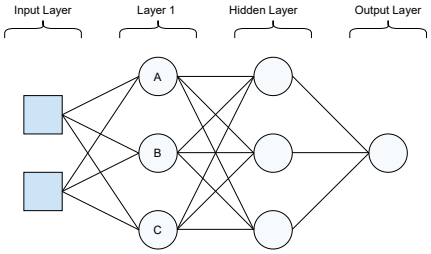


Figure 28: Neural networks example [25]

The neural network represented has an input layer, a hidden layer, and an output layer. The input layer represents the input data to the neural net. Hidden layers consist of the neurons that are between the input layer and the output layer. The output layer is one neuron or multiple neurons that determines the output generated by the network [25] [63].

In figure 29, a portion of figure 28 is represented. In other words, neuron A is represented. It is a multiple-input neuron structure. A neuron typically has multiple inputs. Each input has an assigned weight W, as seen in the figure. It should be noted that W and b are adjustable parameters for each neuronal

network. The designer chooses the activation function, or transfer function, and the parameters W and b are adjustable by the training of the neural network [25].

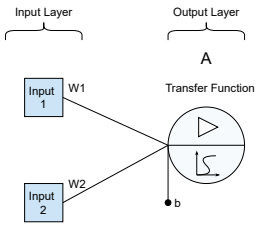


Figure 29: Neural network, neuron A linear activation vs non linear

The transfer function can be linear or non-linear. A particular transfer function is chosen to best fits the problem the neuron is trying to solve [25] [63]. Taking the example in figure 29, it is possible to say that the output of the network will be computed in two steps. First:

$$Z = Input1 * W_1 + Input2 * W_2 + b \tag{6}$$

If the activation function is linear, the output is just proportional to (Z). Otherwise a common non-linear activation function is the sigmoid.

$$\phi(z) = \frac{1}{1 + exp(-z)} \tag{7}$$

Non-linear activation functions are used since linear ones may not be flexible enough. Another characteristic is that the non-linear activation function limits the output value in a range between zero and one as shown in equation 8 and 9 respectively.

$$\lim_{z \to -\infty} \phi(z) = 0 \tag{8}$$

$$\lim_{z \to +\infty} \phi(z) = 1 \tag{9}$$

The sigmoid function is mainly used in multilayer networks. One reason the function is used is that it is differentiable [25] [63].

2.6.4 Artificial intelligence algorithms Comparision

Table 8 represents the algorithms discussed above. The characteristics of these algorithms are listed as well as some possible applications that these can have. It is also important to note that these algorithms were presented according to their complexity. The decision tree is presented as the least complex algorithm (i.e. less powerfull), and the neural network is the most complex algorithm.

Table 8: Artificial intelligence models comparision [24]

artificial intelligence	Description	Applications
Decision Tree	It is a supervised Learning, for classification and regression. It is structured as a tree, with the rote node representing the entire population. The branches represent the decision rules and each leaf sub-node represents the result. The tree is traversed until no children nodes are found.	• Predicting occupancy in places: • hotels, stadiums, restaurants • Using demographic data to • find prospective clients • Prospective growth opportunities • for businesses based • on historical data
SVM	Is an artificial intelligence model that performs linear and non-linear classification and regression. Linear SVM is composed of hard-margin classification and soft-margin classification. Is used when the dataset is separable. One algorithm is more flexible and may present some outliers. In non-linear SVM the dataset is not separable, so, it is added new features to make the dataset linearly separable. It could also be used the kernel trick.	• Facial Expression Classification • Texture Classification • Text Classification • Speech Recognition
NN	NN is mostly used in supervised learning and can perform classification and regression. Consists of an input layer, one or more hidden layers, and an output layer. The most basic type is the feed-forward neural network. This can be linear or non-linear due to the activation function. The output of the neural network in the linear activation function is the sum of the scaled inputs. In the non-linear, the sum of inputs multiplied by the weights passes the sigmoid function.	• Facial Recognition • Stock Market Prediction • Healthcare • Signature Verification • Weather Forecasting

2.7 Hardware design

This chapter will introduce important concepts for hardware design. It will first demonstrate the process adopted for hardware development and then present aspects that should be considered when developing a printed circuit board (PCB).

2.7.1 Design process

The steps for hardware design can be abstracted into the Requirements Phase, Schematic Phase, Layout Phase, Hardware Verification, Validation, and Production phases [47] [15]. Figure 30 shows the hardware design process adopted.

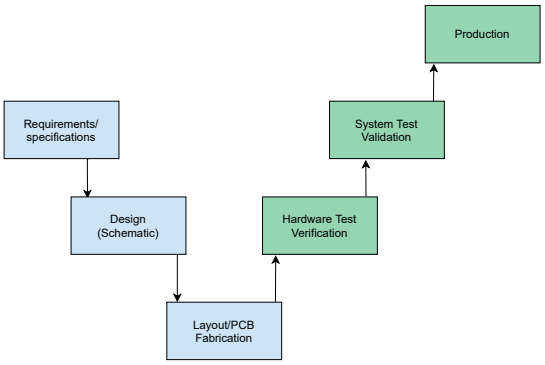


Figure 30: Hardware design steps [47]

The first phase is the development of the specifications and requirements for the board to be developed. There is a set of survey questions, such as what it is intended to develop and the requirements for that. With this, the essential components for the realization of the product are selected. The block diagram is developed to summarize the components and their interaction. The schematic phase is initiated with the

components selected and the problem well defined. In this phase, simulations are developed for circuit validation in addition to circuit development. The layout of the schematic is performed and validated. The next phase is a test of all the hardware developed and the validation of the system for future production.

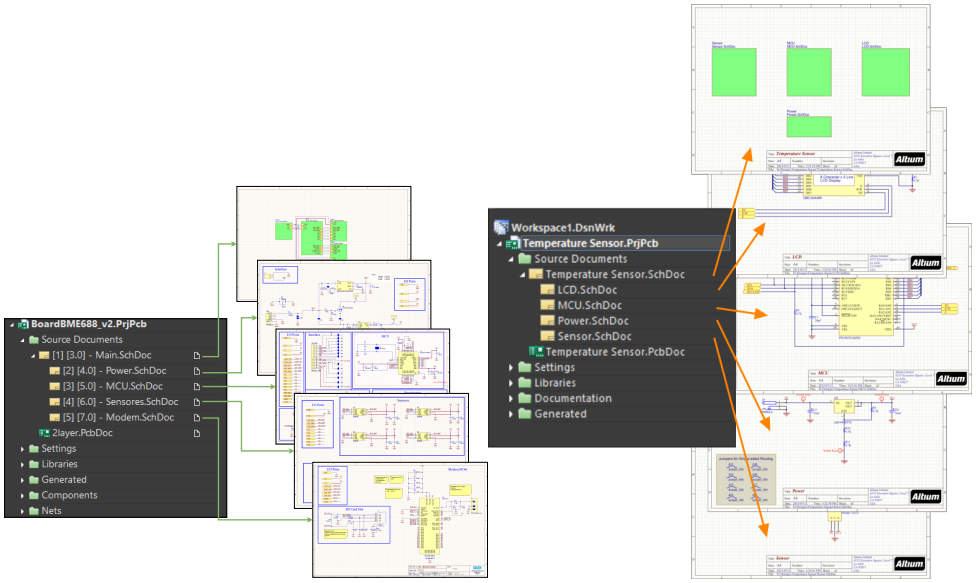
2.7.2 Schematic design

The user creates the schematic by placing the symbols of the components and connecting them via their pins. The symbols are created by the user or can be available in libraries. These can be provided by the tool used or by the producer of each component. The circuits can be designed based on the guidelines that each component has [39].

Hierarchical vs flat design

A project can be displayed on a single sheet, large enough to cover the entire schematic, or on multiple sheets. A multiple sheets project has advantages because it is more easily displayed and understood. A multiple-sheet design can have several modular elements. Keeping these modules as individual documents allows several designers to work on a project simultaneously. This division improves the readability of the design, which is important for those who need to read and interpret the schematic later in the product's life. Thus, there are two approaches to the structure of a design, flat and hierarchical [39] [45].

A flat design is equivalent to a single-sheet design, which is separated into multiple sheets. Different parts of the schematic sit on each of these sheets. This structure uses horizontal connectivity, that is, connectivity created directly from any sheet to any other sheet. There can be a top sheet in a flat design, which will have a sheet symbol for each of the sheets in the project. These sheets in the top sheet cannot have any connectivity between them. A hierarchical design has sheet symbols to create parent-child relationships between them. The connectivity of these is through sheet entries in these sheet symbols. In a flat design, the child sheet is identified by setting the name of the sheet symbol. In a hierarchical design, a child sheet may include a sheet symbol, where a lower-level sheet is referenced [39] [45].



(a) Hierarchical design (b) Flat design [45]

Figure 31: Hierarchical and flat design

Electrostatic discharge ESD considerations

Although ESD, EMC & EMI concerns are not always required and are not a specific part of the schematic phase design, if needed, they should be introduced as soon as possible into the design phase. It seemed appropriate to include them here.

Electrostatic discharge is the discharge behavior of static electricity, for example between the body and the surface of a device. If the device is not properly protected it can lead to system failure, which can be reversible or even irreversible, compromising the entire system [37] [27] [82].

Several components are subject to electrostatic discharge, and it is important to protect them. It is therefore beneficial to understand the basics of ESD, the types of protection used and how they work, and the type of tests that can be developed for ESD testing [37].

There are two types of ESD. At the device level and the system level. The first is relative to the Integrated Circuit IC and the type of discharges possible for this. The second one is relative to the whole system, like a PCB. Damage to an IC can occur anytime, from assembly to board-level soldering to end-user interactions. All ICs have built-in device-level ESD structures that protect the IC against ESD events during the manufacturing phase [82] [27].

These events are simulated by three different device-level models, the human body model (HBM), the machine model (MM), and the charged-device model (CDM). These models are used for testing the manufacturing environment, where final assembly testing and board-level soldering are performed in controlled ESD [27] [82]. Each model's definition is presented in table 9.

ICs are usually specified to survive ESD strikes only to a $2kV$ HMB [27]. Device level models are sufficient for ESD testing for devices. However, these modes are not suitable for performing system-level

tests. That reason is that the ESD strikes from both voltages and currents can be much greater in the end-user environment. For system level ESD testing, there is another model, IEC 61000-4-2. In short, device-level testing is intended to ensure only that ICs survive the manufacturing process. System level testing is intended to simulate end user ESD events in the real world [27] [82].

Table 9: ESD models [27]

	Device Tests			**System Test**	
	Human-Body Model (HBM)	**Machine Model (MM)**	**Charged-Device Model (CDM)**	**IEC 61000-4-2 Model**	
				Contact Discharge	Air-Gap Discharge
Definitions	Human Body Discharging accumulated static	Robotic arm discharging accumulated static	Charged device being grounded	Real World ESD events	
				Eletrode test simulator held in contact with the device under test (DUT)	The eletrode spark to the DUT actuate the discharge
Test Levels	500V to 2kV	100V to 200V	250V to 2kV	2kV to 15kV	
Pulse Width (ns)	150	8	1	150	
Peak Current at Applied 2kV A_{pk}	1.33A	-	5A	7.5A	
Rise Time	25 ns	-	400 ps	1ns	
Number of Voltage Strikes	2	2	2	20	

To protect the system against electrostatic discharge usually are used discrete stand-alone transient-voltage-suppressant diodes or transient-voltage suppressors (TVS) [27]. There are two types of TVSs, bidirectional and unidirectional. The TVSs are designed to be an open circuit during normal operation and a short to ground during an ESD event. In bidirectional, when an ESD strike, positive or negative, hits the board (for example, an I/O), one diode becomes forward-biased, and the other enters its breakdown region, creating a path in which ESD energy is immediately dumped to ground.

In unidirectional, when an ESD strike (positive) hits the board, the diode enters its breakdown region, and the ESD energy is dissipated through the path created to ground.

In conclusion, circuits are becoming smaller and smaller and more susceptible to destruction due to electrostatic discharges. Protection at the device level is not sufficient to protect the device when it is at the system level. For this reason, protections must be used to protect the system as TVS. It is also necessary to consider some parameters when choosing the TVS like the breakdown voltage [27].

EMI / EMC design

Nowadays, wireless communication has become almost ubiquitous, being used in all kinds of devices such as mobile phones, radios, and wireless charging (mobile phones, cars), among others. These devices intentionally transmit electromagnetic waves. However, all these signals will have to coexist, making electromagnetic interference (EMI) and electromagnetic compatibility (EMC) important considerations in device manufacturing. The problem does not reflect only wireless communication devices but covers all devices [3] [29].

EMI problems are caused by the changing current of conductors within a component, known as di/dt noise. The change of current causes electromagnetic emissions. External electromagnetic energy can induce noise in circuits causing false switching logic and leading to incorrect operation of devices. EMI problems are mostly caused by fast switching processes [3].

EMI can be either conducted or radiated. Conducted EMI noise comes from a component in the system. The noise propagates through current, direct conduction, or capacitive or inductive coupling. Radiated EMI noise is emitted and received radiatively. Therefore radiated EMI is a form of crosstalk. It is usually caused when the current paths in the PCB cover an area large enough to form antennas from which the signals are radiated [29]. In several places, bandwith reduction filters may mitigate EMI.

EMC has two aspects. The first is based on the fact that the device does not transmit interference in operation. The second is based on protecting the device from possible interference by becoming an EMI victim. EMC is achieved by first considering a careful layout, then adding components in the circuit to act as filters to eliminate unwanted signals. These components include ferrite beads, capacitors, line inductors, or others. In the layout phase, one of the precautions to be taken will be to make a correct shield and protect sensitive analog tracks with guard tracks [3][29].

2.7.3 Layout design

A schematic is just a representation of a circuit. For the circuit to be printed to a PCB, it is necessary to import the whole developed schematic to a layout tool. The layout of the schematic is made, being the components placed in the way desired by the user and then made the routing of each component. The layout can be updated when a modification is made in the schematic without having to restart the whole layout [39].

When the layout is finalized, two tests are performed for validation. The design rules check identifies physical manufacturing errors such as hole sizes, the clearance between components and wires, and minimum wire size. The layout versus schematic checks connectivity by comparing the layout and schematic connections, trying to find crossed wires or unconnected pins [39].

Layers stack

The stack layer of a PCB is determined by taking into account several factors that determine the resolution of the problem. It is necessary to make several trade-offs, such as the cost of the board, the manufacturing technology, or even the board's functionality to be developed. The layer stack can have just one layer or have 30 or more layers [3].

For boards intended for high speed and high performance, the number of layers is usually high. A four-layer board is a good option for low-cost projects. Two-layer boards are also used, but these present EMC design challenges. A 4-layer board usually has two signal layers and two plane layers. A plane layer is a layer that has a complete trackless polygon (convex), typically used for power and ground signals [3]. Between each layer, a dielectric material is used. This material ensures a non-conductive substrate between the different layers. A 2-layer board typically has two layers that have signal, power, and ground traces or polygons[3].

Vias type

The signals between different layers on a PCB are connected through vias. There are various vias such as through hole, microvia, buried via, and blind via. The most commonly used are through hole, and microvia [31].

Through-hole vias span the entire PCB stackup and can connect to any layer [38][53].

Blind vias are located on the outer layer of the top and bottom of the circuit board. This type of via is used for signals to travel through an internal layer [38][53].

Buried vias start from one internal layer and go all the way to another internal layer without reaching the surface layer. These vias are holes that are plated inside the core of the circuit board [38][53].

Microvias are miniaturized blind or buried vias. They can only pass through two layers if the dielectric is not too thick. However, it is ideal to only cross one layer for maximum reliability. These tracks are conical in shape [53]. The microvia has several advantages. Due to their small size, these require less space, saving board size. From an electrical point of view, the microvias increase the board's performance, as the inductance is reduced due to the reduced path created by the microvias compared to the through-hole. As a result, problems such as cross-talk or EMI are reduced [38].

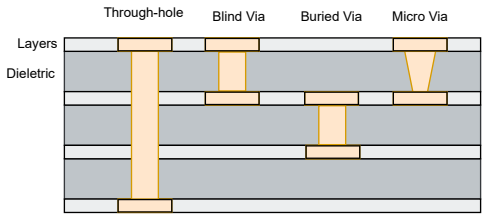


Figure 32: Vias types [53]

2.7.4 Conclusion

This chapter presents state of the art in gas detection. Different gas detection methods and sensors that integrate these methods are presented. Different artificial intelligence algorithms are also presented, namely a decision tree, SVM, and NN. Finally, the different steps for hardware development were depicted, and important notions for designing a printed circuit board were presented.

3 System specification

In this chapter, the system requirements and methodology used to solve the proposed problem will be discussed. After, the options chosen for developing a PCB that satisfies the indicated requirements will be studied. Finally, the firmware for integrating the BME688 sensor will be analyzed.

The system should present a solution for detecting (H_2 & CO_2) gases concentration using a low-cost sensor, in this case, the BME688. It is intended to monitor the environment where the board to develop is, in order to classify the same environment.

This solution has the architecture presented in figure 33. The architecture consists of the sensing element, the BME688, the processing unit to be chosen, and the unit responsible for sending the data to the server.

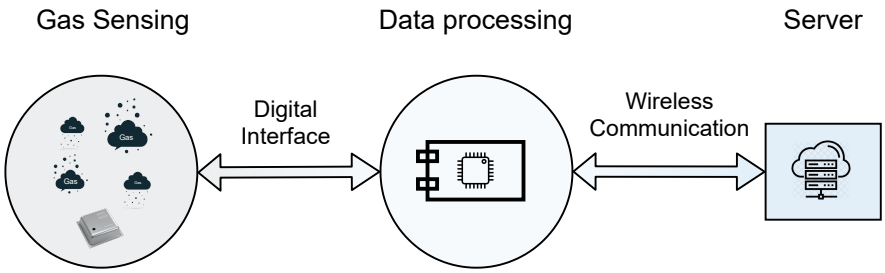


Figure 33: System architecture

The system has the following requirements:

- Should be low power;
- Should integrate the BME688 sensor for gas detection and be capable of execute AI algorithms to detect H_2 & CO_2 gases;
- Should be able to integrate artificial intelligence algorithms;
- Must be able to send data remotely to a server;

In the first phase, the BME688 *DevKit* board was used to study the sensor and to survey the resources needed to solve the problem [61]. With this kit, the hardware used to integrate the sensor was analyzed.

The necessary API's to interface with the sensor were implemented at the software level. Subsequently, a custom PCB was developed to meet the problem's requirements and a software stack was deployed in the developed PCB.

Changes were made at the hardware level, aiming essentially at reducing energy consumption. As such, the processing unit was changed, and the number of BME688 sensors was also changed.

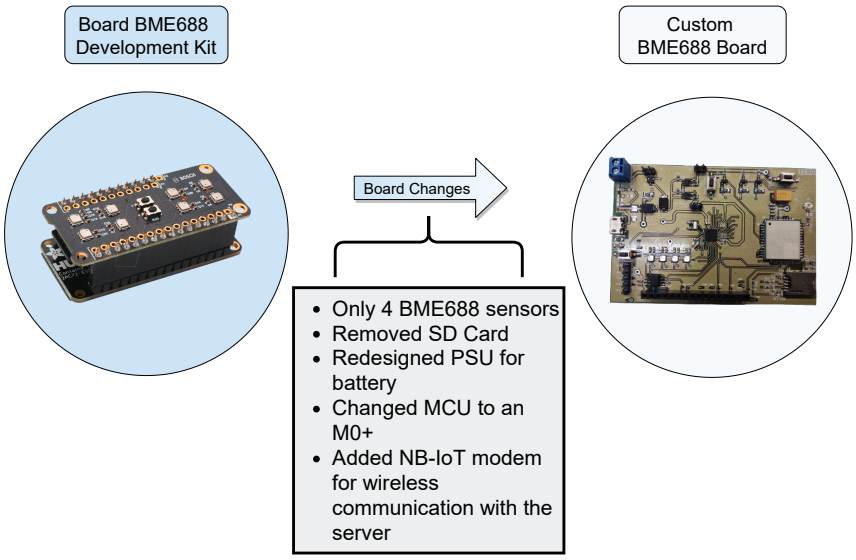


Figure 34: Hardware development

Figure 34 shows the board provided by "Bosch Sensortec" and the board developed to meet the requirements of this dissertation. The main changes were in the power supply management unit, the microcontroller, the number of BME688 sensors used, and the modem that was added. The next section will present the choices made at the hardware level for the development of the board and the software design used for communication with the sensor.

3.1 Hardware

The block diagram of the BME688 Development kit board is depicted in figure 35.

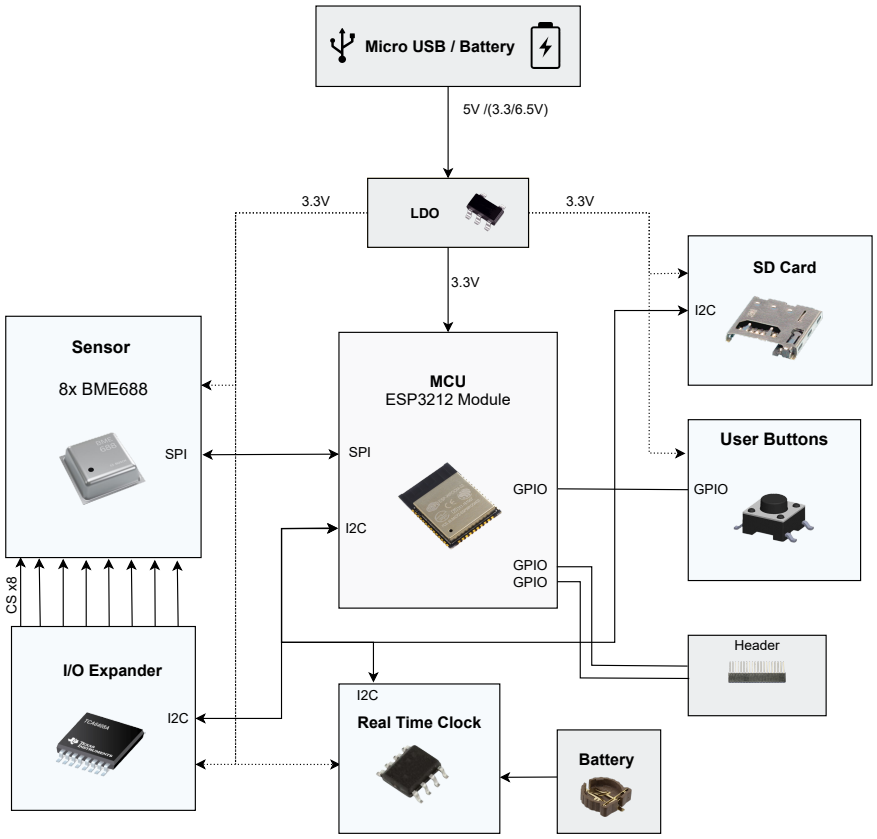


Figure 35: BME688 Development kit block diagram (reference board/ commercial board)

This board can be powered via USB or via battery. It supports a battery with a voltage level between 6.5V and 3.3V since the low dropout voltage regulator (LDO) equipped with the board supports values in this range. This regulates the input voltage to a voltage value of 3.3 V. It is responsible for powering the MCU, eight BME688 sensors, an I/O expander, a real-time clock (RTC), an SD card, and two general purpose push buttons.

The MCU integrated into the board is a dual-core 32-bit LX6 microcontroller from ESP Tensilica. It has an operating voltage between 3.0 and 3.6V. It has 448kB ROM and 520kB SRAM [69].

The board has an external RTC, the PCF8523 from the manufacturer "NXP Semiconductors" [59]. If the board is not powered, the RTC will be powered by a dedicated battery. This behavior is due to the switch-over circuit of the RTC. The RTC communicates via I2C with the MCU. It provides the year, month, day of the week, day, hour, minute, and second based on a 32.768kHz crystal [59]. Also, the board comes with an SD slot, wich makes it possible to save sensor data to an SD card. All the buttons are for general use and can be utilized for various purposes.

The kit has two PCB attached to each other. The main PCB that integrates the MCU and the $2^n d$ PCB that integrates the 8 BME688 sensors, the RTC, the I/O expander, the SD card slot, and the general purpose buttons.

The I/O expander is a 16-pin device with a voltage between 1.65V and 5.5V. It provides 8-bits of general

purpose parallel input/output. It was used as a solution to increase the number of I/O. It is a trade-off between the number of GPIOs used and the software complexity. The I/O expander is configured and controlled by the MCU via I2C [28].

The I/O expander is used as a means of communication with the sensors. Using register 3, it is possible to set the direction of the output ports of the I/O expander. If register 3 (8 bits) has a zero bit, the port corresponding to that bit is set as input, with a high impedance output driver. If the bit has a one, the corresponding pin is set as output [28]. The table 10 represents register 3 of the I/O expander. By default, all expander pins are configured as output.

Table 10: I/O Expander register 3 [28]

Register adress (Hex)	**BIT**	**C-7**	**C-6**	**C-5**	**C-4**	**C-3**	**C-2**	**C-1**	**C-0**
03	Default	1	1	1	1	1	1	1	1

In register 1, it is possible to put the value (high/low) for the pins that were defined as outputs. From this register, the logic levels of the pins defined as input can be read [28]. The table 11 represents register 1 of the I/O expander.

Table 11: I/O Expander register 1 [28]

Register adress (Hex)	**BIT**	**O-7**	**O-6**	**O-5**	**O-4**	**O-3**	**O-2**	**O-1**	**O-0**
01	Default	1	1	1	1	1	1	1	1

Thus, the I/O expander is used to enable the chip select of each sensor. Register 3 allows to define the pins as outputs, and register 1 allows to control the value of each pin (high/low). The pins of the I/O expander are connected to the chip select of each sensor. To start communication with one sensor, the pin is set to a low level using register 1. At the end of the communication, the pin goes to a high level. This option allowed to free eight GPIOs from the MCU, which can be used for other purposes. However, there was an added complexity at the hardware and software level for communication with each of the sensors.

The complexity at the software level is because it was necessary to use I2C to initiate communication with the sensor and then use SPI to continue communicating with it.

This kit was crucial not only for learning how the sensor works but also for realizing the new hardware, as it was a reference for the new board.

As such, the diagram in figure 36 shows the block diagram of the hardware developed in this dissertation.

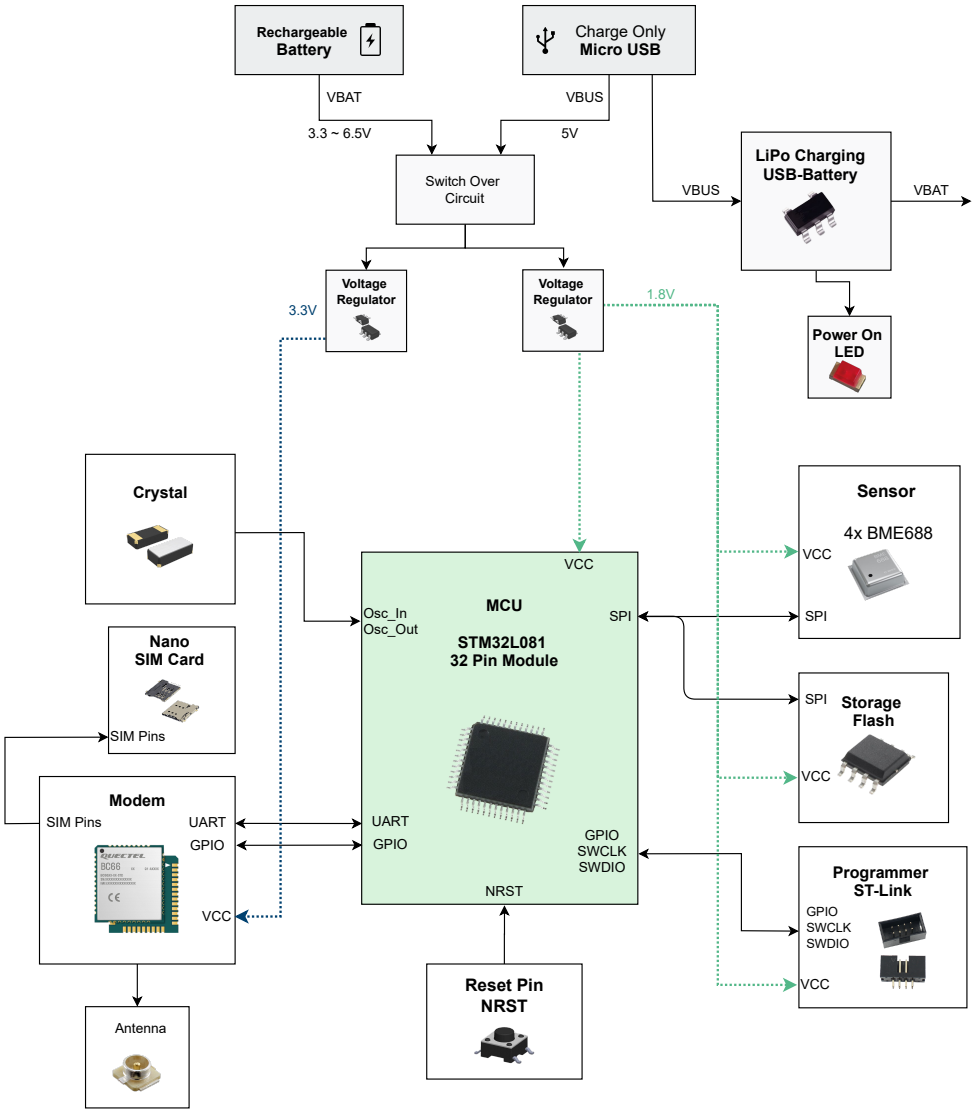


Figure 36: BME688 Custom board block diagram

Only four BME688 sensors will be used in the new board. This number of sensors is placed as a form of selectivity for future use. As these sensors have different responses for the same gas and the same concentration, it may be beneficial to use all four sensors for future classification of the gas. Thus, reducing the number of sensors is a trade-off between low power and low cost versus classification accuracy. This will be addressed in the implementation section.

The use of a modem is essential for sending data to the server and is therefore integrated into the developed PCB.

All processing will be done using a low-power MCU, the Arm cortex M0+. External hardware, the ST-LINK, is used for programming and debugging this microcontroller. As such, a header responsible for the interface between the MCU and the ST-Link was included.

A USB connection is used to power the board. A battery can also be used. This system will be controlled

in such a way that when the USB connector is connected, the battery is charging. Without the USB, the battery will be powering the whole circuit.

Headers were included to access the microcontroller pins that are not used, such as GPIOs, and for test points. It is possible to access the desired voltage levels on this header and use SPI or UART.

A crystal oscillator is also included for the MCU. A SIM card and an antenna are included together with the modem for the NB-IoT connection.

Next, all the choices concerning the hardware developed will be analyzed. Whenever necessary, notions regarding specific components will be presented to facilitate understanding the entire content.

3.1.1 Power circuit

This section will introduce the components used to power the board and why each component was chosen.

The board can be powered by two methods, via a USB cable or by using a battery. For this purpose, hardware logic is required so that the battery powers the board, and if a USB connection is used, it is responsible for the power supply. If the battery and USB connection are used simultaneously, the USB connection shall be the power supply used.

With this, a circuit capable of charging batteries was developed. The board should be connected via USB, and the battery terminals should be connected to the board in order to charge the battery. The battery must meet the requirements of the component responsible for its charge.

The circuit that switches between the battery and USB power supply is shown in figure 37. It uses a resistor, a P-Channel enhancement type MOSFET and two Schottky diodes. These diodes present a low direct voltage drop. This equals less power dissipation during conduction, and this parameter is essential for low-power applications. These types of diodes are used instead of the PN diodes since low power is a system requirement, and the circuit input is a low voltage input, in which significant losses could be a problem.

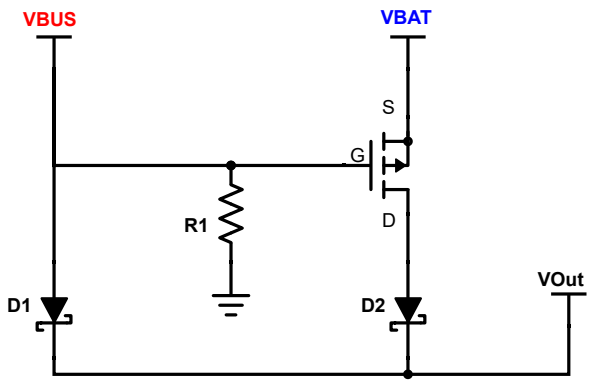


Figure 37: Power circuit V_{BUS}, V_{BAT}

The R1 resistor is a high-value resistor ($1M\Omega$) used so that the gate voltage at the MOSFET does not oscillate when there is no USB connection. Thus the gate is connected to the ground by a resistor. The Schottky diode D1 serves to protect the circuit against reverse currents. If a battery is being used, the circuit upstream of the diode is not powered. This happens if there is no USB connection. If there is a USB connection, diode D2 protects against reverse currents. The upstream circuit is not powered.

In the presence of the USB power supply, V_{Bus} is equal to $5V$. The value of V_{BAT} will be between $3.6V$ and $5V$, being the typical values for a battery. Therefore, the voltage between the gate and source will be between:

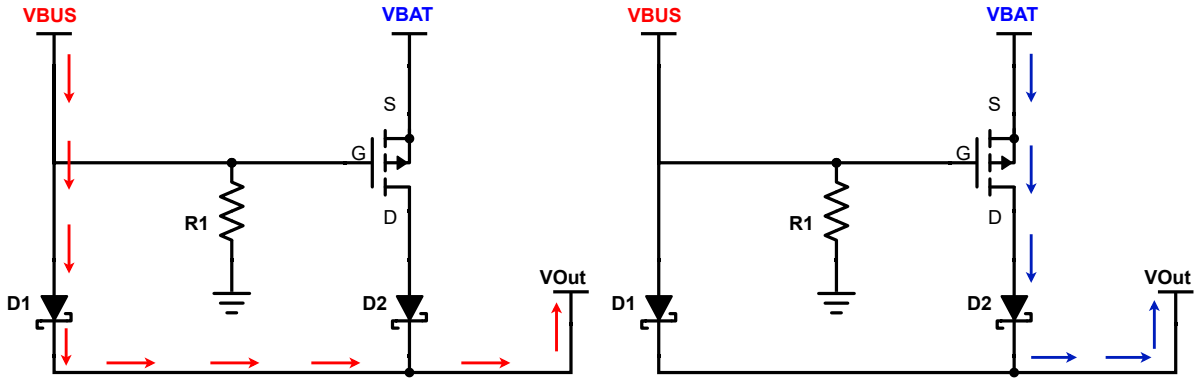
$$\begin{aligned} V_{BUS} : ON \rightarrow V_{GS} = 5 - 3.6 \Leftrightarrow V_{GS} = 1.4 \Leftrightarrow Off \\ V_{BUS} : ON \rightarrow V_{GS} = 5 - 5 \Leftrightarrow V_{GS} = 0 \Leftrightarrow Off \end{aligned} \tag{10}$$

For a P-Channel MOSFET, the voltage V_{GS} must be less than V_{th} (Threshold value) for it to start conducting. In the mentioned conditions, the MOSFET will not start conduction, so the whole circuit will be powered by V_{BUS} as shown in figure 38a.

When the USB connection is not used, the source voltage is higher than the gate voltage.

$$\begin{aligned} V_{BUS} : Off \rightarrow V_{GS} = 0 - 3.6 \Leftrightarrow V_{GS} = -3.6 \Leftrightarrow On \\ V_{BUS} : Off \rightarrow V_{GS} = 0 - 5 \Leftrightarrow V_{GS} = -5 \Leftrightarrow On \end{aligned} \tag{11}$$

Therefore, V_{GS} is lower than V_{th}, and the MOSFET enters conduction. The voltage V_{BAT} will feed the circuit as shown in figure 38b.



(a) Power circuit V_{BUS} (b) Power circuit V_{BAT}

Figure 38: Power circuit V_{BUS}, V_{BAT}

Power supply simulation

A simulation was developed to evaluate the power supply switching between the USB plug and the battery. For that, the *LT-SPICE* tool was used. The circuit is shown in figure 39.

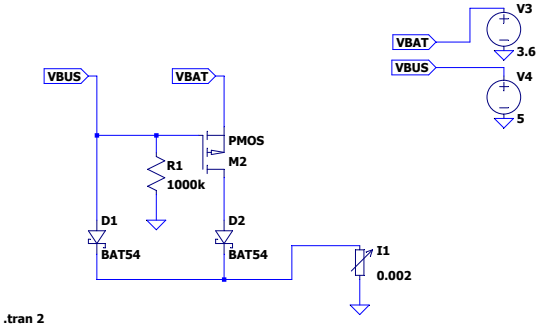


Figure 39: Simulation circuit

For simulation purposes, a load with a current value of $2mA$ was used. V_{BAT} is the voltage at the battery terminals, and V_{BUS} is the voltage at the USB plug terminals.

For values of $V_{BUS} = 5V$ and $V_{BAT} = 3.6V$ it was concluded that the Schottky diode D2 does not conduct, and no current flows through it. Diode D1 conducts, and the circuit is powered via USB.

When the value of V_{BAT} is between 3.6V and 4.8V and $V_{BUS} = 0V$, it was seen that diode D1 does not conduct. The whole circuit is powered by the battery. When there is a voltage at V_{BUS}, diode D1 conducts, and the circuit is powered via USB.

Battery charger

As already mentioned, there will be a circuit capable of charging the batteries when the board is powered via USB. Therefore, the Microchip MCP73831/2 was included [43].

It meets all the specifications required of the USB power bus and is therefore suitable for charging the battery via USB [43].

Charging is carried out on Li-Polymer and Li-Ion batteries. The battery charger component is set for charging a battery with a nominal voltage of 4.2V.

The constant current value is set via an external resistor. If the value of this resistor is $10k\Omega$ the current is limited to $100mA$. If the resistor is $2k\Omega$ the current will be $200mA$.

The main features of the component are summarised as follows:

- Programmable charging current $15mA$ up to $500mA$
- Voltage regulated to 4.2V
- Selectable end of charge (5%, 7.5%, 10% or 20%)

The circuit developed for the battery charger is shown in figure 40.

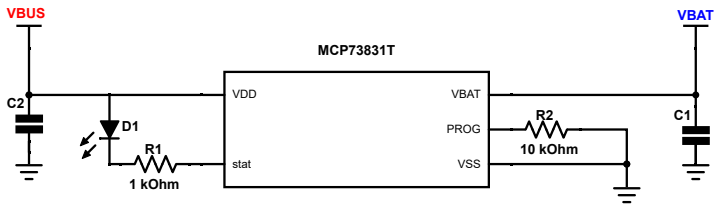


Figure 40: Battery charger circuit

The capacitors used are two tantalum capacitors. The Microchip manufacturer recommends using these capacitors for greater output current stability [43]. The value of these capacitors is $4.7uF$. The output current has been limited to $100mA$ by a $10k\Omega$ resistor.

A connector has been placed to connect the battery terminals. This connector is from the manufacturer "Te Connectivity ®" [14]. It has two contacts, one for the VCC terminal and one for the ground. The schematic is shown in figure 41.

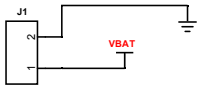


Figure 41: Battery conection

Battery

The batteries that can be used for charging purposes will be Li-Polymer and Li-Ion type batteries. However, this would restrict the type of batteries used on a development board. Although the battery charger will be implemented on the board, it will not be bought and soldered onto the development board. This way, there will be financial savings and no need to restrict the type of batteries to be used. So, the batteries can have a voltage level between 3.3V and 6V. This value is the maximum voltage level allowed by the voltage regulator.

It can be seen in figure 42, the batteries used that were available in the laboratory. They are batteries with a nominal voltage of 1.2V, connected in series. In total, there are 4 Saft [58] batteries that supply 4.8V and a capacity of $2200mAh$. These batteries are of the nickel-metal hybrid VHT Cs type and present a discharge temperature between -20°C and 55°C [58].

(a) Saft battery (b) Batteries used

Figure 42: Battery

USB ESD and EMI protection

The USB socket can be subject to electrostatic discharge since it is a metal connector. If it is not properly protected, it may damage the board being developed or prevent it from working correctly. Therefore, it is beneficial to understand the basic principles of ESD, the types of protection used, and how they work [37].

Protection for the USB port will be required. A transient voltage suppressor (TVS) diode must be selected for this purpose. When selecting a TVS diode, it is essential to look at the breakdown voltage of the TVS, V_{BR}. When it is intended to protect the board against the ESD, it is vital to know the voltage going through the TVS in normal operation. For example, if it's intended to protect an I/O of a board with working voltage of 1.8 V, the breakdown voltage of the TVS cannot be less than 1.8 V. If this happens, the TVS will come into operation whenever the port is used [27].

There are various USB discharge scenarios. An example will be direct ESD discharge to a USB drive or even a charged human holding a cable and plugging it into a USB connector [41]. For this reason, a bidirectional TVS diode will be used to protect the USB path. The TVS used will have to have a breakdown voltage above 5 volts, as this voltage is the input voltage of the USB. The symbol of a TVS diode is shown in figure 43.

Figure 43: TVS bidirectional diode [27]

In the next iteration of the hardware, a Schottky diode should be placed on the V_{BUS} line for negative voltage protection since the bidirectional TVS will only protect against negative voltages above -5V.

Figure 44 represents the circuit of the USB connection, and the ESD protection used. A fuse was also used. This is from the manufacturer *Bourns ®* and is a high inrush current fuse. It offers protection against overcurrent and presents a nominal current of 2A [65].

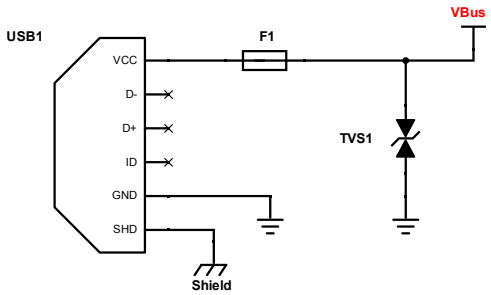


Figure 44: USB protection circuit

When the USB cable is plugged in, the shield of the USB connector may have problems with EMI and may transmit or radiate interference. It is, therefore, essential to protect this device from possible electromagnetic interference. The same principle was used to protect the data card and SIM card.

Thus, it is necessary to find a solution for the USB shield in order to protect the circuit. Therefore, four options have been analyzed:

1. Connect the shield directly to ground
2. Connect the shield to ground through a resistor and a capacitor
3. Connect the shield using a ferrite bead
4. Do not connect the shield

The first option is not an ideal option. Directly connecting the shield to the ground will create a direct path from the ground plane to the shield, turning the USB cable into an antenna [4].

Some suppliers and producers publish design recommendations and guidelines for the connection of the shield USB. From this research it was analyzed that some of them recommend the second option [4] [36] [10].

Atmel [4] states that it is recommended to ground the shield through an RC filter to limit the USB cable's antenna effect.

Microchip [10] also reports that positive EMI behavior has been observed in stand-alone design when the USB cable shield is connected to the ground via a single-resistor circuit in parallel with a capacitor.

Texas Instruments recommend in a guideline to use a ferrite bead to prevent EMI from entering the cable shield [75].

As for the last option, is a consensual recommendation from the manufacturers to not leave the shield unconnected. Not connecting the shield will not lead to any solution to the initial problem. It will not bring any additional protection to the circuit in order to prohibit the USB shield connector from radiating EMI.

Therefore, there are two options available, as manufacturers recommend both the option of connecting the shield to a one or more resistors circuit in parallel with a capacitor and connection to a ferrite bead.

A circuit of a resistor in parallel with a capacitor will be designed. This option is because the components are more common than a ferrite bead. Figure 45 shows the equivalent circuit.

Regarding the selection of components, all producers and suppliers use different values for the resistance and capacitor value. However, they mostly use a high resistance value (in the order of $k\Omega$ to $M\Omega$) and a low capacitor value (in the order of nF).

The USB 2.0 protocol has a data rate of 480Mbit/s. Therefore everything above this value will be considered noise [57].

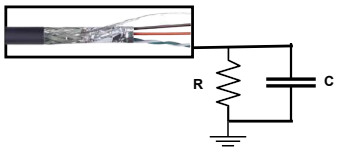


Figure 45: USB shield

$$\begin{aligned} &R_{braid} = 0.1\Omega \\ &R >> R_{braid} \\ &R \sim 1M\Omega \end{aligned} \tag{12}$$

Thus, it is possible to conclude:

$$\begin{aligned} &0.1 \times C > \frac{1}{5 \times 10^8 \times 2 \times \pi} \Leftrightarrow \\ &0.1 \times C > 2 \times 10^{-9} \times 2 \times \pi \Leftrightarrow \\ &C > 3.3nF \end{aligned} \tag{13}$$

Therefore, the capacitor will have to be greater than 3nF. The most commonly used, and therefore easiest to obtain, are the 100nF capacitors. In conclusion, a 100nF capacitor and a resistor with a value of $1M\Omega$ will be used.

Figure 46 shows the circuit developed.

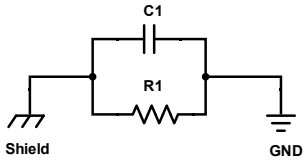


Figure 46: USB shield protection

Voltage regulator

For the board's power supply, it is necessary to choose the component to be used to lower the voltage. As such, two options have been analyzed. Switching regulators and linear regulators, in particular low-dropout-regulators (LDO). Switched voltage regulators generally provide higher efficiency than the LDO, which can be from 80% to 95%. However, they have a higher ripple [64] [56]. For cases where there is a high input and output voltage difference and high currents, this is generally the best option [34].

For the current problem, a battery with a voltage of 4.2V to 5V can be used. The battery charger, in this case, will have no effect as the batteries must have a nominal voltage of 4.2 V, and any battery with a voltage above this value will not be appropriated for charging. The power domains of the developed board will be 1.8V and 3.3V. The modem is powered at 3.3V, and the other components at 1.8V. Therefore it will be necessary to step down the voltage from 4.2V to 1.8V and 3.3V. In these circumstances, it is possible to conclude that there is a big difference between the input and output voltage. Therefore the use of an LDO may not be effective [34].

Therefore a switched voltage regulator will be chosen. There are two possible choices for this type of regulator pulse-width modulation (PWM) and pulse-frequency modulation (PFM). PFM-switched regulators show better efficiency for low-load currents. The board's purpose is low power, so the best solution is to use a PFM-switched regulator [12] [70].

The two regulators used are from the supplier Torex, of the XC9265 series [78].

These regulators are ultra-low power, featuring an input voltage range from 1.8V to 6V. They can achieve an output current of around 200mA. The standby current has a maximum value of 1μA. The current consumed (I_q) by this device for the desired voltage levels is maximum 0.9μA and 2.1μA.

As previously stated, these regulators see their efficiency increase when there is a more significant difference between the input and output voltage [34]. As such, these regulators are placed in parallel. Putting them in series would reduce efficiency because the input voltage would be lower.

Figure 47 shows the circuit of the voltage regulators.

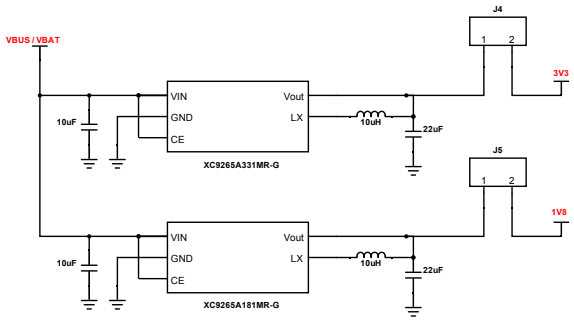


Figure 47: Voltage regulator schematic

Two jumpers were placed after the voltage regulators, the J4 and J5. The jumper J4 disable the power supply of the modem. The J5 disables the power supply to the microcontroller and all sensors.

3.1.2 MCU circuit

This chapter will describe the choices made for the microcontroller circuit. The microcontroller chosen will be presented, and all the choices made to develop the circuit will be discussed. All the circuits will be presented.

MCU

For processing the sensor data controlling the 4 BME688 sensors of the PCB, and fulfilling all the system requirements, the use of an MCU was required as well as the development of an SW stack (firmware + application layer).

Since the system is low-power, the microcontroller chosen was the ARM Cortex M0+. This Arm Cortex M core is indicated for low-power applications, presenting a high energy efficiency. Therefore, the microcontroller chosen will be the STM32L081KZU6 microcontroller. This microcontroller was available in the laboratory, being well suited for the purpose. There would be other options that could be analyzed, but given the shortage of components this MCU was selected and used.

The microcontroller has 32 pins. It has a flash memory of about 192kBytes and a RAM memory of 20Kbytes. It has a low-power timer, 40 GPIOs, and a low-power UART. It has a maximum frequency of 32MHz and is 5x5mm^2 in size. The MCU operates between 1.8 and 3.6V. It has different operating modes, and different power consumption in those modes [16].

- Sleep Mode: In sleep mode, only the CPU is stopped, and the peripherals can wake up the CPU when an interrupt, or an event occurs.
- Low-Power Sleep Mode: In Low power sleep, only the CPU clock is stopped. The main difference between sleep mode and low-power sleep mode is that the voltage regulator goes to low-power mode. When the event or interrupt is triggered, the system goes into running mode. In both modes of sleep, the clock frequency and the number of enabled peripherals are limited.
- Stop Mode: In Stop mode exists two possible modes, stop mode with RTC and stop without RTC. In this mode, the device achieves the lowest power consumption while retaining the SRAM and register contents. In the stop mode with RTC, the device can be woken up by the RTC. In the other mode, this is not possible.
- Standby Mode: In the standby mode, there are two options, the standby mode with RTC or without RTC. This mode presents the lowest power consumption. After entering standby mode, the RAM

and register contents are lost except for registers like the RTC, wake-up logic, LSI, and RCC_CSR register. In the standby mode with RTC, the device can be woken up by the RTC. In the other mode, this is not possible.

The figures 48, 49 and 50 summarise all the information regarding the operating modes, indicating the resources that remain enabled or not [16].

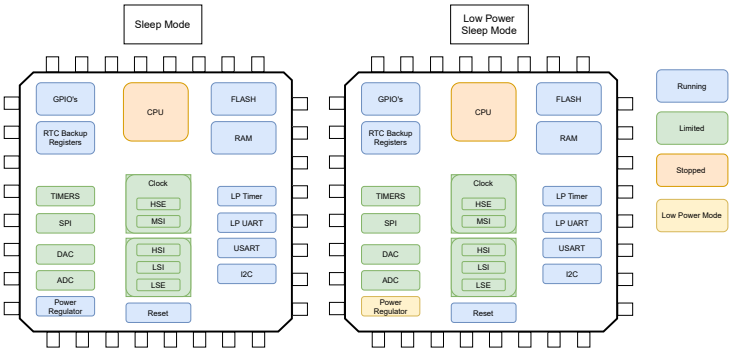


Figure 48: Sleep mode M0+ [6]

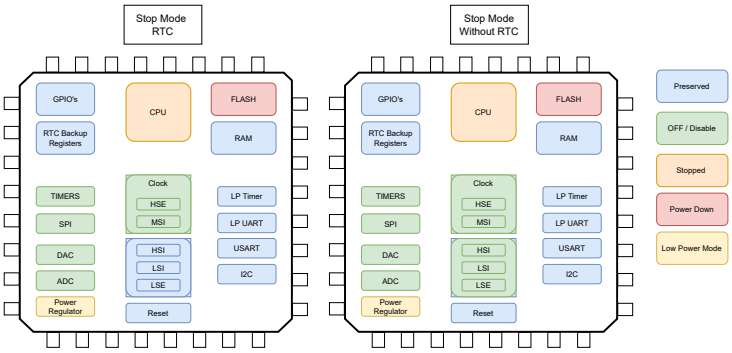


Figure 49: Stop mode M0+ [6]

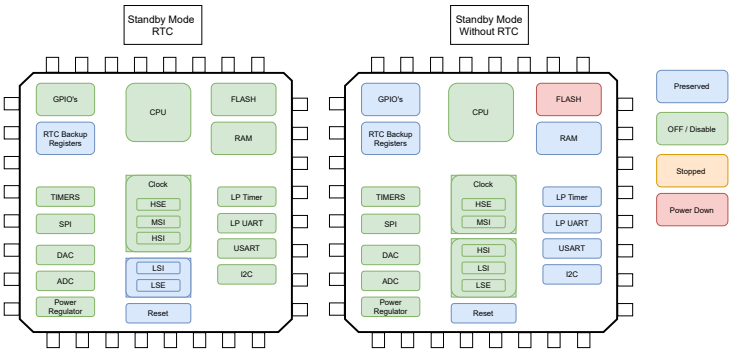


Figure 50: Standby mode M0+ [6]

Crystal

The MCU features a real-time clock and five backup registers, all of which are powered in any of the operating modes shown. It also provides programmable alarms in which the device can wake up when it is in low power mode. It is provided a calendar with seconds, minutes, hours, and years in a binary code format [16]. The RTC clock source can be between several options an external 32.768kHz crystal or the internal low-power RC oscillator.

A few factors were considered for choosing the RTC source. The first is that without the external crystal, the RTC would not work since some sources are disabled in some operating modes. Another feature is that the internal oscillator is more unstable than the crystal. In the worst case, the internal low power oscillator (LSI) has a frequency drift of 4% [16]. The crystal oscillator has a frequency tolerance of 20 ppm (0.002%) [2]. However, adding an external oscillator increases the final board's price and the board's size, and the crystal layout needs to be carefully considered.

Considering the pros and cons, it was decided to put in a 32.768kHz crystal from the manufacturer ABRACON ®[2].

The circuit that implements the crystal is shown in figure 51.

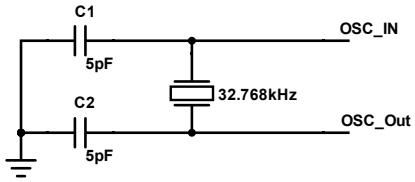


Figure 51: Crystal schematic

GPIO headers

As mentioned before, GPIO headers were included. This interface was made with a single-row header with 16 positions. Some power pins were also placed on this interface. Like the USB socket, this header needs protection against electrostatic discharges.

For this purpose, TVS diodes were placed on all pins of the header. As mentioned before, the diodes can only start conducting from a certain voltage level for good protection. Since there will be pins coming from the modem, whose voltage level is 3.3V, TVS diodes will be placed to operate for this voltage level.

The TVS diodes chosen are ESD9L3.3ST5G from the manufacturer "Onsemi ®" [18]. The breakdown voltage on these diodes is 4.8V minimum [19].

However, not all pins operate at 3.3V, and there are pins with 1.8V. Therefore in the next hardware iteration, the TVS diodes should be rectified.

All VCC pins were filtered using coupling capacitors.

ST-Link

Programming the microcontroller will require using an external device, the ST-Link [68].

The ST-Link is a debugger and programmer from STMicroelectronics for programming STM32 and STM8 microcontrollers. The JATG/SWD interface allows communication with the desired microcontroller. The USB interface provides communication with a PC, and an STM32 device via development environments such as Keil [68]. The features that the device presents for the JTAG and SWD interface are:

- Supports serial wire debug (SWD) and serial wire viewer (SWV) communication;
- 1.65V to 3.6V application voltage supported on the JTAG/SWD interface and 5V tolerant inputs;
- Supports JATG communication;
- Supports serial wire debug (SWD) communication;

The chosen ST-Link is designed around an STM32F103C8 device, which incorporates an Arm Cortex M3. In the datasheet, it is possible to see the pinout of the JTAG and SWD [68].

To make the interface between the board and the ST-Link possible, a header with the necessary pins for programming the microcontroller has to be developed.

The serial wire debug (SWD) protocol will be used. Therefore, the pins present in the header will be the reset, target VCC i.e. the MCU VDD (1.8V), serial wire clock, serial wire data I/O, and ground pins.

For the reset pin, a push button was placed. By pressing the button, a reset is done to the microcontroller. The reset is active low, so it is necessary to set the pin to ground to perform a reset. The pin has an internal pull-up. However, an external pull-up has been placed. This ensures that the signal is not at 0V. A capacitor has been fitted to debounce the button. This capacitor should be as close to the pin as possible. Figure 52 presents the circuit.

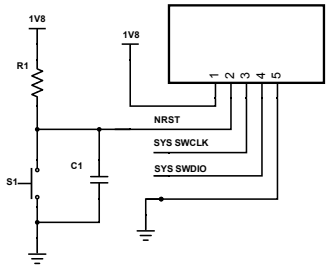


Figure 52: ST-Link interface, reset button

STM32L081KZU6 circuit

Figure 53 presents the schematic which implements the microcontroller support circuitry.

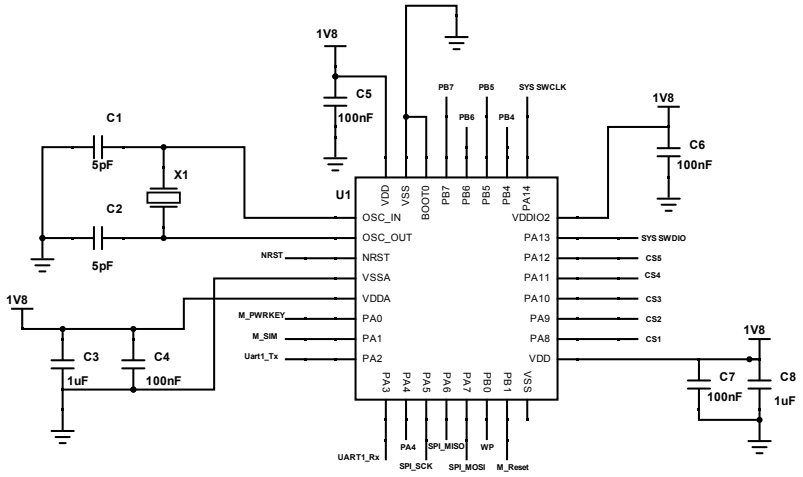


Figure 53: Microcontroller schematic

It is possible to observe the microcontroller pins that go to the ST-Link programmer interface. The pins that refer to the modem are visible with the letter M behind each pin. The capacitor bypass used and the crystal circuitry implemented in the MCU is also verified. The communication protocols pins are also visible, as well as the chip select pins (CS) that go to the sensors.

3.1.3 Modem circuit

The "Quectel ®" BC66 NB-IoT modem was used [55], since it fulfill the communication link and the low power requirement.

The BC66-Na is a high-performance NB-IoT module with low power consumption. This module supports a wide range of frequency bands. It features a compact size of 17.7mm × 15.8mm × 2.0mm. It has different interfaces, such as UART and USB, and different protocols, such as UDP, TCP, and MQTT [55].

The modem has a voltage of between 2.1V and 3.6V and . Supports 1.8V micro SIM card. It is controlled by the MCU using AT commands via a UART link. The UART has a logic level of 1.8V. If the whole application is equipped and powered at 3.3V, a level translator is required [55].

In this situation, a trade-off has to be made. The whole board would be powered at 3.3V, and a level translator would be used for communication with the modem. The other option is to power the microcontroller at 1.8V and all other components and uses two voltage regulators to power the modem at 3.3V. As can be verified the last option was chosen. The advantage of this choice is reflected in the fact that at 1.8V, the sensor consumption would be reduced, and it would not be necessary to choose a

level translator. A level translator could increase the overall consumption of the board, as well as the price. However, this choice leads to the necessity of using an extra voltage regulator, i.e., one to get 1.8V and another for 3.3V.

Next, the different sub-circuits that compose the modem circuit will be analyzed. First, the circuit developed for the SIM Card will be analyzed. The antenna circuit will be presented. The interaction between the microcontroller and the modem will be analyzed. Some simulations will be shown to validate the developed circuits. Lastly, the circuit with the BC66 modem will be presented. All the choices made of all circuits will be explained.

SIM card slot

For the modem to communicate, it will need a SIM card for LTE network access. The SIM card will be powered by the module's internal regulator and supports 1.8V.

Following the datasheet guidelines, the circuit of the SIM card slot was developed. The schematic diagram is shown in figure 54.

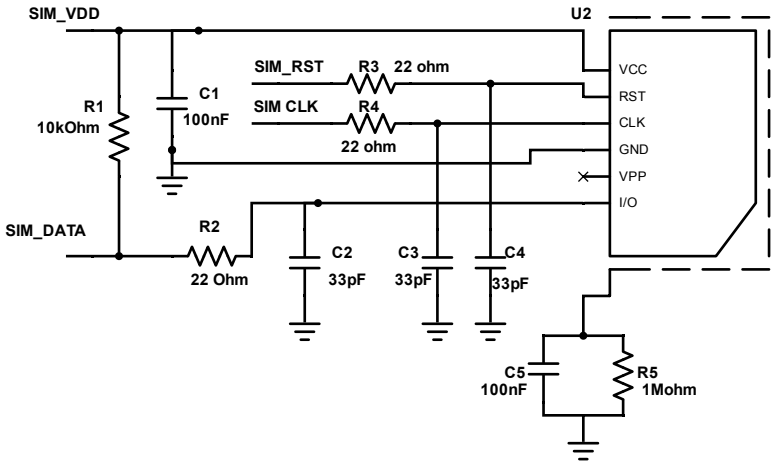


Figure 54: SIM card schematic

The "SIM_DATA" and "SIM_CLK" lines should be kept away from each other to avoid cross-talk. They must be protected separately with surrounding ground in the layout phase [55].

The 22Ω resistors should be connected in series between the modem and the USIM card connector to suppress EMI transmission and enhance ESD protection [55].

The radio frequency bypass capacitors (33pF) on all signal lines improve EMI suppression and should be placed close to the USIM card on the layout phase [55].

A resistor was placed in parallel with a capacitor to reduce EMI emission by the SIM card shield. For the "USIM_DATA" line a pull-up resistor is used [55].

Antenna

The modem will have an interface for an NB-IoT antenna. At the layout stage, precautions will be needed for a good implementation of the antenna in order to achieve good RF performance.

Table 12: Frequncy band BC66-NA module[55]

Mode	Frequency bands
LTE HD-FDD	B1/B2/B3/B4/B5/B8/B12/B13/B17/B18/B19/B20/B25/B26/B28/B66/B71/B85

Figure 55 presents the schematic developed for the antenna. A PI filter was used to obtain a superior RF performance. This filter consists of two capacitors and a resistor. It will be placed close as possible to the antenna for better performance. Since this modem was not used during the thesis, the components were not chosen. Therefore the resistor was a 0Ω shunt, and the capacitors were not placed, leaving freedom for these components to be chosen empirically later.

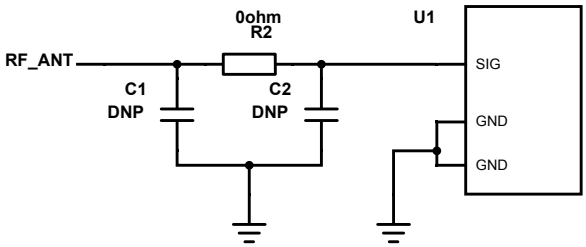


Figure 55: Antenna schematic

Modem Action

The microcontroller will control the modem's operation, having pins to enable and disable features. The pins that the microcontroller will control will be the *PWRKEY*, *PSM_EINT* and *Reset*.

The PWRKEY pin turns the modem on. This pin is active low. The maximum input low-level voltage $V_{IL}max$ is equal to $0.3{*}VBAT$ and the minimum input high level voltage $V_{IH}min$ is equal to $0.7{*}VBAT$.

The reset pin performs the reset of the modem. This pin is active at low. It must remain at a low level for more than 50ms to perform the reset.

The $V_{IL}max$ is equal to $0.25 * VBAT$ and the $V_{IH}min$ is equal to 0.75*VBAT. The "PSM_EINT" pin activates an external interrupt to wake the modem from deep sleep mode.

The "PSM_EINT" pin activates an external interrupt to wake the modem from deep sleep mode. This is active on the falling edge. The $V_{IL}max$ is equal to $0.3 * VBAT$ and the $V_{IH}min$ is equal to 0.7*VBAT.

The circuit that implements the logic to activate each pin is shown in figure 56.

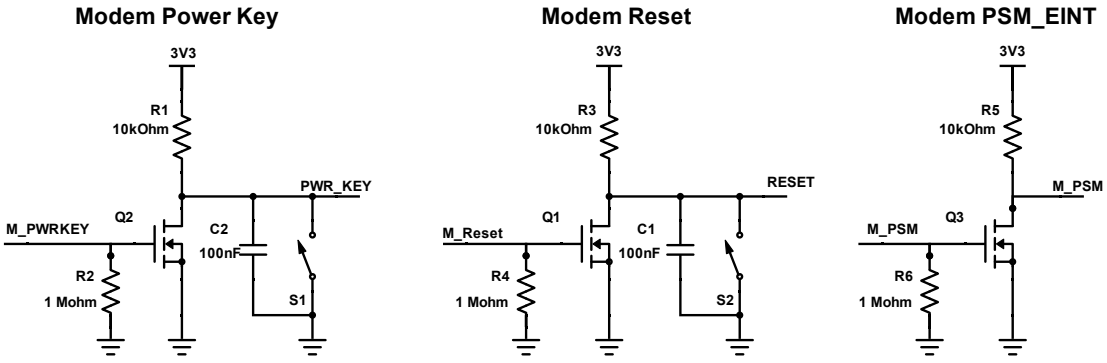


Figure 56: Modem schematic

The pins are internally in pull-up. However, external pull-up resistors have been placed as a safety precaution. Push buttons and debounce capacitors were placed for the reset pin and power.

In all circuits, enhancement-type MOSFETs were used. The gate of the MOSFETs are pins from the MCU. These pins are active whenever $V_{GS} > V_{th}$. Whenever the microcontroller wants to reset, it activates the "M_RESET" pin. This pin switches to a voltage of 1.8V, putting the MOSFET into conduction and grounding the "RESET" pin. The same happens with the remaining circuits.

Simulations of all circuits were carried out to validate them. As the circuits are all similar, it will only be observed the simulation of one. The simulation's schematic is shown in figure 57.

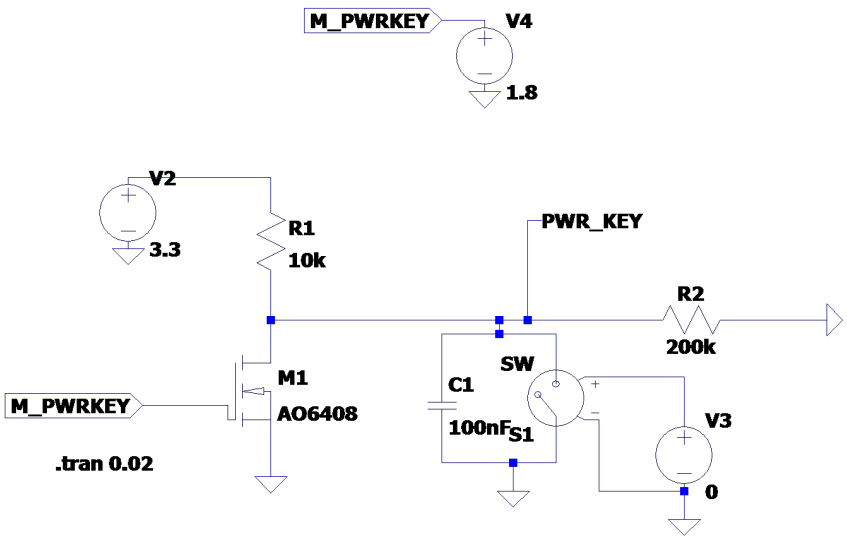


Figure 57: Modem PWRKEY simulation

This circuit simulates the action of the "PWRKEY" pin to turn the modem on.

In this simulation, three scenarios are analyzed. The first is when the pin "M_PWRKEY" is set to high

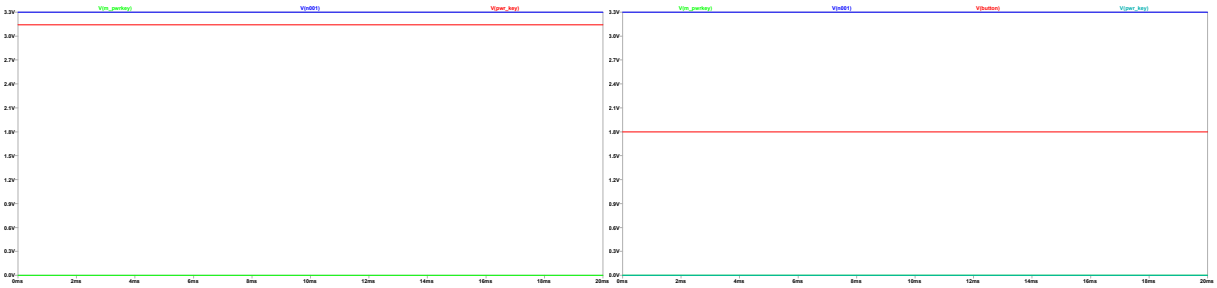
by the microcontroller. The second is when no voltage is placed on the pin, and the third is when the push button is pressed.

The results for the first scenario are shown in figure 58. The pin voltage coming from the microcontroller is equal to 1.8V. Therefore, the MOSFET enters conduction, and the "PWR_KEY" pin that goes to the modem has a voltage of 0V.

Figure 58: Modem PWRKEY simulation first scenario

The second scenario is visible in figure 59a. In this situation, the microcontroller pin is not set high. Therefore, the "PWR_KEY" pin stays at a high level and is not active.

The last scenario is shown in figure 59b. In this situation, the microcontroller pin is at a low level. However, the push button is pressed. The "PWR_KEY" pin goes to 0V, and the modem is turned on.



(a) Modem PWRKEY simulation second scenario (b) Modem PWRKEY simulation third scenario

Figure 59: Modem PWRKEY simulation

BC66-NA

Finally, the final circuit of the modem is shown in figure 60.

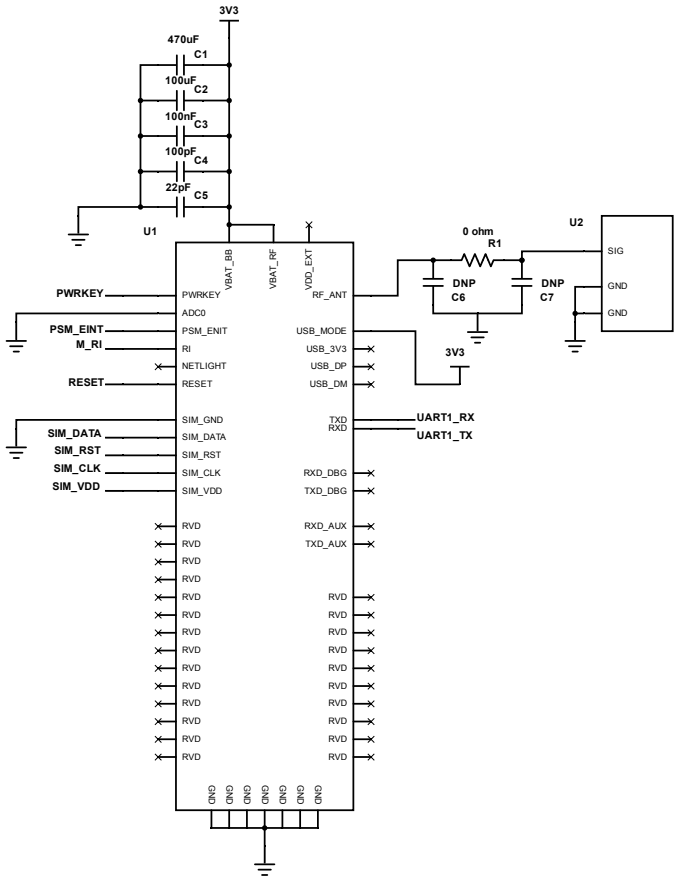


Figure 60: Modem BC66-NA schematic

It is possible to observe the implementation of the antenna and the PI filter in the circuit. The bypass capacitors can also be seen. They are used for the modem's power supply. The modem's inputs and outputs coming from the MCU and the SIM Card are all visible.

The value of the capacitors is shown in figure 60. A 470uF tantalum capacitor was used, followed by 100nF, 100pF, and 20pF capacitors. In the layout phase, the smallest value capacitor should be as close as possible to the component. The others should be placed in increasing order.

3.1.4 Sensors circuit

As mentioned before, four BME688 sensors will be used for CO_2 & H_2 gas detection. The BME688 is a low-power sensor integrated with other temperature, pressure, and humidity sensors.

It has two digital interfaces, SPI and I2C. The interface chosen to interface the sensors will be SPI as depicted in figure 61. With this choice, it will be necessary to have 4 GPIOs for the chip select of each sensor.

The circuit that implements the BME688 sensor is shown in figure 61. It s composed of the SPI lines, the power supply, and the bypass capacitor. The capacitor used is the one recommended by the

manufacturer [6]. The SPI lines used are the clock, MISO, MOSI, and the chip select. Only one sensor is presented for simplicity reasons in this circuit.

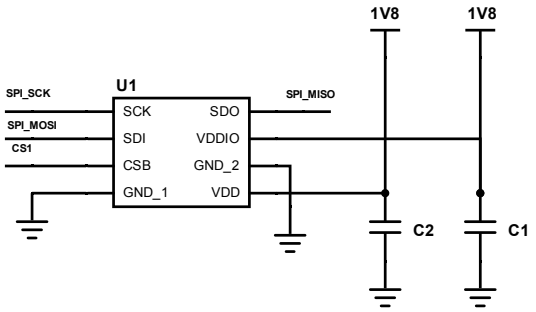


Figure 61: BME688 schematic

3.2 Software

In this section the BME688 sensor was studied. It is important to know this sensor in detail. After studying the sensor, the system stack is presented, as well as the system applications. Then the AI algorithms to be implemented are presented. Finally, a use case is analyzed. This one aims to study the life time of the batteries used in a real gas monitoring context.

3.2.1 BME688

This section describes the theoretical aspects of the BME688 sensor. Through the understanding and study of the sensor, it was possible to develop and implement the software to interface the sensor and run the models to detect CO_2 and H_2. Thus, it is important to understand the digital interface of the sensor in addition to the physical operation already studied.

As already mentioned, the SPI interface will be used by the MCU to communicate with the sensor. On the SPI interface, a 7-bit address is used, and the memory is divided into pages. There are two pages (0 and 1) as shown in table 13. In the reset, page 0 is selected.

Table 13: BME688 SPI interface [6]

Digital Interface	Register Adress Range	Register spi_mem_page	Memory Page
SPI	0x80 to 0xFF	0 (default; power-on state)	Page 0
SPI	0x00 to 0x7F	1	Page 1

The sensor can be heated to different temperatures. It is possible to program the sensor to have different temperatures for a predefined time, creating a unique fingerprint for different gas compositions.

This is called a heater profiling. It is a profile in which the sensor is heated to desired temperatures in a specific period of time. An example of a heater profile can be seen in figure 62.

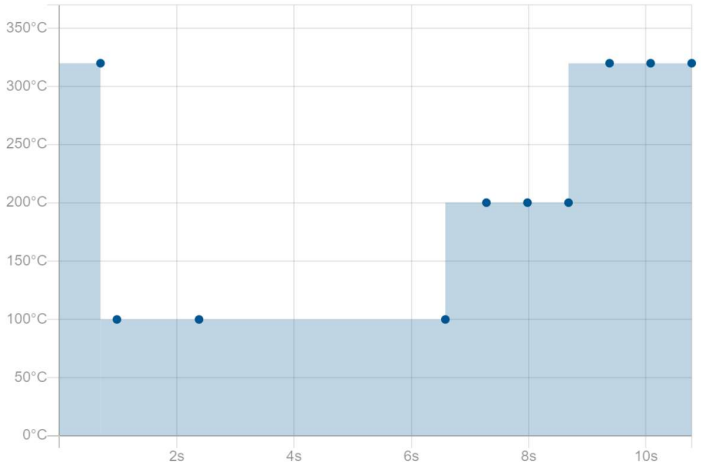


Figure 62: Standard heater profile [6]

The figure shows that the sensor has been programmed to cycle through 10 different temperatures, i.e., 10 heating steps, for a total time of 10.78 seconds. In each step, temperature, pressure, humidity, and gas measurements are done. This cycle takes a total of 140ms. Some steps at different temperatures are repeated. As such, the sensor can remain longer at different heater steps. On this particular heater profile, the cycle must be repeated five times for the first heater step and twice for the second heater step until all the cycles have been completed in all the heater steps. The temperature, pressure, gas, and humidity values are saved in registers, and only the last cycle performed in each heater step is considered valid. The sampling time for each cycle can be changed to a value that the user desires.

The sensor has 3 operating modes which are presented in table 14. For the sensor to perform the heater profile with the cycles and heater steps, it must be in parallel mode.

Table 14: BME688 Operation modes [6]

Operation Mode	Mode 1:0	Key Features
Sleep	00	• No measurements are performed • Minimal Power Consumption
Forced Mode	01	• Single TPHG cycle is performed • Sensor automatically returns to sleep mode afterwareds • Gas sensor heater only operates during gas measurement
Parallel Mode	10	• Multiple TPHG cycles are performed • Sensor will not automatically returns to sleep mode • Gas sensor heater operates in parallel to TPHG measurement

Figure 63 demonstrates the difference between parallel and forced mode operation. In the figure, the T,P,H and Gx represent the temperature, pressure, humidity and gas measurements, where x represents the heater step.

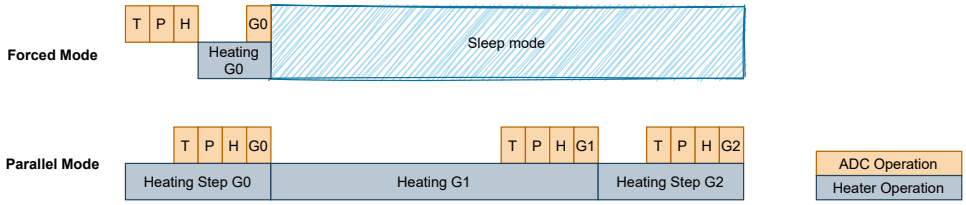


Figure 63: BME688 operation modes diagram

An example of the parallel mode of the BME688 is shown in the diagram of figure 64.

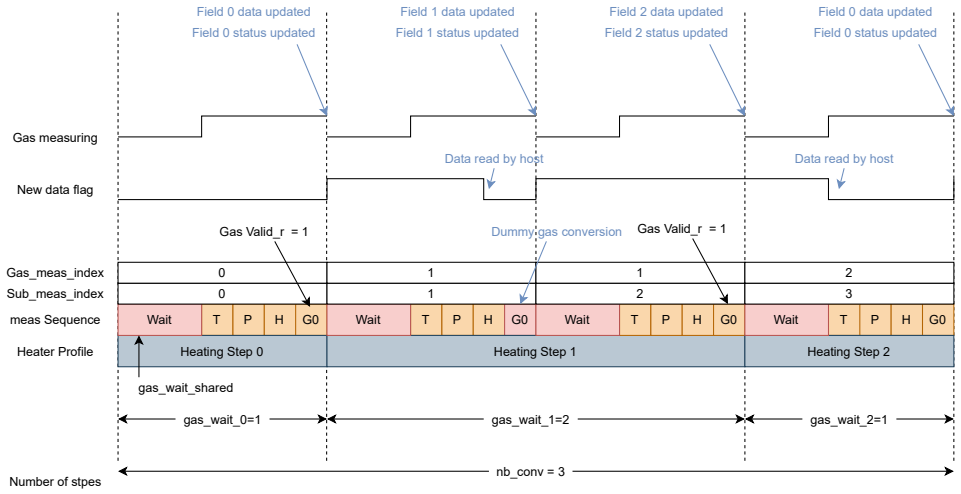


Figure 64: BME688 parallel mode diagram [6]

In the diagram "nb_conv" defines the number of steps. Thus, the heater profile selected has 3 heater steps. The first heater step performs only one temperature, pressure, humidity, and gas measurement cycle (T,P,H,G0). It is possible to define the sampling time for each cycle, and "gas_wait_shared" will be the difference between that time and the time needed by the sensor to make the temperature, pressure, humidity, and gas measurements. At the end of this cycle, the data will be saved in the registers, and the new data flag will be set to one.

For the second heater step, in the example, the sensor would perform two cycles of temperature, pressure, humidity and gas measurements. At the end of the second cycle the data is valid and it will be possible to read the data from the registers 65. The "Gas_Valid_r" bit stores the information if the data is valid or not, at the end of each cycle.

After the sensor performs heater step 2, it will go to heater step 0 again, being the whole process repeated cyclically. The heater profile will have a duration equal to the number of cycles the sensor has done (4) multiplied by the time the user defines for each cycle.

The data is saved for registers (3 fields in figure 65). The index of registers represents the corresponding fields. At the end of each measurement cycle, the data is stored in all the registers of each field. The sensor writes the data to those registers sequentially. When the sensor writes the data to the registers, the bit

"new_data" of the register "meas_status" of each field is set. Therefore the sensor knows which fields (group of registers) are filled.

These registers should be read so that they are not rewritten by the sensor. To do this, the microcontroller must be able to read these registers at a rate greater than the defined time for each heater step cycle.

To get the data, it is necessary to check if the registers have new data to ensure that it has never been read and is effectively new data. As previously said, the register "meas_status" has the bit "new_data", which indicates whether the data is new on that field (group of registers).

To get valid data, it should be checked in the registers "gas_r_lsb" if the data read is valid "gas_valid_r" and if the heating process of the sensor was stable "heat_stab_r". If this data is not valid, may not be used.

An example would be when it is intended to read the data from heater step 1, shown in the example of diagram 64. In the first cycle, the data is stored in the registers of field 1, and in the second cycle, it is stored in the registers of field 2. When this group of registers is read, the new data of the register of field 1 is 1, being new data on that registers. However, that data is invalid since the "Gas_Valid_r" of that field is set to zero. Then the registers of field 2 are checked. Since that group of registers has new data, it will be checked if that data is valid and process it.

In the figure 65 it is possible to see the registers of each field, as well as where the bits "new_data", "heat_stab" and "gas_valid_r" are stored. Each register can be seen in more detail in the BME688 sensor datasheet [6].

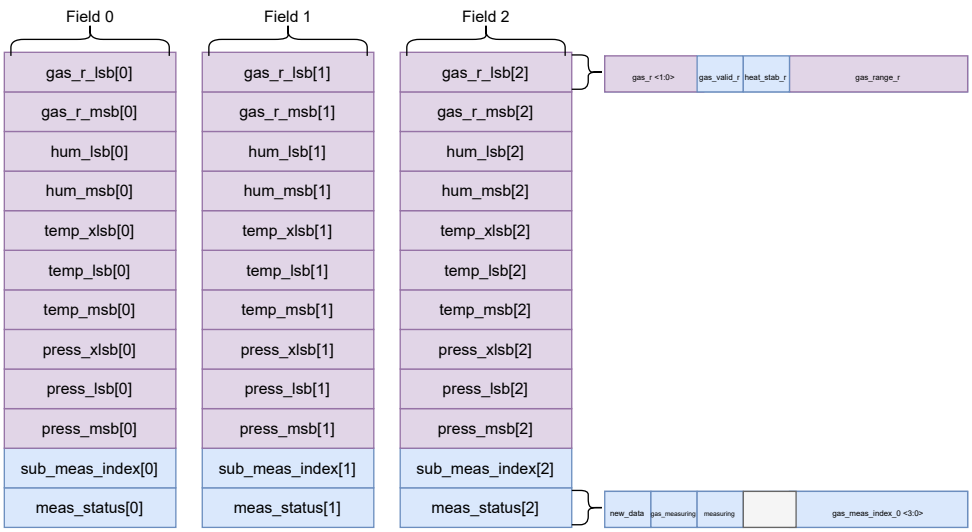


Figure 65: BME688 registers field diagram [6]

3.2.2 System stack

Figure 66 shows the system stack of the system.

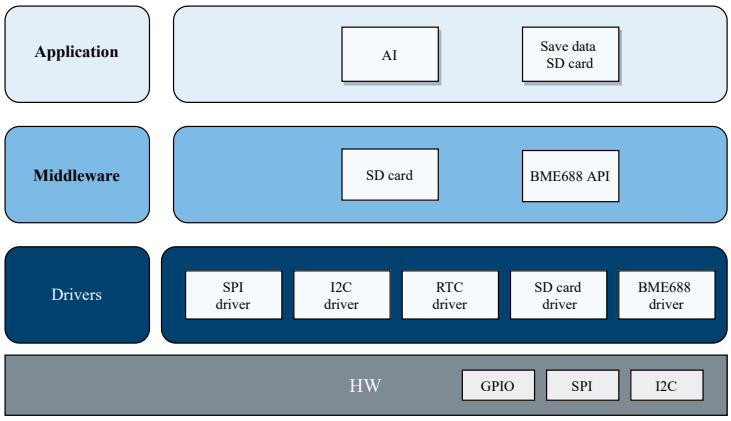


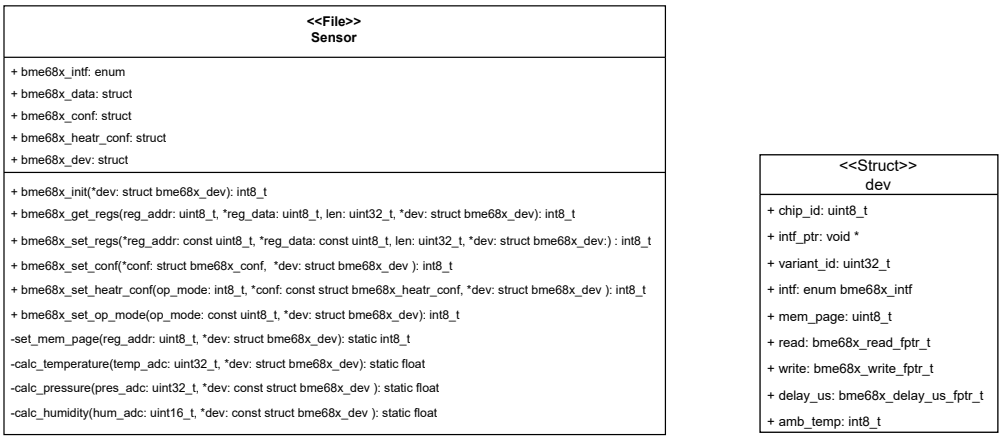
Figure 66: System stack

The system stack consists of 4 layers. The top layer is the application layer. The system stack presents two distinct applications. The first presented in this dissertation is to save data from the BME688 sensor on an SD card. The other application is the artificial intelligence algorithms. The next layer is the middleware layer which comprises the BME688 sensor API and the SD card middleware. The third layer is the drivers. This layer is dependent on the available hardware and, as such, differs for the developed board and the Devkit. On Devkit, besides all the drivers, it also presents the I2C driver, being this protocol used due to the I/O expander.

The bottom layer is the hardware layer. This layer also depends on the board used, Devkit or custom board. As such, it is necessary to analyze the hardware configuration of each board. On figure 35 (page 40), it is possible to see the Devkit block diagram and on figure 36 (page 42), the custom board diagram. On the custom board, in addition to the hardware developed, a connector for SD card (external hardware) is added to save the sensor data.

BME688 API

Bosch Sensortec has developed and provided a low-level API with the main functions to configure and communicate with the sensor. Some of the main functions will be presented below and are presented in figure 67a.



(a) Class diagram of sensor module API

(b) Diagram of struct

Figure 67: Class diagrams

The main structure defining the sensor is called "BME68x_dev" presented in figure 67b. In this structure, it is possible to define the type of interface used, SPI or I2C. This structure comprises two pointers to the read and write functions. Using these pointers it is possible to define any possible interfaces without modifying the API. The pointer jumps to a function that will be responsible for writing or reading values from the sensor registers. The user must implement the methods to which the function pointer jumps, being then possible to use any of the available interfaces.

To pass values to the function, the (void *) is used, and then the cast is made to the type of data desired.

In conclusion, this function pointer provides an abstraction, being necessary to implement functions that perform the writing and reading of sensor registers without changing the developed API.

The function "bme68x_init(struct bme68x_dev *dev)" is responsible for reading the sensor's chip ID and performing its calibration. The function "bme68x_get_regs(uint8_t reg_addr, uint8_t *reg_data, uint32_t len, struct bme68x_dev *dev)" is responsible for reading the registers, and so, used on function explained above. It has as input the register to be read, where the real value will be stored, the amount of data to be read, and the "bme68x_dev" structure previously mentioned. This function uses the function pointer to execute the function implemented by the user, which reads the registers. Likewise, there is a function "bme68x_set_regs(const uint8_t *reg_addr, const uint8_t *reg_data, uint32_t len, struct bme68x_dev *dev)" to write the registers.

The function to set and get the registers uses an additional function, the "set_mem_page(uint8_t reg_addr, struct bme68x_dev *dev)". It checks if the desired address to read/write the register is on the correct SPI page. If not, the function pointer is used to write to the register to change the SPI page.

The function "bme68x_set_conf(struct bme68x_conf *conf, struct bme68x_dev *dev)" performs sensor settings such as temperature, pressure and humidity.

The function "bme68x_set_heatr_conf(uint8_t op_mode, const struct bme68x_heatr_conf *conf,

struct bme68x_dev *dev)" is used to make the settings for the chosen heater profile. This function defines the temperature required for each heater step, the number of cycles performed in each one ("sub meas index"), the time required for each cycle, and the number of heater steps in the chosen profile.

A function responsible for changing the sensor operation mode, "bme68x_set_op_mode(const uint8_t op_mode, struct bme68x_dev *dev)", is provided. This function writes to the specific register for operation mode control.

Sensors communication

After analyzing the functions in the API, it is necessary to develop the functions responsible for reading and writing the registers. These are the functions to which the function pointer jumps and allows communication with the sensor to be finalized. Since these functions depend on the developed hardware, they will be different on the DevKit board and on the developed board. As such, the implementation of these functions can be analyzed in the implementation chapter (page 77).

3.2.3 Save Sensor Data

The sensor monitors temperature, pressure, humidity, and gas resistance values. As such, these are the values stored in the registers. Implementing artificial intelligence algorithms is a requirement of the system, so the sensor's output data is important for training these algorithms. As this data is important it is necessary to save it on an SD card.

As such, methods for storing data on an SD card for both the commercial Devkit board and the custom board will be developed. The methods are presented in the implementation chapter (page 80).

The commercial board was used to get the dataset for CO_2 and H_2 gases. The developed board was not finalized when the dataset was obtained and so, was not used. However, as a means of validating the developed board, the SD card application was developed and implemented on the developed board.

Data collection

The data that will be stored on the SD card is the timestamp of each sample, the sensor number, the temperature, pressure, humidity, and gas resistance, the heater step, and the valid data bit. In table 15 is presented an data frame example.

Table 15: Data frame example

Data frames
Timestamp,Sensor Index,Temperature,Pressure,Humidity,Gas Resistance,Heater Step,Valid Data
00:00:01,1,31.036335,100257.484375,42.107227,343970.437500,0,1

Given the large amount of data the file will present, python will be used for the data processing. It facilitates the construction of graphs and the selection of the data that compose them.

As previously said, this data will be used for training the AI algorithms.

3.2.4 Artificial intelligence

This section will discuss the algorithms for gas detection. The algorithms to be implemented will be two classifiers and a regressor. The classifiers will be decision trees and support vector machines. The regressor will be the neural network. The target gases will be presented in the implementation section, as well as the acquisition of the dataset.

Were implemented algorithms were for only one sensor. The use of 4 or 8 sensors has not been explored in those. However, all board's sensors will be used to monitor the target gas and save a dataset on SD card. The implementation of these algorithms will only be done on the developed board.

Decision Tree

The decision tree will be one of the classifiers implemented. The output generated by the decision tree will be a tree with a depth specified by the user. This depth depends on the results of the decision tree. With the dataset chosen, a lesser or greater depth may be required.

This implementation will be done using "if" conditions. The inputs of this algorithm will be the gas resistance and the heater step of a sensor. The heater step corresponds to the temperature at which the sensor took the gas resistance value.

SVM

The SVM algorithm used will be a classifier. It will have as input the gas resistance value and the heater step. These algorithms are not scaled invariant, so it is recommended to scale the data. One way to do this is to normalize them to have zero mean and variance one [1].

The input data will need to be passed through a function to be normalized. The normalization function is represented as:

$$z = \frac{(x - \mu)}{s} \tag{14}$$

where μ is the mean of the training samples and s is the standard deviation of the training samples. The data to be scaled is the variable x.

Standardization of a dataset is a common requirement for many machine learning estimators. These can show an unusual behavior if the individual features do not look like normally distributed data [1].

Neural Network

The neural network to be developed will be a regressor. A prediction of the target gas concentration will be obtained. Thus the output will be a value and not a class as in previous examples. Its design took into account these inputs:

- Gas Resistance
- Time
- Temperature Heater Step

The interaction between the different inputs is crucial in the network design. It will be necessary to check the linearity or non-linearity of the network inputs as well as the existence of different interaction terms. To understand the terms of interaction and the linearity of inputs, an example is given.

Considering, as input in a given neural network, the variables X and Z, the graphs shown in figure 68 were drawn. The Y-axis represents the neural network's output, and the X-axis represents the X variable. Graphs of the output as a function of X, for each possible Z, are plotted.

Thus, with the graphs obtained, it is possible to analyze the linearity and interaction terms of the variables.

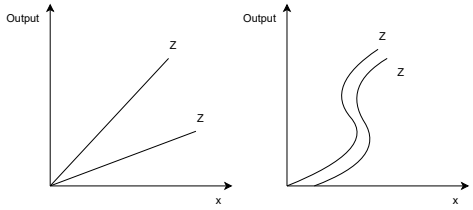


Figure 68: Variables linearity and terms of interaction

Taking into account the graph 68 on the left, the linearity of the variable X is verified. However, the variables have an interaction term since the curves are not parallel. On the right graph, it is verified the non-linearity of the variable. However, this does not have an interaction term.

Given the two graphs, the neural networks would result in the following networks shown in figure 69.

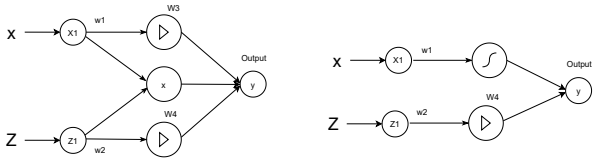


Figure 69: Neural network example

It is possible to verify the interaction term between the two input variables in the left network and the non-existence of the interaction term in the right network. It also verified the non-linearity of the variable x, which is represented as a sigmoid activation function in the network.

Like SVMs, these algorithms need to scale their values. The form used in neural networks is an alternative to the one used before.

The transformation method is called "MinMaxScaler". It is an estimator that scales and translates each feature individually such that it is in the given range on the training set, for example, between zero and one [66].

The transformation is presented on the equation

$$\begin{aligned} X\_std &= (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0)) \\ X\_scaled &= X\_std * (max - min) + min \end{aligned} \tag{15}$$

where,

min and max is the feature range

$X.min(axis = 0)$ and $X.max(axis = 0)$ are the minimum and maximum values of the feature.

On this case the feature range min and max are 0 and 1 respectively.

Therefore, when the neural network is designed, its inputs' linearity and interaction terms will be studied, taking into account the notions presented. The inputs will be scaled using the "MinMaxScaler".

3.2.5 Use Case

As a way to explore the AI algorithms in a real context, a program will be developed in which the neural network is implemented, together with the BME688 sensor. The idea will be to test the life of the chosen battery in a real operating environment.

An example of the use case is shown in figure 70. The sensor is intended to run for three minutes, as an example. This time depends on the time that the neural network is trained. At the end of the three minutes, the gas concentration is calculated. After that, the MCU is put in a low-power mode taking full advantage of its low-power tools. The microcontroller will be in a low power mode until one hour has expired. At the end of this time, it wakes up, and the cycle repeats.

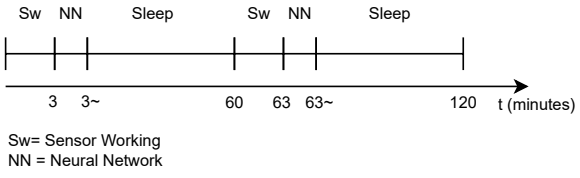


Figure 70: Use case time example

Figure 71 represents a state chart of the problem. The start configurations are done, and the sensor is placed in parallel mode. The system is in idle, with the sensor running during this period. Alarm b is activated after 3 minutes. This time is the time set by the neural network training. The sensor's data is

saved, the neural network is executed, and the microcontroller goes to low-power mode. A new alarm wakes up the microcontroller, and the cycle repeats.

The microcontroller's real-time clock (RTC) gives information regarding the current time.

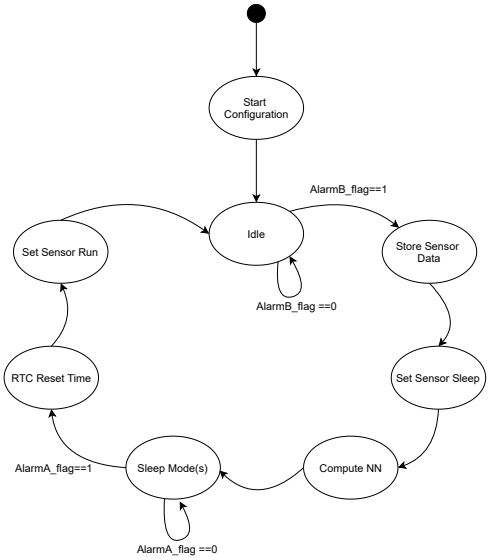


Figure 71: Use case state chart

In this example, three power modes will be used: sleep mode, low-power sleep mode, and stop mode. By default, the microcontroller is in Run mode after the system or power reset [16]. Three power modes will be studied for battery life purposes. Other modes present lower consumption, although those modes don't preserve the SRAM and register contents.

4 Implementation

The implementation phase will satisfy the system requirements. In this chapter, a review of what was done at the hardware and software levels is made.

At the hardware level, the schematics were implemented. The layout of the new board has been made.

At the software level, the flowcharts that serve to the implementation of the code to obtain a dataset of the target gas are presented. With the obtained dataset the artificial intelligence algorithms were implemented.

4.1 Hardware

The design that was adopted to develop the schematic is hierarchical. In this type of design, it is possible to observe the hierarchies and links in the sheet symbols.

Figure 72 shows the hierarchical design of the developed board.

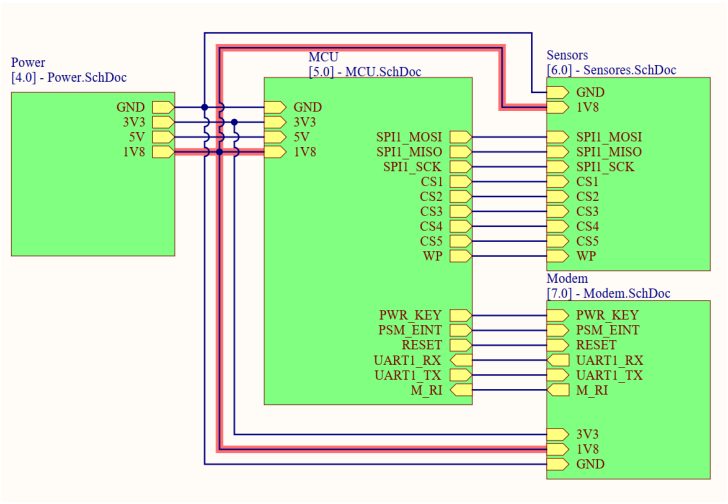


Figure 72: Hierarchical design

4.1.1 Schematic phase

A sheet symbol was developed for the board power circuit, for the MCU, for the sensors, and for the modem. On each sheet symbol the circuits developed on the system specification section (3.1) were implemented.

4.1.2 Layout phase

The components were placed on the board at this implementation stage, and all their connections were traced. Before moving forward, some questions are raised, and some trade-offs are made. The first is the number of layers to use. Two or four layers could be used. 2 layers have been chosen. The reason is the manufacturing price, as a 2-layer board is cheaper. However, the board will present larger dimensions. Choosing a 4-layer board would result in a decrease in size but an increase in price. Another choice is to place components on only one side of the board. This will reduce the price per board developed.

As a 2 layer board was chosen a ground polygon was placed on the top and bottom of the board. The whole board is filled with ground. Rules were created for some components. Some cover the clearance between the ground polygon created and the traces and components placed.

Vias are used to connect the polygon on the bottom and the top of the board. The signal vias are 0.6mm in diameter and 0.3mm in hole size. The power vias have a diameter of 2mm and a hole size of 1mm.

The via stitching technique was used. This joins all the copper areas between the top and bottom layers. Via stitching creates a strong vertical connection and helps to maintain a low impedance and short return loops. It was ensured that there was no dead cropper on the board, having all components grounded.

Wherever possible, teardrops were used for the connections. These present several advantages, such as tolerance in drilling and increased connection reliability. This is due to the increased copper used in the connection.

As for the tracks and connections, thermal relief could be used. This technique facilitates the soldering process. However, it may affect the integrity of the signal. It was chosen to preserve the integrity of the signal, so thermal relief was not used on the connections and vias.

Figure 73 shows the layers' stack used. It is possible to see all the layers, the material used, and the thickness used in each layer.

#	Name	Material	Type	Weight	Thickness	Dk	Df
	Top Overlay		Overlay				
	Top Solder	Solder Resist	Solder Mask		0.4mil	3.5	
1	TOP	CF-004	Signal	1oz	1.378mil		
	Dielectric 1	Core-039	Core		59.055mil	4.8	0.02
2	BOTTOM	CF-004	Signal	1oz	1.378mil		
	Bottom Solder	Solder Resist	Solder Mask		0.4mil	3.5	
	Bottom Overlay		Overlay				

Figure 73: Layers stack manager

Clearance layout standards

The distance between lines and polygons on the PCB should be large enough so that there is no short circuit. Thus, the distance will depend on the applied voltage.

The distance between the shield polygon and the ground polygon for the USB and SIM Card must be analyzed. This will depend on the applied voltage and the withstand voltage that the device will be subjected to.

The USB input and the SIM Card may be subjected to 2kV in the ESD test [27]. Therefore, it is essential to check what the standards require to obtain the required spacing. The 796 standard specified that the withstanding voltage between two conductors must be equal to 40V/mil [79]. Since the connection will be subject to 2kV, the spacing between the two polygons is 50mil. A rule has been created that the spacing between the two conductors must be a minimum of 50mil.

Oscillator design

For the crystal layout, recommendations and guidelines were followed [51]. Using a guard line connected to GND all around the crystal was recommended. The use of symmetry between the oscillator capacitors, a few test points, and not passing signal lines beside the crystal were also recommendations [51]. As such, a guard line with ground paths was made all around the crystal. The ground plane was also isolated from the crystal. The result is shown in figure 74.

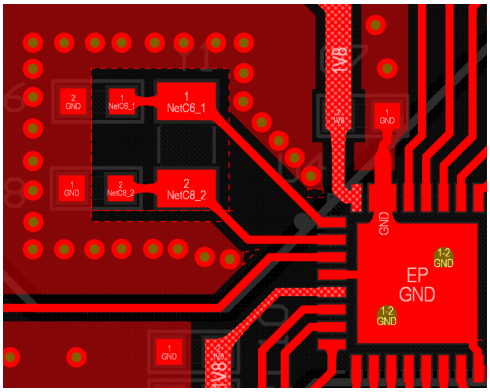


Figure 74: Crystal layout

Antenna design

Some precautions were taken for the antenna design circuit, having followed the manufacturer's guidelines. The GND pins adjacent to the RF lines have not been placed as thermal relief, being completely connected to the ground. The RF trace distance was kept as short as possible. Ground vias were added around the antenna and the trace to improve RF performance. The distance of the vias was at least twice the width of the RF trace. The RF lines were moved away from any interference and no traces were placed on adjacent layers.

For the distance of the trace care has been taken. The length of a trace is determined by:

$$v = f * \lambda \tag{16}$$

The frequency range at which the modem will communicate is between 699MHz and 2200MHz [55]. As such, the wavelength will be:

$$\lambda = \frac{v}{2200MHz} \tag{17}$$

The speed at which a charge flows through copper equals 2/3 the speed of light. Since the speed of light is equal to $(3 * 10^8)$, the speed of the waves through the copper will be $(2 * 10^8)$. Therefore the wavelength will be:

$$\begin{aligned} \lambda &= \frac{2 * 10^8}{2.2 * 10^9} \\ \lambda &= 0.090m \\ \lambda &= 9cm \end{aligned} \tag{18}$$

So the wavelength will be equal to nine centimeters. The whole trace from modem to antenna should not exceed this value. If the trace exceeds this value, it may result in communication problems. At the bottom of the PCB trace, a wave may be at a different period than at the start of the PCB trace if this trace is large. Thus, the drawn trace did not exceed half of this limit so there are no standing waves. The total length of the trace corresponds to 8.31mm. The antenna layout is shown in figure 75.

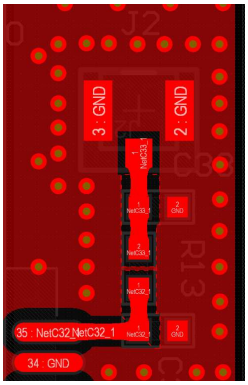
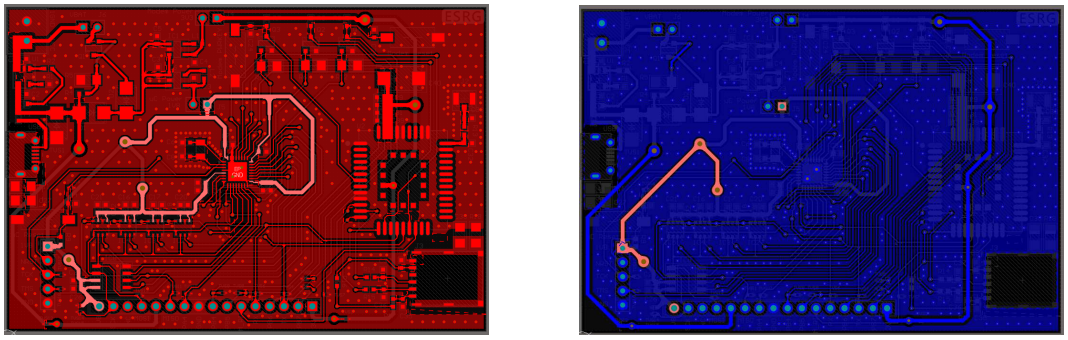


Figure 75: Antenna layout

Layout 2D-3D

The result of the final board layout is shown in figure 76. It is also possible to observe the generated 3D model. Figure 77 shows the 3D model of the developed board.



(a) Board top layout

(b) Board bottom layout

Figure 76: Board top and bottom layout

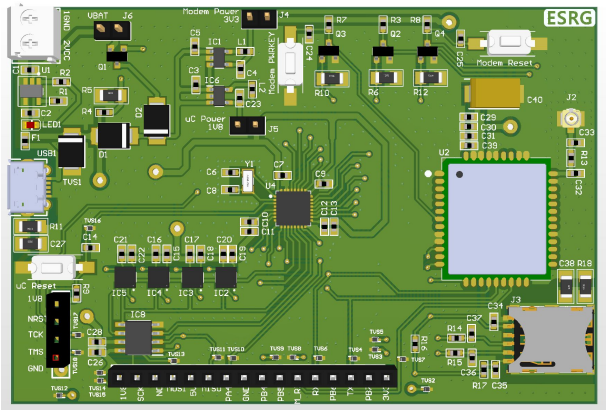


Figure 77: Board 3D model

4.2 Software

This section will discuss the implementation of the software. The software implementation will be presented for the Devkit and for the developed board. Next, the collection of the gas dataset will be discussed. Lastly, the implementation of AI algorithms will be shown.

4.2.1 Sensors comunication

The flowcharts that support the communication between the MCU and the BME688 sensor will be seen in this phase. The functions responsible for reading and writing to the sensor registers were developed. The API invokes these functions through the function pointer whenever it is necessary to write and read from the registers. As previously mentioned, each board has its method of reading and writing the registers, so the flowcharts for each board will be presented.

The flowcharts of the BME688 *DevKit* board will be represented first. Then the flowcharts for the custom board will be demonstrated.

Devkit flowcharts

The sensor configuration flow chart will initially be presented, that is, the comMuxBegin in figure 78. This function must be invoked before any communication with the sensor.

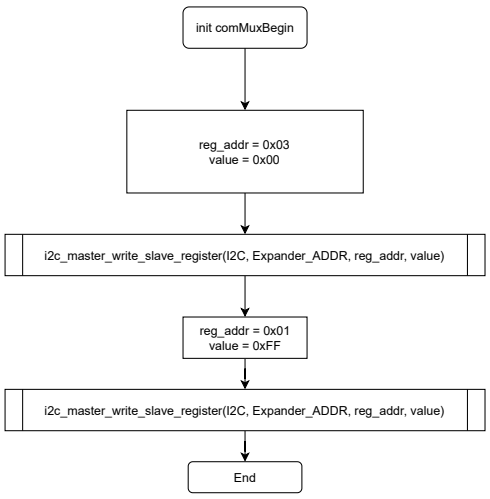
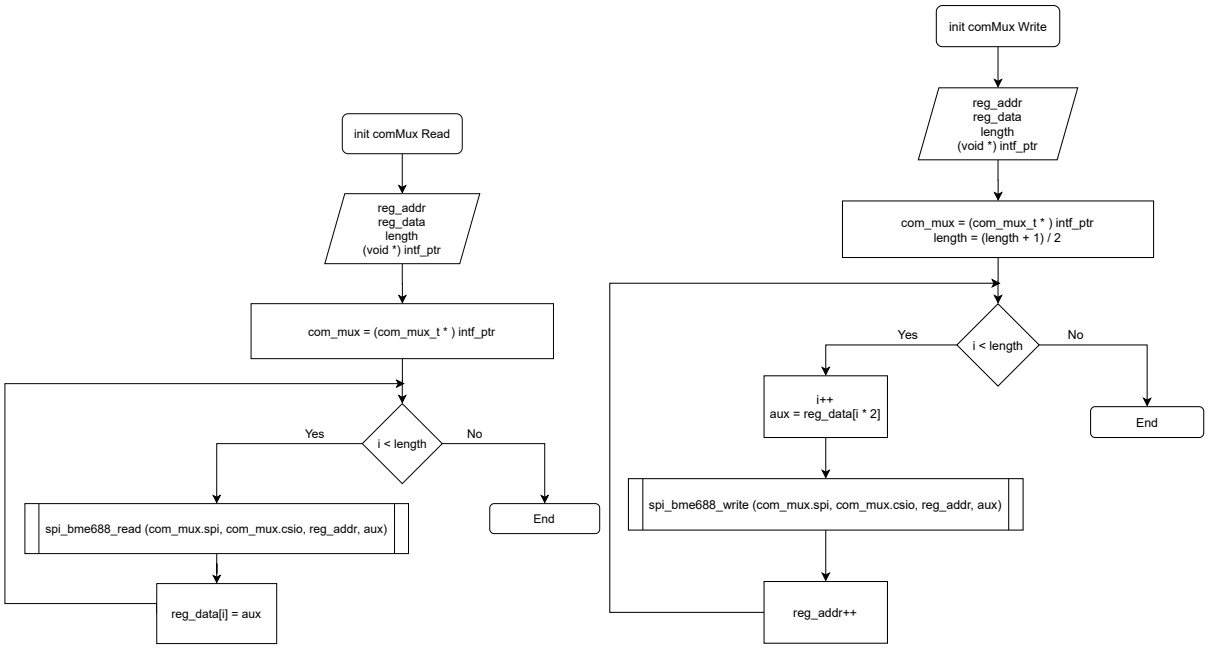


Figure 78: Flowchart *DevKit* communication initialisation

The ”comMuxBegin()” function performs communication with the I/O expander via I2C. Before communications are performed, it is necessary to ensure that all sensor chip selects are set to a high level and to do that is necessary to configure the I/O expander. Therefore, the function initially sets all the pins of the I/O expander to output, through register 0x03. It then uses register 0x01 to set all pins to 1 (value = 0xFF). These pins are the chip select of each sensor.

As previously said, each sensor has a struct associated with it which is "bme68x_dev" (figure 67b on page 66). This holds some information about the sensor when it is initialized, such as the function it will point to when it is necessary to read and write a register ("comMux_write/read") and a void pointer to cast any data. At sensor initialization, the struct "commMux_t", which has the sensor chip select, is stored in the struct "bme68x_dev" through the void pointer. Thus the functions "comMux_write/read" can cast to the data type "comm_mux_t" and access the desired sensor chip select. Thus, the flowcharts of the functions " int8_t commMuxRead(uint8_t reg_addr, uint8_t* reg_data, uint32_t length, void* intf_ptr)" and " int8_t commMuxWrite(uint8_t reg_addr, uint8_t* reg_data, uint32_t length, void* intf_ptr)" are presented on figures 79a and 79b.



(a) Communication read registers (b) Communication write registers

Figure 79: Flowchart *DevKit* write and read registers functions

The read and write function receive as parameters the register address, the register data, the length of the registers, and void pointer to perform the cast. The register data stores the data read from the registers in the read function, and it is used to store the data to be written to the registers, in the write function. The "spi_bme688_(write/read)(com_mux.spi, com_mux.csio, reg_addr, aux)" functions are responsible for setting the chip select of the sensor they are communicating to zero at the beginning of the communication and back to 1 at the end. They use I2C to switch one of the I/O expander pins, being the pin passed to the function through "com_mux.csio".

In short, whenever the API wants to read a value from the sensor registers it jumps to the read function stored in the "bme68x_dev" struct. This function receives as parameters the register to read, the register where the data is to be saved, the number of registers to read, and the void pointer passed by "bme68x_dev". A cast is made with the struct "comm_mux_t" and the "void *" to access the chip

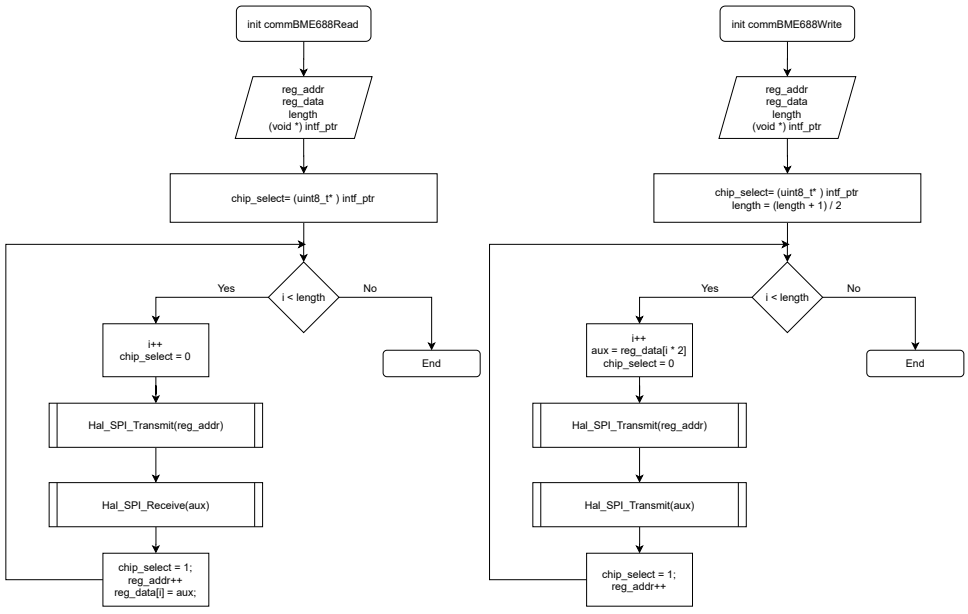
select of that sensor. The data is read via SPI. To start communication, the chip select line is set low through I2C by I/O Expander. At the end of each communication is set to high again. This is done in the function "spi_bme688_read(comm_mux->comm_mux_intf->spi, comm_mux->cs_io, reg_addr, &aux)". Finally, the data is saved in the variable "reg_data", and the program flow returns to the API. The same happens whenever it is necessary to write to the sensor registers being used the write function.

Custom board flowcharts

The flowcharts developed for the custom board are presented next. In this, the implementation is easier since no I/O expander is used. Once again, the API defined a "bme68x_dev" structure for each sensor, which has a pointer to read and write function and a void pointer. This struct is used in all API functions, and whenever a read is requested, this void pointer will be used to know which sensor it is due to the chip select. So in the read and write functions, the registers that are being accessed will be from the correct sensor.

Before any communication, it is important to set all chip selects to high as a precaution. The board presents 4 sensors and so, the chip select pins are PA8, PA9, PA10 and PA11, as presented in figure 82 (page 82).

The functions "commBME688Write(uint8_t reg_addr, const uint8_t *reg_data, uint32_t length, void *intf_ptr)" and "commBME688Read(uint8_t reg_addr, const uint8_t *reg_data, uint32_t length, void *intf_ptr)" are the functions wich the function pointer jumps to. The flowchart of these functions is presented in figure 80.



(a) Communication read registers (b) Communication write registers

Figure 80: Flowchart custom board communication BME688 write and read registers

In the initialization of each sensor, an integer containing the chip select value of each sensor was passed to the void pointer of the "bme68x_dev" struct. So in the read and write functions is only made a cast between an integer and the void pointer, having immediate access to the chip select. Therefore, to communicate with the registers, the function will first set the chip select of the respective sensor to 0 and then transmit the desired address via SPI. If the function is a read function, this address will be the address from which the data is to be read. If it is a write function, the address will be where the data will be saved.

In the case of the read function, the next step is to receive the data and end the communication with the chip select set to 1. In the case of the write function, the data will be transmitted to the desired address. Finally, the communication is terminated, with the chip select set to 1.

4.2.2 Save sensor data SD card

The API can read and write the registers of each sensor with the above functions. As such, it is now possible to configure the sensor with the desired heating profile and the time of each cycle. Then is possible to read the registers' temperature, pressure, humidity, and gas resistance values. The values will be saved in an SD card for future study and data analysis.

Therefore, flowcharts that will serve to implement the code to save the data on an SD card, were developed. In figure 81, it is possible to see the developed flowchart.

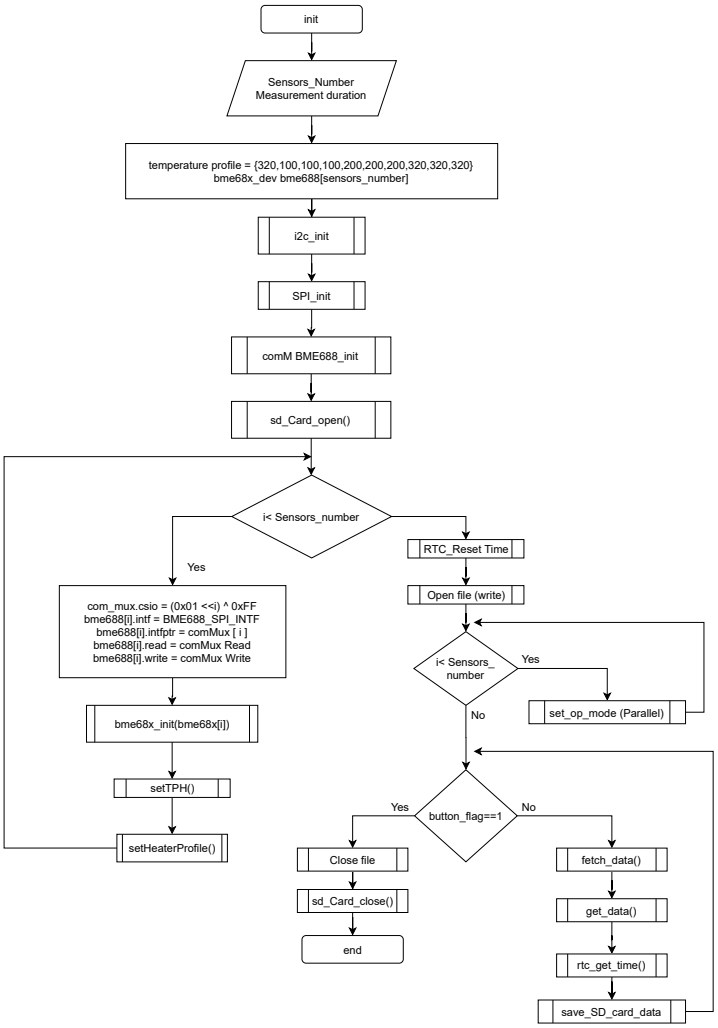


Figure 81: Flowchart SD card store data

The flowchart applies to the *DevKit* board and the custom board. The only difference in the custom board flowchart is that it doesn't use I2C. Therefore the initialization function of I2C is not necessary.

Initially, the number of sensors is configured, which varies according to the board used. Then the time of each cycle for measuring temperature, pressure, humidity, and gas is configured.

The sensor is configured for the desired temperature and for the number of cycles it will perform for each step. This is the heater profile. The heater profile used is the one mentioned throughout the dissertation (chapter system specification). This was recommended by the manufacturer and so used. The struct "bme68x_dev" is then defined for the number of sensors used.

The settings are made for each sensor and written to their registers. Then the RTC is reset. A write file is opened on the SD card, and finally, the sensor is started in parallel mode. All sensors will be scanned, and all sensor data will be saved. In addition to the sensor data, the collection time of the sensors is saved.

The data is saved until a push button is pressed. The file is closed when pressed, and the SD card's data is saved.

Custom board pinout

As previously said, the developed board does not have a slot for an SD card, and so external hardware was developed to place one. It will communicate via SPI. As such, the generated project is presented in figure 82.

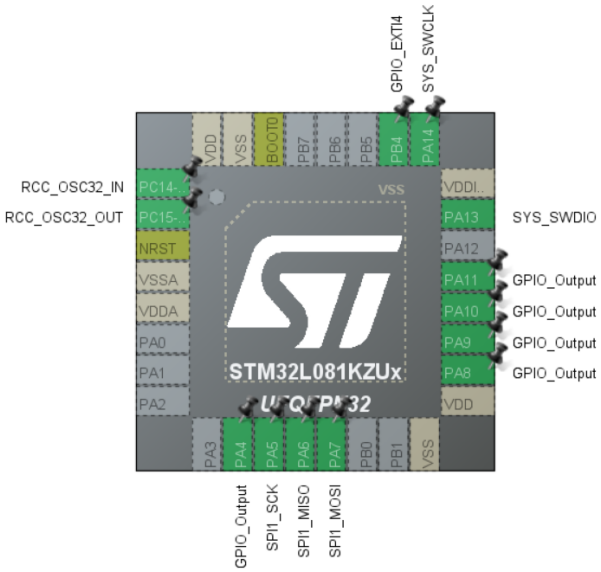


Figure 82: STM32M0+ CUBE pinout

The pins PA8 through PA11 are the chips select for the BME688 sensors. The PB4 pin is a pin configured for the external interrupt. This is the pin used for the button, and the data is saved on the SD card when pressed. PA4 is the pin corresponding to the select chip on the SD card.

4.2.3 Dataset

This section presents the setup and methodology for obtaining the dataset from the BME688 sensor for different gases. To perform classification or regression of different gases, a dataset was obtained for the training of the AI algorithms. As such, and in collaboration with the International Iberian Nanotechnology Laboratory (INL), the sensor was exposed to different concentrations of two gases. The gases were hydrogen and carbon dioxide. Using the *DevKit*, the sensors' data was saved on the SD card and later analyzed.

The board's sensors were subjected to different concentrations of the two types of gases. For this purpose, the sensor was placed inside a box. This box kept the gases inside without leaking. A circuit with software-controlled valves allowed the amount of gas entering the box to be controlled. Each valve had a maximum value of 200 standard cubic centimeters per minute (sccm). The setup is depicted in figure 83.

Figure 83: Set-up INL

For safety reasons, the DevKit was powered using a battery. By connecting the battery, the sensors would start working, and the data would be recorded on the SD Card. However, the sensor was only exposed to the gas after closing the box and controlling the valves. For this reason, the first minute of data logging will not be taken into account. The same was done at the last minute. In summary, the sensor stays inside the box for seven minutes and is only exposed to the gas for five minutes. After seven minutes, the set-up is disassembled. The values present in the SD card are saved, and the sensor is placed again at a different concentration.

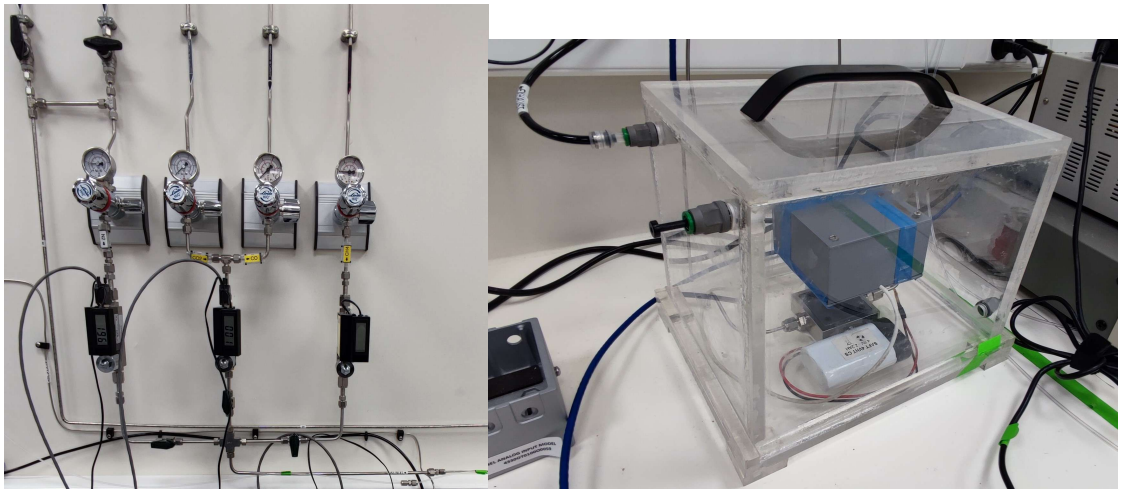
Table 16 presents the concentrations that the sensor is subjected to. The concentrations are calculated considering the maximum value of the valves, which is 200sccm. These valves control the gas composition, one with CO_2 or H_2 gas and another with N_2. For the CO_2/H_2 valve, and for a maximum of 200sccm, the composition in the box is equal to 4.1% gas or 41000ppm. For a value of 180sccm for the CO_2/H_2 valve and a value of 20sccm for N_2, the concentration would be 3.69%. Thus, the calculations are:

$$\begin{aligned} &200sccm = 41000ppm \\ &Valve\, CO_2/H_2 = \frac{(200 * Desired\, Concentration)}{41000} \\ &Valve\, N_2 = 200 - Valve\, CO_2/H_2 \end{aligned} \tag{19}$$

Table 16: Desired concentration

Desired Concentration	Valves N_2 (sccm)	Valves CO_2/H_2 (sccm)
41000	0	200
36900	20	180
32800	40	160
28700	60	140
24600	80	120
20500	100	100
16400	120	80
12300	140	60
8200	160	40
4100	180	20

Figure 84a represents the valves that control the gas flow and figure 84b represents the two boxes where the sensor is located. The grey box contains the sensor. The gas enters through a pipe placed below the box. The transparent acrylic box also has an exhaust pipe and prevents the escape of the gases into the laboratory.



(a) Set-up Valves (b) Set-up Boxes

Figure 84: Gas control set-up and boxes set-up

The results obtained will be presented. Initially, the sensors' data will be presented in normal air. After, the data obtained for hydrogen and then for carbon dioxide will be demonstrated. For each gas, the results from different sensors will be analyzed.

Normal air

The results obtained from sensor 1 and sensor 2 are shown in figure 85 and 86 respectively. The x-axis shows the test time, and the y-axis shows the gas resistance value.

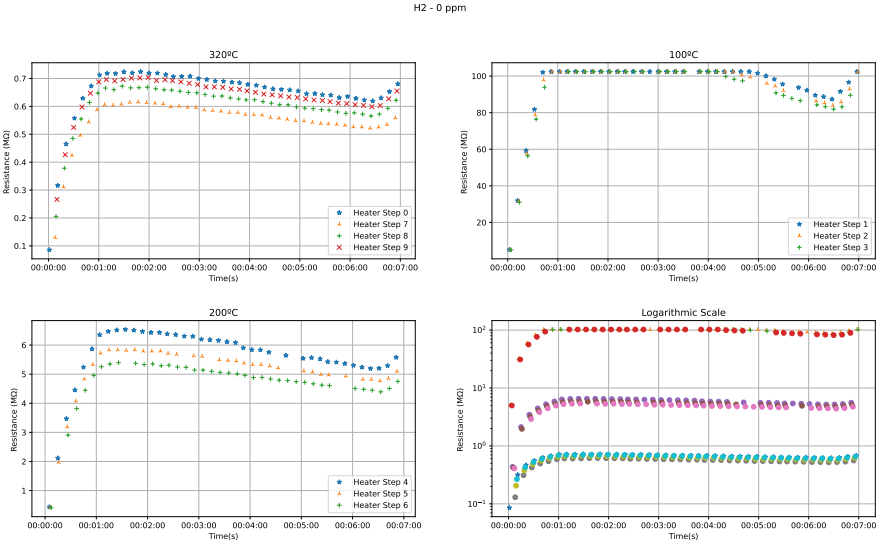


Figure 85: Sensor 1 normal air

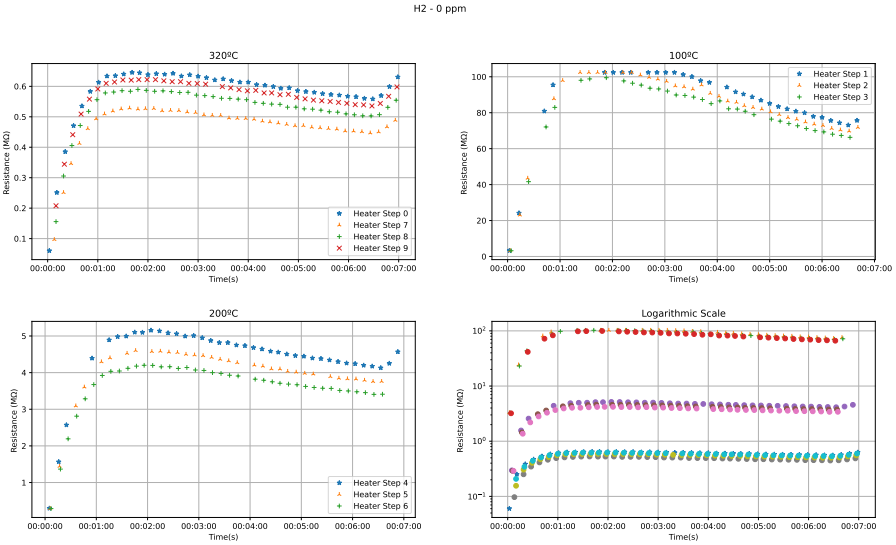


Figure 86: Sensor 2 normal air

By analyzing the results, it is possible to conclude that, despite the sensors being in the same environment, they present different responses. This is due to the physical properties of each sensor. The manufacture of each sensor also influences this response since each layer of tin dioxide ends up not being millimetrically equal.

The artificial intelligence algorithms will be trained with the values of a particular sensor. For these algorithms to work well, the values passed to them should be from the sensor for which they were trained. This is because each sensor behaves differently. Thus, the algorithms will not work properly if they are implemented on a board whose sensor was not the one used to collect the dataset and train the algorithm.

The *DevKit* board was used to obtain the sensor datasets. The artificial intelligence algorithms will be trained with one of those sensors. These algorithms will be implemented on the developed board. However, they will not work properly because the sensors on the developed board are different.

For the algorithms to work properly, it would be necessary to obtain a new dataset from one of the sensors on the developed board. The algorithms should be trained again with the data obtained from these sensors. However, collecting a new dataset for the developed board was not possible. As such, the sensor output of the developed board will not be used, and each algorithm will be tested with values taken from the dataset of the *DevKit* board.

Hydrogen

The results obtained for hydrogen gas are shown in the figure 87 and 88. An application developed in *Python* language was used to read out the gathered data file and produce the graphics. The data presented in figure 87 refer to only one sensor on the board, and the data obtained are for a concentration of 4100ppm. On figure 88 the data refer to the same sensor, with the concentration of 41000ppm.

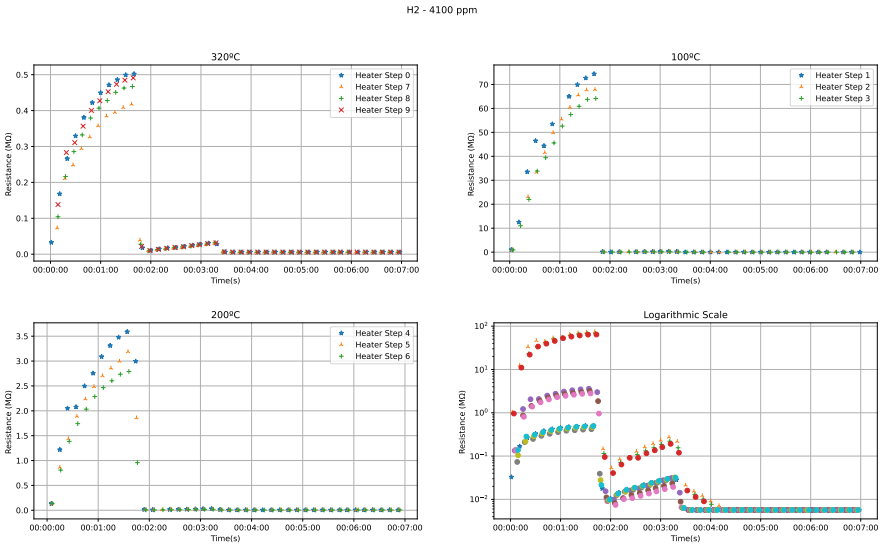


Figure 87: Hydrogen sensor 1 concentration 4100ppm

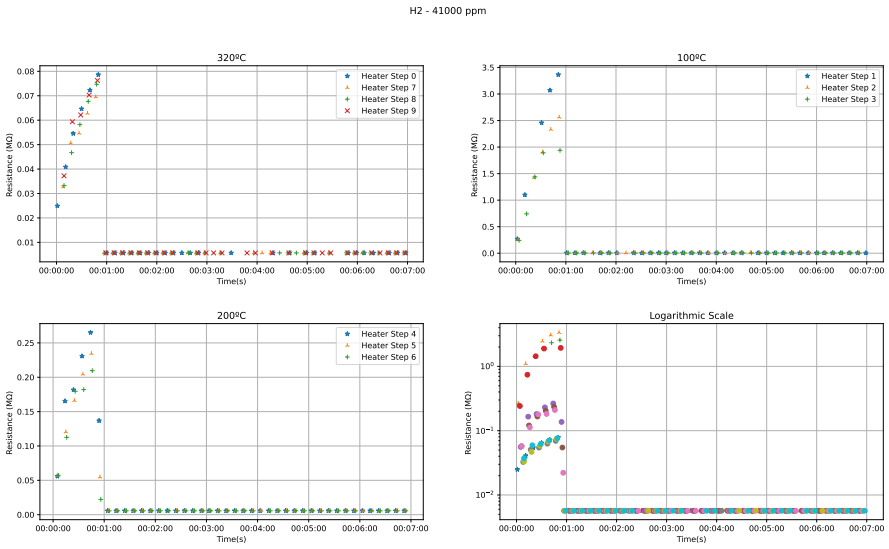


Figure 88: Hydrogen sensor 1 concentration 41000ppm

In the graphs, it is possible to observe the results for each heater step placed in the sensor. The x-axis of the graphs corresponds to the test time, and the y-axis is the gas resistance value given by the sensor.

The first conclusion is that this sensor has a saturation even for a low value of H_2 concentration. The intermediate concentrations were checked, and the same happened. The sensor saturates for a low concentration of H_2 and remains saturated at higher concentrations. Other sensors were checked. The result for another sensor is shown in figure 89. The concentration is 4100ppm.

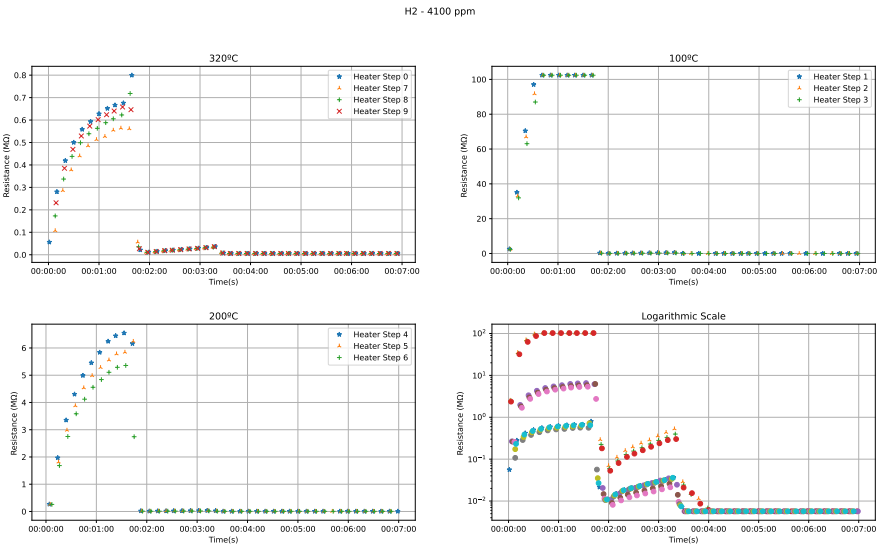


Figure 89: Hydrogen sensor 2 concentration 4100ppm

It is concluded that all the sensors saturate in the presence of H_2 above 4100ppm. With this, the sensor will serve only as an early warning for this gas. It is not possible to test concentrations below the

indicated concentration due to the setup.

Carbon dioxide

The results obtained for carbon dioxide are presented below. The figure 90 shows the sensor behavior for the 4100ppm concentration. The graphs obtained again show on the y-axis the sensor resistance value and the x-axis the time of the test.

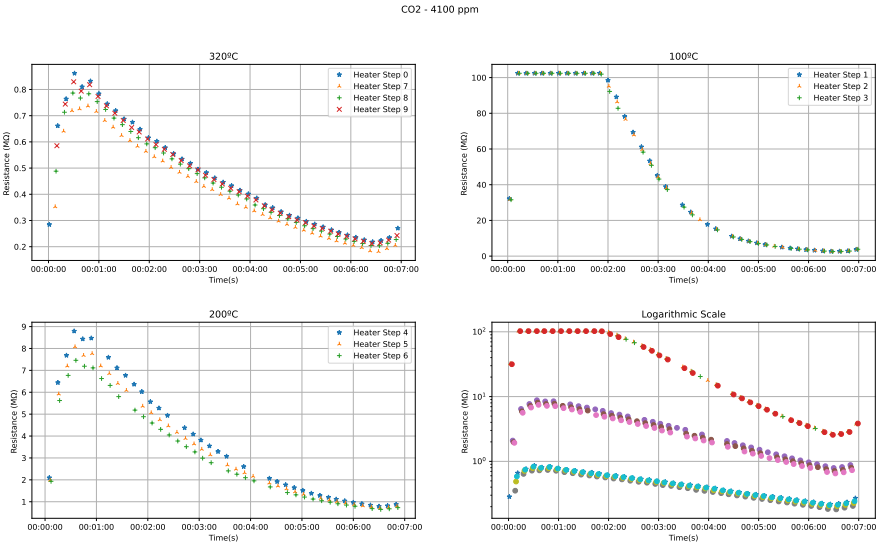


Figure 90: Carbon dioxide sensor 1 concentration 4100ppm

Figure 91 represents the concentration of 12300ppm and figure 92 represents the concentration of 41000ppm, or 4.1%. The sensor is the same as in figure 92.

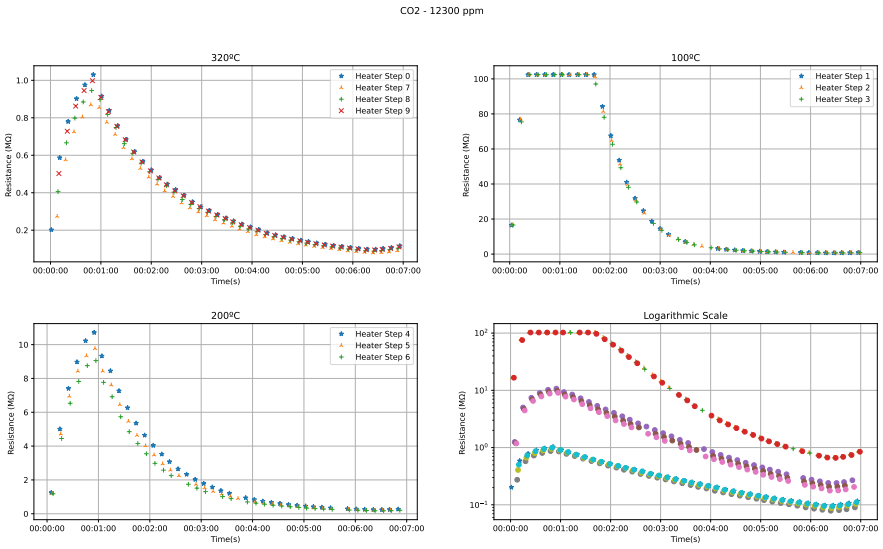


Figure 91: Carbon dioxide sensor 1 concentration 12300ppm

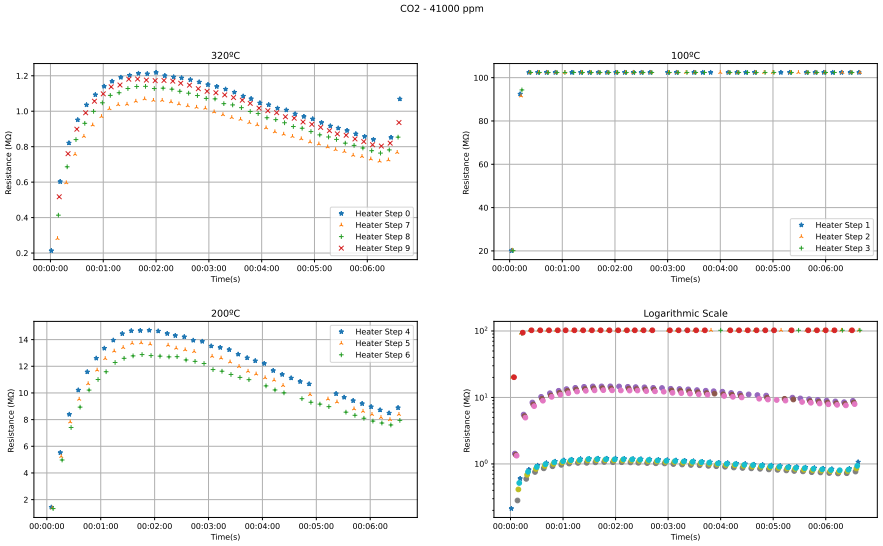


Figure 92: Carbon dioxide sensor 1 concentration 41000ppm

Therefore, given the behavior of the sensor for the different concentrations, this gas will be the target gas. The implemented artificial intelligence algorithms aim at carbon dioxide detection. The main reason is reflected in the sensor response to this gas.

4.2.4 Decision tree

The implemented decision tree is a classifier. It will perform the classification between normal air and a concentration of 41000ppm, or 4.1% CO_2.

The depth of the decision tree was 5, and the features are the gas resistance and the temperature at which this value was taken. This temperature will be one of the heater profile, that is, 100°C, 200°C, or 320°C.

The decision tree training had as output the tree represented in the figure 93.

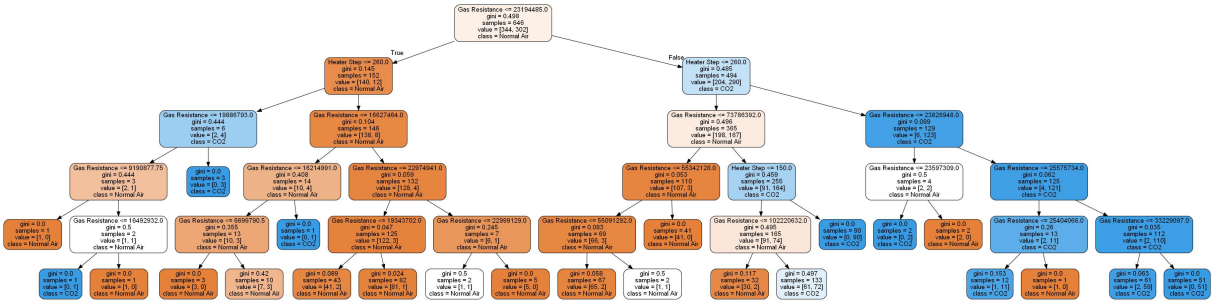


Figure 93: Decision tree output

The decision tree was implemented using if's. The whole tree is traversed until the impurity of the nodes is low. If a node with a low impurity was reached, it would not be necessary to reach the leaf with a depth equal to five, being the classification done. As the classification will be between normal air and CO_2, it was defined an "enum Class" and a variable external to the module, the "enum Class classDT". The function that implements the decision tree is called "DecisionTreeClassifier". It is of the "enum Class" type and has the gas resistance and temperature as input. The output will be a variable of the type "enum Class". So it will have as output, the class CO_2 or the class normal air. The external variable "classDT" will have the value returned by the function. The declaration of the enum and the function that implements the decision tree are presented in listing 1.

```
enum Class {
  Normal_air,
  CO2
};

extern enum Class classDT;

enum Class DecisionTreeClassifier( float gas_resistance, uint16_t heater_step);
```

Listing 1: Decision tree implementation

4.2.5 SVM

The SVM implemented is a classifier. It will perform the classification between normal air and carbon dioxide. The dataset used was 41000ppm.

Since this dataset is quite complicated, and since the data is not separable, the solution created was to use the kernel trick. As previously mentioned, this is a mathematical technique in which the same results are obtained as when polynomial features are added without actually adding them [24].

The kernel trick function is the polynomial function [1]. This function is represented by:

$$(\gamma < x, x' > +r)^d \tag{20}$$

where d is the degree of the polynomial function, r is coefficient 0 and γ is the parameter "gama" or C hyperparameter.

All these parameters are set in the training of the algorithm. The C hyperparameter achieves a balance between instance violations. The instances are separated, some being instances of class CO_2 and others being instances of normal air. The separation can have violations, i.e., instances of normal air being on the CO_2 side and vice versa. As previously said, the low C hyperparameter leads to a larger street with more violations. With a high C, there are fewer margin violations but a smaller margin. The degree controls the separation of instances. The higher the degree, the better the results of the separation. However, a high degree can be overfitting the model. A low degree can cause underfitting. The hyperparameter coef0 controls how much the model is influenced by high-degree polynomials versus low-degree polynomials [24].

A degree equal to 2, a coef0 of 0.02, and a C equal to 5 were defined for this classification.

To implement the SVM using a mathematical formulation, it is necessary to use variables generated by training the algorithm. These variables are, "dual_coef", "support_vectors" and "intercept_b" [1]. The mathematical formulation is:

$$\sum_{i \epsilon SV} y_i \alpha_i K(x_i, x) + b, \tag{21}$$

where the product $y_i \alpha_i$ is the "dual_coef_", the "support_vectors_" is x_i, K the kernel function, x the features which in this case will be standardized gas resistance and temperature, and b the "intercept_" [1].

The training output variables are represented in listing 2.

```
1 /* Variables from SVM Algorithm */
2 const float SupportVectorTemp[SVM_size] =
    ↪ {-1.36403,-1.36403,-1.36403,-1.36403,-1.36403,-1.36403,...,-1.36403};
3 const float SupportVectorRes[SVM_size] =
    ↪ {-0.76835,-0.77683,-0.09522,-0.1264,0.54108,0.46173,-1.07753,...,1.08201};
4 const float DualCoef[SVM_size] = {-5.0,-5.0,-3.17748,-5.0,-5.0,-5.0,-5.0,-5.0,...,5.0};
5 const float intercept= -0.16967394;
```

Listing 2: SVM output variables implementation

The implementation of the kernel trick function is represented in the listing 3.

```
/* Scalled inputs */
float kernel_function(float SupportVectorTemp, float SupportVectorRes, float gas_resistance,
    ↪ float heater_step)
{
  float dp = 0;
  dp = (SupportVectorTemp * heater_step) + (SupportVectorRes * gas_resistance);
  dp = ((0.5 * dp) + 0.02) * ((0.5 * dp) + 0.02); //Kernel function coef0=0.02, C=5, degree=2
  return dp;
}
```

Listing 3: SVM kernel trick function implementation

The function that performs the normalization of the data is presented in listing 4. The function was explained previously, as well as its inputs.

```
#define meanTemp 223.993
#define stdTemp 90.902
#define meanGas 61178501.5414
#define stdGas 38097197.93
/* Produce the feature scale for the gas resistance and heater step */
void svm_feature_scale(float gas_resistance, uint16_t heater_step, float *z)
{
  /* Z[]   index = 0 for Heater_Step, index=1 for Gas_Resistance */
  /* Standardization of Values */
  z[0] = (heater_step - meanTemp)/ stdTemp;
  z[1] = (gas_resistance - meanGas)/ stdGas;
}
```

Listing 4: SVM normalization function implementation

Thus, the function capable of performing the classification was developed. This function, similarly to the decision tree, returns a variable of the type "enum Class". The function first scales the features. Next, it performs the mathematical formulation of the kernel trick. According to the result obtained, it determines the class to which the features belong. Listing 5 implements the function.

```
enum Class SVMClassifier( float gas_resistance, uint16_t heater_step)
{
  float result = 0;
  float z[2] = {0,0};
  uint16_t i = 0;
  /*   Feature Scale  z[0]= Temp, z[1]=Gas_Res*/
  svm_feature_scale(gas_resistance,heater_step, z);
  for (i=0; i<SVM_size; i++)
  {
```

```
    result = result + (DualCoef[i] * (kernel_function(SupportVectorTemp[i],SupportVectorRes[i
    ↪ ], z[1], z[0])));
  }
  result = result + intercept;
  if (result < 0)
    classSVM = Normal_air;
  if (result > 0)
    classSVM = CO2;
  return classSVM;
}
```

Listing 5: SVM function implementation

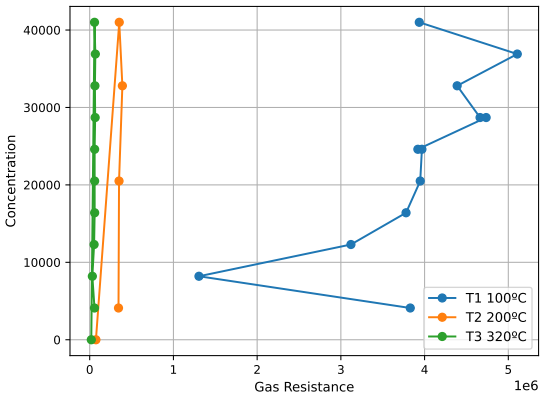
4.2.6 Neural network

The neural network was developed considering the graphs recorded after collecting CO_2 data at different concentrations. With the data collection, it was possible to ascertain the linearity or non-linearity of the neural network inputs and the existence of the different interaction terms.

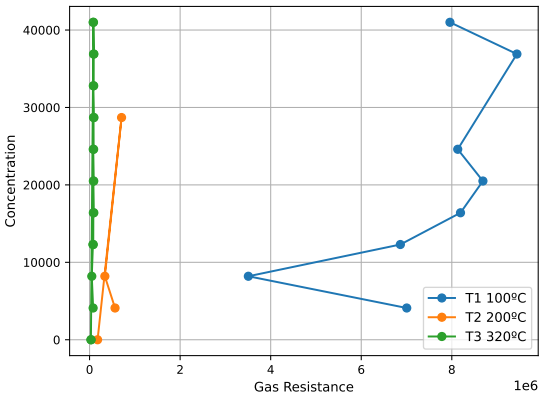
As previously mentioned, the neural network is a regressor. The inputs are gas resistance, temperature, and time. The output is the concentration of CO_2 present in the air.

The neural network was trained with the sensor 0 dataset from the *DevKit* board. To study the non-linearity of the variables, the graphs of the sensor 0 results when placed in different concentrations of CO_2 will be analyzed.

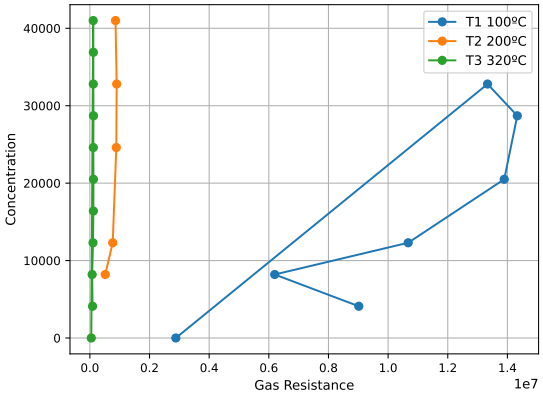
Therefore, the first graphs considered data from one of the sensors. The concentration was plotted as a function of resistance for different temperatures (100°C, 200°C, 320°C), resulting in 3 graphs. Different graphs were made for different data collection times. For example, three graphs for a time of fewer than 5 seconds, three graphs for a time between 10 and 15 seconds, etc.



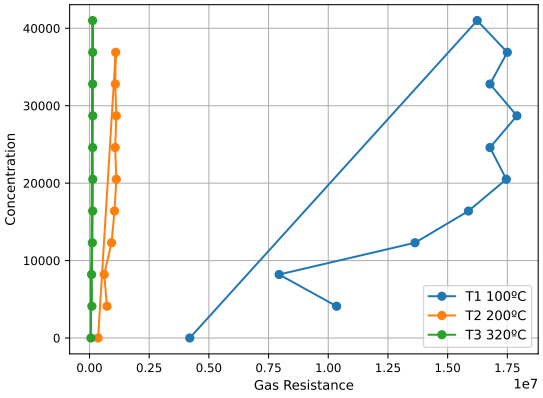
(a) Time < 5 seconds



(b) Time]10 , 15] seconds



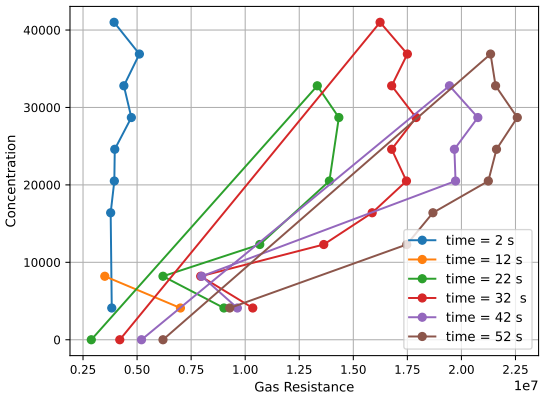
(c) Time]20 , 25] seconds



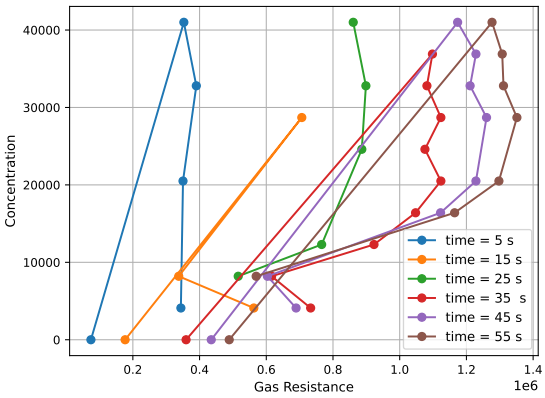
(d) Time]30 , 35] seconds

Figure 94: Concentration vs gas resistance as a function of temperature

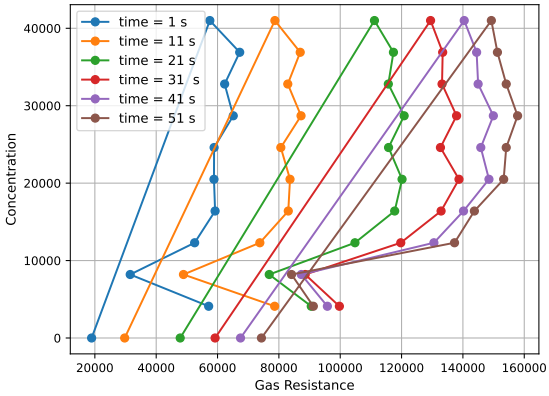
Thus, taking into account the graphs 94 obtained, it is possible to conclude that the curves are non-linear, so the value of gas resistance will also be non-linear. The curves are not parallel, so there will be an interaction term between resistance and temperature.



(a) Temperature 100°C



(b) Temperature 200°C



(c) Temperature 320°C

Figure 95: Concentration vs gas resistance as a function of time

Thus, analyzing the results obtained in the graphs of figure 95, it is possible to conclude that the curves are non-linear. Once again is concluded that gas resistance is non-linear. None of the curves is parallel, which leads to the conclusion that there will be an interaction term between gas resistance and time.

For the graphs obtained in figure 94, none of the curves is parallel independently of the analyzed time. In the graphs of figure 95, none of the curves is parallel independently of the temperature variation. Therefore, it will be possible to conclude that there will be a triple interaction term between the gas resistance, the time, and the temperature, besides the existing double interaction terms.

Therefore, the design developed for the neural network is shown in figure 96.

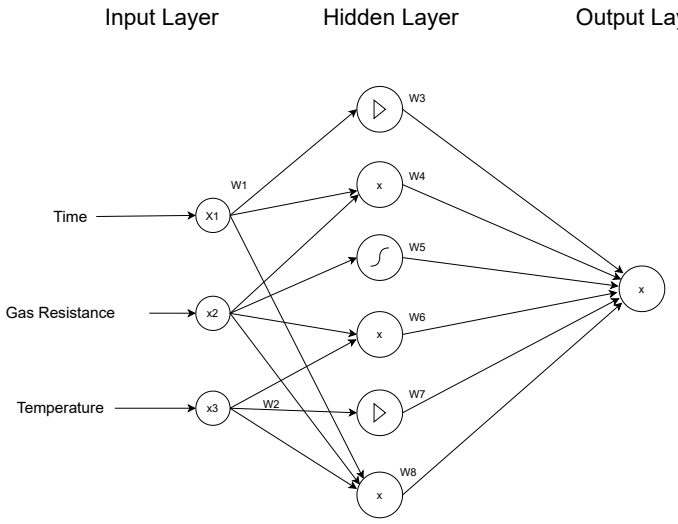


Figure 96: Neural network design

The neural network was trained between 120 and 360 seconds, and the results for the different weights are presented in table 17. The maximum and minimum input values of the neural net are given in table 18.

Table 17: Neural network weights

Weights	W1	0.985594
	W2	1.74085
	W3	-0.0963083
	W4	1.21031
	W5	0.462069
	W6	16.6142
	W7	0.135626
	W8	10.2562

Table 18: Neural network inputs/outputs

		Max. Value	Min Value	Units
Input /Output Values	Time	360	120	seconds
	Gas Resistance	25191620	12364.7607	Ohms
	Temperature	320	100	°C
	Concentration	41000	0	ppm

MinMax scaler transformation

The function that implements the transformation of each feature, "MinMaxScaller" is presented in listing 6. A function that performs the inverse operation of the transformation of the values was developed. This function will place the network output between the values 0 and 41000ppm.

```
1 float MinMaxScaler(float var, enum var_type var_type)
```

```
{
    float x_std = 0;
    float x_scalled = 0;
    float x_min = 0, x_max = 0;
    switch( var_type )
    {
      case Time:
          x_std = (var - timestamp_min) / (timestamp_max - timestamp_min);
          break;
      case Temperature:
          x_std = (var - heater_step_min) / (heater_step_max - heater_step_min);
          break;
      case GasResistance:
          x_std = (var - gas_resistance_min) / (gas_resistance_max - gas_resistance_min);
          break;
      case Concentration:
          x_std = (var - concentration_min) / (concentration_max - concentration_min);
          break;
      default:
          break;
    }
     x_scalled = x_std * (1-0) + 0;
     return x_scalled;
}
```

Listing 6: NN MinMax scaller implementation functions

LUT

An approach for implementing the network would be given by the equation 22. The sigmoid function $\phi(z)$ is given by the equation 23.

$$y = (X1 * W_1 * w_3) + (X1 * X2 * W_4) + (\phi(X2) * W_5) + (X2 * X3 * W_6) + (X3 * W_2 * W_7) + (X3 * X2 * X1 * W_8) \tag{22}$$

$$\phi(z) = \frac{1}{1 + exp(-z)} \tag{23}$$

However, this approach may prove to be bad for some reasons. One is the execution time. It would take too long to run the sigmoid function of a given value every time the neural network needs to be executed. Since the sigmoid function is an exponential function and time-consuming to execute, it will increase the power consumption every time it is executed, being that another reason.

One strategy adopted will be the implementation of LUT (look-up table). With this strategy, a thread-off between memory resources (Flash and SRAM), processing time, and consumption will be done [46] [33].

A lookup table is a table stored in the microcontroller and used when it is pretended to reduce the calculations to be carried out. The lookup table can be used to fill in certain values of the sigmoid function to reduce the number of calculations performed at runtime. At the beginning of the program, all the values are calculated to fill the table. This table can also be filled via hard-coded. Then a lookup of pretended values is performed.

This choice brings some error to the algorithm since not all points of the sigmoid function will be in the table.

The table is filled with values between $400k\Omega$ and $22M\Omega$. These values are the maximum and minimum values recorded for sensor 0 for a heating temperature of 100°C in the presence of the different CO_2 concentrations. The table presents increments of about 144000 ohms, so the table has about 150 positions.

These values are scaled by the method previously mentioned, so they will only be used to demonstrate the process of filling in the table. Thus, the table will have four columns. The first column will be responsible for storing the value of the gas resistance. The second column stores the sigmoid function value. The third one stores the regression value. The last one stores the constant value to add to the linear regression. The third column will have the following regression:

$$\frac{y_i - y_{i-1}}{x_i - x_{i-1}} * R \tag{24}$$

where,

x_i is the resistance gas value immediately above the measured one

x_{i-1} is the value of gas resistance preceding x_i

y_i is the sigmoid value of the gas resistance value immediately above the measured one

y_{i-1} is the value of sigmoid preceding y_i.

An example is presented in the table 19 with simulated values to better understand the LUT.

$x = M\Omega$	y	$\frac{y_i-(y_{i-1})}{x_i-(x_{i-1})}$	C
30	0.1	-	-
35	0.2	0.1/5	-0.5
40	0.3	...	...
...			

Table 19: LUT example

$$f(35) = \frac{0.1}{5} * 35 + C \Leftrightarrow$$
$$C = 0.2 - (\frac{0.1}{5} * 35) \tag{25}$$

With this table filled in, assuming that a gas resistance value equal to $32.5M\Omega$ is measured, the LUT procedure will be as follows:

- The value immediately above 32.5 will be found, in this example being the value of 35
- The regression is calculated based on the parameters in columns 3 and 4

$$Ex : 32.5M\Omega$$
$$f(R) = (\frac{y_i - (y_i - 1)}{x_i - (x_i - 1)}) * R + C \tag{26}$$
$$f(R) = \frac{0.1}{5} * 32.5M\Omega + (-0.5)$$

When the measured values are outside the range of the table, that is, above the value in position 0 of the table and above the value in position 150, the value of the sigmoid will be its maximum and minimum value. The following figure 97 shows the maximum and minimum values of the developed table and the sigmoid function.

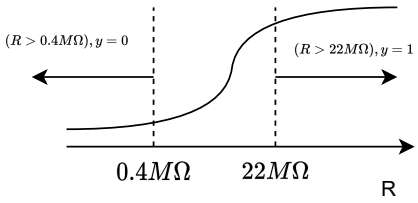


Figure 97: Sigmoid funtion

When the measured resistance gas value is below $0.4M\Omega$, the sigmoid value will be 0. The same happens for the maximum value, $22M\Omega$. When the measured value exceeds this value, y will immediately be 1.

Listing 7 presents the implementation of the sigmoid function and the LUT initialization function.

```
1 float sigmoid(float z)
2 {
3     return (1 / (1 + exp(-z)));
4 }
5 void initLUT()
6 {
7     // Value between 0.4 Mohm and 22 Mohm
8     float minValue = 400000;
```

```
    float maxValue = 22000000;
    float increment = 0.0057189999; //value that increment in the scalled format
    uint8_t i = 0;

    minValue = MinMaxScaler(minValue, GasResistance);
    maxValue = MinMaxScaler(maxValue, GasResistance);

    //Collumns [0= Resistance Value  1= Sigmoid value  2= yi-(yi-1) / xi-(xi-1)  3=C]
    LUT[0][0] = minValue;
    LUT[0][1] = sigmoid(LUT[i][0]);
    LUT[0][2] = 0;
    LUT[0][3] = 0;
    for (i = 1; i < LUT_SIZE; i++)
    {
        LUT[i][0] = LUT[i - 1][0] + increment;
        LUT[i][1] = sigmoid(LUT[i][0]);
        LUT[i][2] = (((LUT[i][1]) - (LUT[i - 1][1])) / (increment));
        LUT[i][3] = (LUT[i][1]) - ((LUT[i][2]) * (LUT[i][0]));
    }
}
```

Listing 7: NN sigmoid and funtion

Listing 8 implements the function to search for a gas resistance value in the LUT.

```
float search_LUT(float scalled_resistance)
{
    uint8_t i = 0;
    float result = 0;
    bool flag = 0;

    // min values of LUT
    if (scalled_resistance <= LUT[0][0])
    {
        flag = 1;
        result = 0.503849;
    }
    //max value of LUT 0.873244
    if (scalled_resistance >= LUT[150][0])
    {
        flag = 1;
        result = 0.873244;
    }

    while (!flag)
```

```
21      {
22          if (LUT[i][0] >= scalled_resistance  )
23          {
24              flag = 1;
25              //regression result
26              result = (LUT[i][2] * scalled_resistance) + LUT[i][3];
27          }
28          i++;
29      }
30      return result;
31  }
```

Listing 8: NN search LUT funtion

Finally, the function that implements the neural network is presented in listing 9. It has as input the network features. This function performs the transformation of each feature. Then it performs the calculation of the neural network, where its weights are used, and the LUT search function is also used. Finally, the result is unscaled.

```
1  float neural_network(uint16_t time, uint16_t Temp, float gas_resistance)
2  {
3      float gas_scaled  = MinMaxScaler(gas_resistance, GasResistance);
4      float time_scaled = MinMaxScaler(time, Time);
5      float Temp_scaled = MinMaxScaler(Temp, Temperature);
6      float regression_scaled = 0;
7
8      regression_scaled = (time_scaled * weights[0] * weights[2]) +
9          (time_scaled * gas_scaled * weights[3]) +
10         (search_LUT(gas_scaled) * weights[4]) +
11         (time_scaled * gas_scaled * Temp_scaled * weights[7]) +
12         (gas_scaled * Temp_scaled * weights[5]) +
13         (Temp_scaled * weights[1] * weights[6]);
14
15     return MinMaxUnScaler(regression_scaled, Concentration);
16 }
```

Listing 9: NN funtion

As already said, two approaches for the implementation of LUTs are possible. The first would be to fill the LUT at runtime, and the second would be to enter the LUT with the values already filled in (hardcoded). This would eliminate the initial time required to fill the LUT and the consumption that may exist during that period. The option chosen is the hardcoded. Thus, the function ”initLUT()” and ”sigmoid(float z)” are implemented externally. An array with 150 positions is generated, which will be entered into the code. The remaining functions continue to be used.

4.2.7 Use case

In this section, the implementation of the use case will be described. The purpose of this code is the evaluation of the duration of a battery in a real context. As such the sensors will be in operation. The algorithm implemented will be the neural network. The sensors will be in operation for 3 minutes. The sensors will be placed in sleep mode at the end of that time. A gas resistance value from the dataset will be introduced, and the neural network will be executed. Then the microcontroller will be placed in one of the energy-saving modes for one hour.

The pin configurations are presented in figure 98.

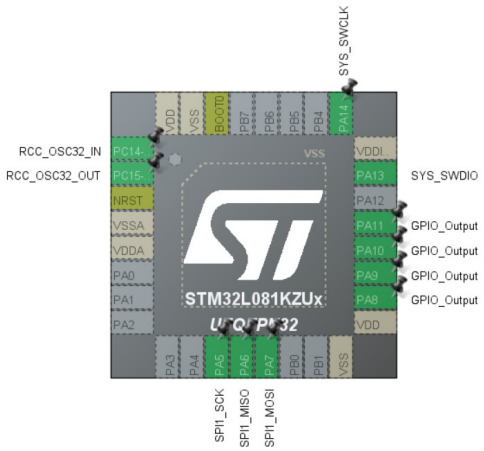


Figure 98: Pin configuration use case

There are several pins in use. The pins for SPI communication, as well as the pins used by the serial wire debug, are used. The external oscillator pins are also selected. The remaining pins correspond to the chip-select pins of each sensor. Although only one of the board's sensors was used, all four chip selects were placed. The default sensors are placed in sleep mode, so it won't be necessary to configure them.

To configure the RTC, it is necessary to select a low-speed external clock in the clock configuration. This selection can be seen in figure 99 and the necessary frequency for the external oscillator of 32.768kHz.

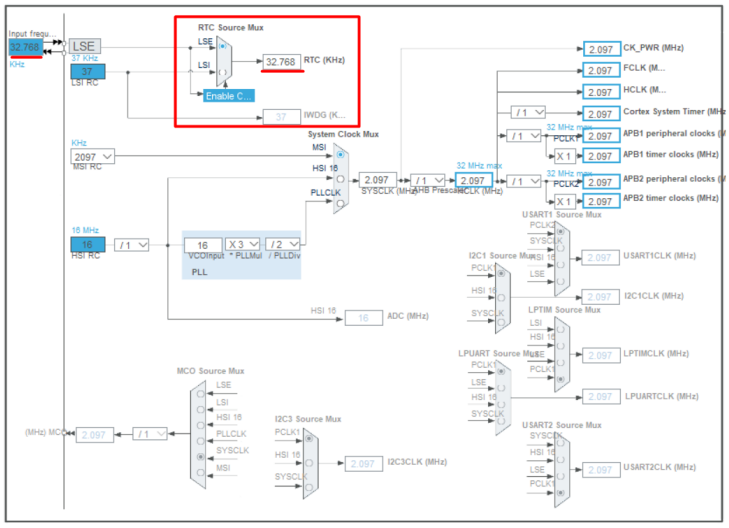


Figure 99: Clock configuration use case

The remaining settings have been made. The SPI full duplex master was selected, and two internal alarms, alarms A and B, were placed in the RTC. Alarms A and B were set for one hour and 3 minutes, respectively. When this time passes, an interruption will be triggered for each alarm.

To enter the different low-power modes, it is necessary to disable/enable some features. First, it is essential to disable the systick interrupt, otherwise, it will wake up the microcontroller whenever it is activated. Next, the microcontroller is placed in the desired mode. With the function *"HAL_PWR_EnterSLEEPMode (PWR_LOWPOWERREGULATOR_ON, PWR_SLEEPENTRY_WFE)"* the microcontroller will go into sleep mode. The first parameter of this function will determine if the microcontroller is put into sleep mode or low-power sleep mode. The second determines if the microcontroller waits for an interrupt or an event to wake it up (macro WFI, wait for interrupt, and macro WFE, wait for event).

To enter in the Stop mode, the function will be *"HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFE)"*, the reasoning being the same.

Figure 100 illustrates the flowchart that represents the flow of the code developed for this use case.

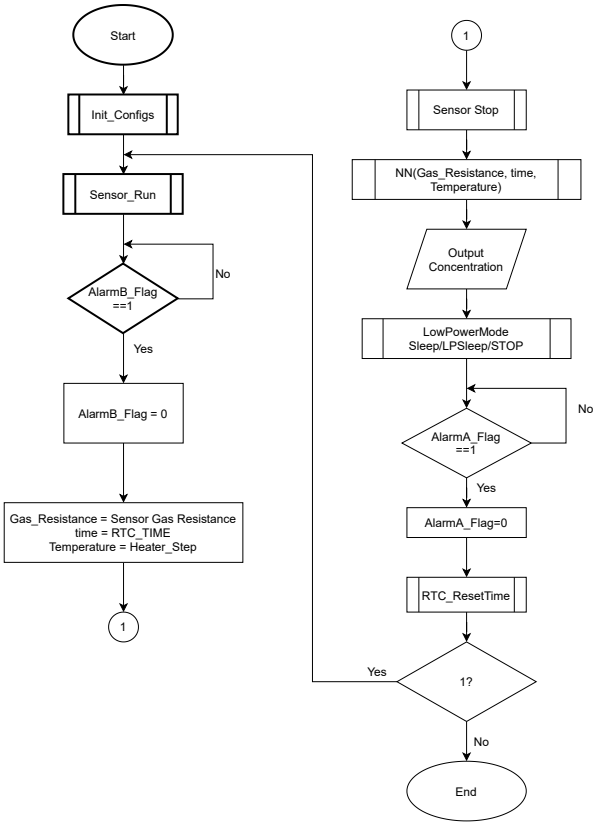


Figure 100: Use case flow chart

First, the settings are made, and the sensor is put in parallel mode. The sensor remains in this mode until alarm B is active. The alarm B flag is activated after the RTC has been interrupted. After being activated, the microcontroller leaves the idle state, clears the flag, and starts the process of reading the sensor registers. However, these values will not be those passed to the neural network, being only a simulation of a real context (the values passed to the neural network are the dataset values). After this process, the neural network is executed. The sensor is placed in sleep mode, and the microcontroller enters one of the low-power modes until alarm A wakes it up.

5 | Tests and results

In this chapter the tests carried out and the results obtained are discussed. Initially, the results of the tests performed on the developed board will be discussed. After that, the tests made on the BME688 sensor will be analyzed in terms of power consumption. The developed algorithms will be tested in terms of duration, time, and power consumption. Finally, the total cost of the board as of spring of 2022 and bill of materials (BoM) will be presented.

Some tools were used to perform the developed tests, namely an oscilloscope and measurement unit. The oscilloscope is presented in figure 101, and the measurement unit is in figure 103.

Oscilloscope

The oscilloscope used is the MDO3012. It presents two analog channels with a bandwidth of 100 MHz and a sample rate of up to 2.5 GS/s. The record length is up to 10M samples and has 16 digital channels [74]. The software for taking screenshots of the oscilloscope, Openchoice desktop, was also downloaded. Using this, it was possible to take printouts of the results obtained.

Figure 101: Oscilloscope Tektronix [74]

The logic analyzer was used to check the SPI frames. The Tektronix P6316 MSO probe was also used. This probe offers two eight channel pods. It is represented in figure 102.

Figure 102: Tektronix P6316 MSO probe [74]

Source/Measure unit

A Keysight B2901Ais a source/measure unit (SMU) that was used to measure the current consumption of the sensors, boards, and algorithms [73]. This, like all other choices, are choices of opportunity on the available hardware at any given time. Figure 103 presents the precision source/ measure unit used.

Figure 103: Keysight precision source [73]

The "QuickIVMeasurement"software was used to collect energy consumption. The SMU has a USB connection, making it possible to send and control data via PC. Through this medium, it is possible to receive the energy consumption data and graph it.

It has 12 current source specifications with different resolutions. The Keysight B2901A supports a 20 us sample rate (50000 points/s) [73]. For all the tests performed, the source unit powered the boards. The positive output and the ground of the SMU were used for that.

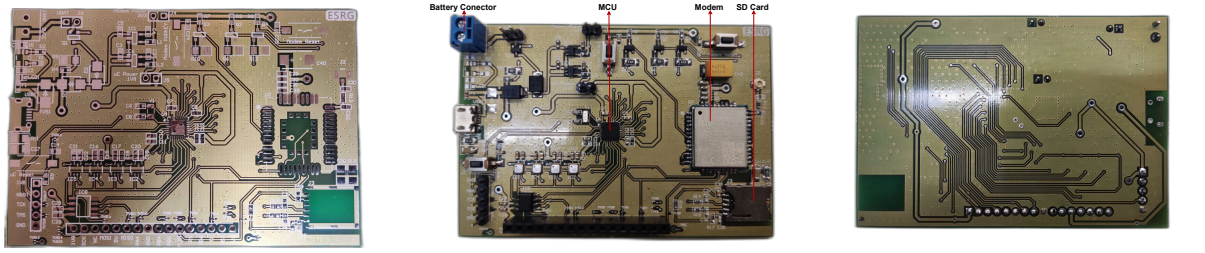
5.1 Developed board

This section will discuss the results of the developed board. The soldering and assembly process will be presented initially. After that, the tests to validate the board will be described.

5.1.1 Assembly and soldering

The board developed before the assembly and soldering process is shown in figure 104a. The board has dimensions of $85.6mm\ x\ 58mm$.

The soldering process was carried out, and the board presented in figures 104b and 104c was obtained. It is possible to distinguish some components from the naked eye. The MCU is in the center, the BC66 modem is on the right side, and the header used to access the pins is below the MCU. On the left side are the BME688 sensors, the USB port, and the battery connector in blue.



(a) Board before soldering (b) Board top after soldering (c) Board bottom after soldering

Figure 104: Board top and bottom before/after soldering

BOM

The total costs of the components making up the board are presented in table 20. The BOM is presented in the appendices section A (page 139).

Table 20: Total costs board developed

Component	Description	Quantity	Unit Price	Total Price
Capacitor		35		4.76 €
Diode Schottky		2	0.55 €	1.1 €
Fuse	2A	1	0.94 €	0.94 €
Header	1x16 Male	1	1.90 €	1.90 €
Header	1x40 Female	1	1.53 €	1.53 €
Voltage Regulator	1V8	1	1.52 €	1.52 €
Voltage Regulator	3V3	1	1.59 €	1.59 €
BME688		4	14.91 €	59.64 €
Battery Connector		1	0.37 €	0.37 €
Nano SIM Card		1	1.81 €	1.81 €
Antenna	Conector	1	1.20 €	1.20 €
Inductor	10uH	2	0.14 €	0.28 €
Mosfet	P-Channel	1	0.51 €	0.51 €
Mosfet	N-Channel	3	0.36 €	1.08 €
Resistor		16	0.518 €	8.29 €
Buttons	Press	3	0.32 €	0.94 €
BC66-NA		1	19.06 €	19.06 €
STM32L081KZU6		1	7.00 €	7,00 €
USB Micro		1	0.31 €	0.31 €
Crystal	32.768kHz		1.22 €	1.22 €
ESD	3.3T5G	17	0.49 €	8.33 €
TVS	Bidirectional	1	0.35 €	0.35 €
2 Layers PCB		1		45 €
Total:				168.71 €

To validate the board, several tests were carried out. The first test was to check the connectivity of all components to the pad and the corresponding signal line. Next, and before connecting any jumper, the correct voltage in the signal pins was verified. It was verified that the microcontroller jumper had a voltage of 1.8V and the modem jumper had 3.3V. After this validation process, it was possible to start programming the microcontroller and implementing the firmware to communicate with the sensors.

The tests carried out for the firmware validation were essentially two. The first one was at the hardware level to validate the communication with the sensor. The next one is at the software level, in which the implementation of saving the sensor data to an SD card is tested.

5.1.2 Sensor communication

The tests performed for the firmware validate the communication with the sensor. In these tests, an oscilloscope and a digital probe are used. It is analyzed the SPI frames exchanged between the microcontroller and the sensor. It is analyzed if the chip selects of each sensor are working and if it is possible to communicate with each sensor.

To test the communication between the sensor and the microcontroller, the chip ID value of a sensor was read. The chip select signal was checked, and the chip ID obtained was verified. The test was then performed in software using debug.

5.1.3 Save sensor data

For final verification of the communication between the microcontroller and sensors, and validation of the code implementation, the board was set to save the sensor data to an SD card. This test was run for 25 minutes. In one phase of the test, it was breathed on the sensors and analyzed if it changed their output. This procedure occurred between minutes 13 and 15 in the test.

The result of the test is shown in figure 105. To visualize the results, only one sensor was used.

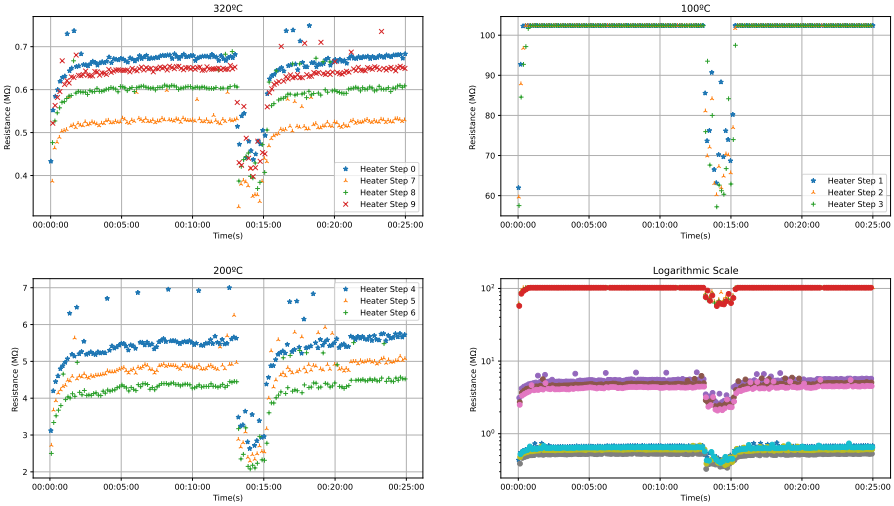


Figure 105: Developed board sensor data

Therefore, it can be concluded that the communication with the sensors works and is functioning properly. The firmware has been implemented correctly. The code for reading and writing data to the SD card was also implemented correctly.

5.2 Sensor BME688

In this section, two tests will be performed on the BME688 sensors.

The first test is carried out to analyze the performance decay of the sensor when it works for approximately one month.

The second test was the acquisition of the power consumption of the BME688 sensor. The two boards were used, the *DevKit* and the developed one. Each board's consumption of a single sensor was analyzed for different heating temperatures and the selected heater profile. Finally, the consumption of each board was compared with all the sensors working.

5.2.1 Stress test

The main purpose of this test is to analyze if the sensor loses its characteristics or goes into malfunctions by using it for one month in a climatic chamber for accelerated aging. During this month, the sensor will sample every hour, but it is working continuously. Samples were taken of temperature, pressure, humidity, and also the measured gas resistance value. The results of the samples taken will be demonstrated, and the data will be analyzed. Lastly, conclusions will be drawn.

The tests were carried out in a laboratory climatic chamber, where it is possible to control its temperature. These tests started on January 26th and ended on March 8th, equivalent to continous around 80 days of service at room temperature (around 1932 hours of service).

$$\begin{aligned} e^{\frac{273.15+70}{273.15+20}} &= 3.22 \\ 25\ days * 3.22 &= 80\ days \end{aligned} \tag{27}$$

These tests were interrupted from the 2nd of February to the 18th of February. This interruption was because the sensor was exposed to extreme temperatures (-20 °C approximately). For this reason, the sensor was removed from the climatic chamber to check if it registered the data efficiently and was still working. The sensor continued to take data, however, during this interval, the data will not be displayed as it is not in the desired temperature range for the study. Then the sensor was inserted again into the climatic chamber, and the data collected by the sensor during that period was removed.

The *DevKit* board was used and only one board sensor was chosen to perform the test. The sensor was configured with the heater profile used in all tests and previously discussed.

The sensor is placed in the climatic chamber to accelerate the aging process. The wear process is accelerated by placing the sensor in an environment under extreme conditions. With this, the fact that the sensor works consecutively during the entire test time combined with extreme conditions can estimate its durability.

Therefore, the results obtained during the climatic chamber are presented in the figures 106 and 107.

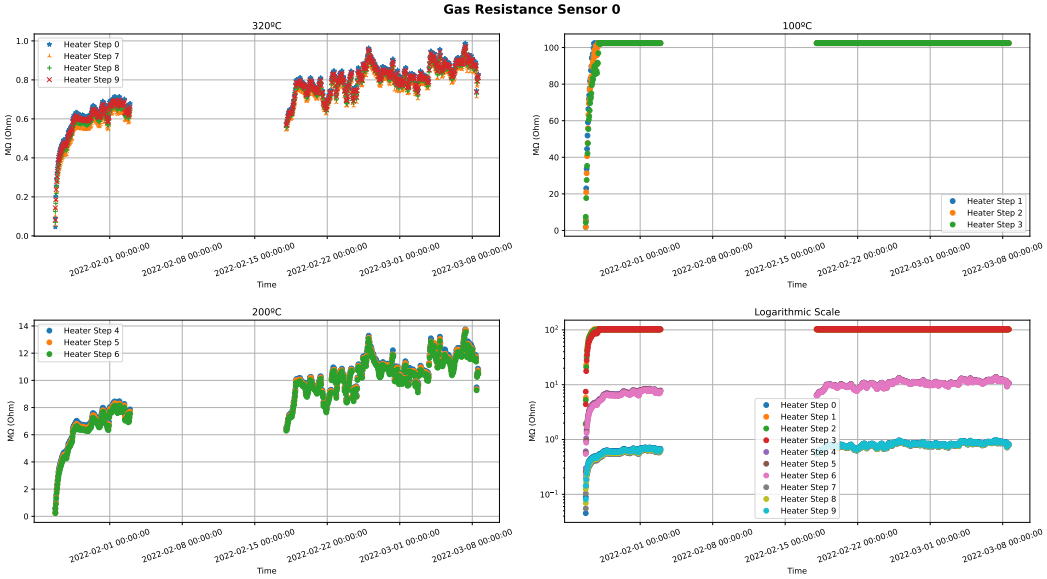


Figure 106: Stress test sensor 0: gas resistance at 320°C, 100°C and 200°C

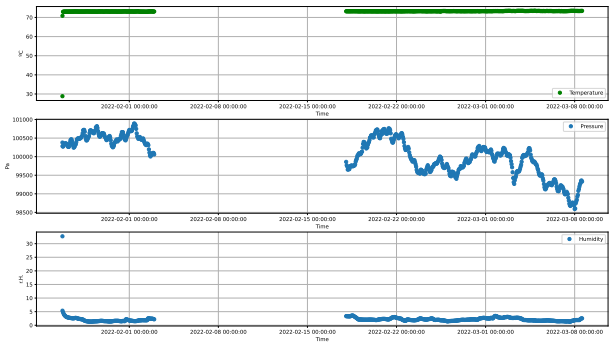


Figure 107: Stress test sensor 0: temperature, pressure, humidity

Figure 106 represents the gas resistance for the different temperatures 100°C, 200°C, and 320°C. Figure 107 represents the temperature, pressure, and humidity measured inside the climatic chamber.

To complete the study, it was done a linear regression and a regression of the transient response of gas resistance of the sensor.

The transient response can be represented by the following equation:

$$R = R_0 + \triangle R(1 - e^{-\frac{t}{\tau}}) \tag{28}$$

The value of R_0 is the initial resistance value on the collected data. The $\triangle R$ is the resistance variance value of the gas when time tends to infinite. The variable τ is the time constant of system and t is the time of the collected data.

The equation is similar to an RC circuit. The dependent variable in the case of the RC circuit is voltage, in this case, the measured resistance value.

The regression uses the least squares method. The least squares method is a mathematical technique that allows determining the best way of fitting a curve on top of a chart of data points. In short, the desire is to minimize the distance between the measured data and the predicted data. As such, the absolute value of the distance is used. In this example, the desired would be:

$$min \sum_i d_i^2 = \sum_i (y_i - f(x_i))^2 \tag{29}$$

$$f(x_i) = R_0 - \triangle R(1 - e^{-\frac{t}{\tau}}) \tag{30}$$

The minimum value of a function is obtained when its derived value equals zero. Since it is desired the minimum value of the distance, the equation (29) is equal to:

$$\begin{cases} \frac{\partial \sum_i (y_i - f(x_i;\triangle R;\tau))^2}{\partial \triangle R} = 0 \\ \frac{\partial \sum_i (y_i - f(x_i;\triangle R;\tau))^2}{\partial \triangle \tau} = 0 \end{cases} \tag{31}$$

From here, is calculated the value of τ and $\triangle R$.

The same reasoning is done with the linear regression with an equation equal to:

$$R = mt + b \tag{32}$$

A *python* script was developed to upload the data and perform the two regressions, calculating the values of τ, $\triangle R$ and m.

Therefore, the gas resistance value for each temperature is presented in figures 108, 109 and 110 with the regressions shown in the graphs, as well as the equations of each regression.

Figure 108: Stress test sensor 0: gas resistance *burn in* at 100°C

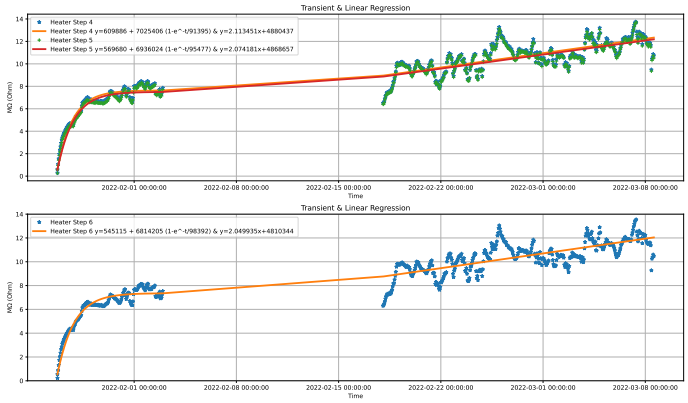


Figure 109: Stress test sensor 0: gas resistance *burn in* at 200°C

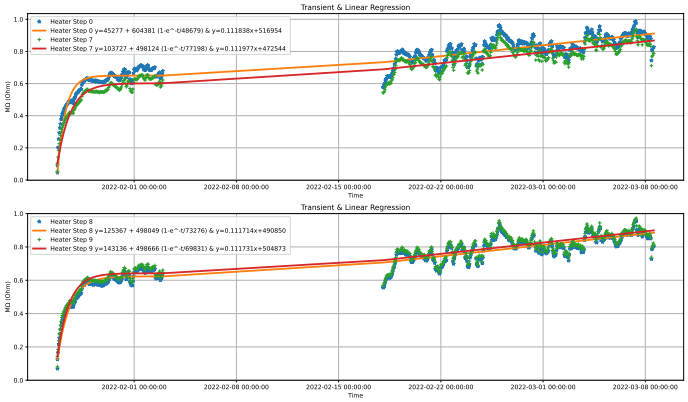


Figure 110: Stress test sensor 0: gas resistance *burn in* at 320°C

The regression is done to study the variation of the values on the sensor and to compute the value of τ and m for the transient regression and linear regression. The value of τ represents the thermal inertia and is, therefore, the time constant of the system. The slope of the linear regression assesses stability. With the slope, it is possible to find out how many ohms the gas resistance varies per unit of time.

To draw conclusions about the data, further tests were carried out on the sensor. It was collected data outside the climatic chamber and at room temperature to compare the gas resistance values. The data collected by the sensor in normal air, outside the climatic chamber, and after the stress test are presented in the figures 111 and 112.

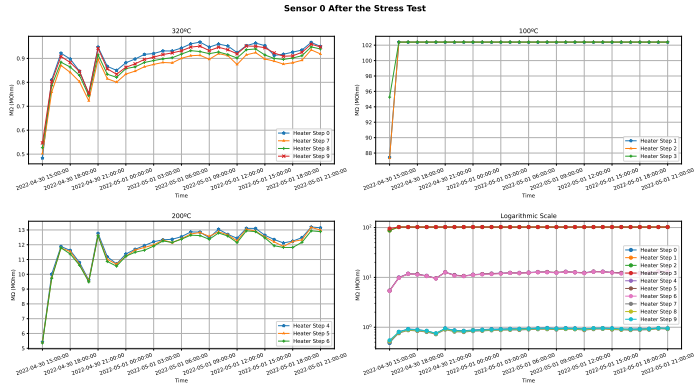


Figure 111: Test sensor 0: gas resistance after climatic chamber

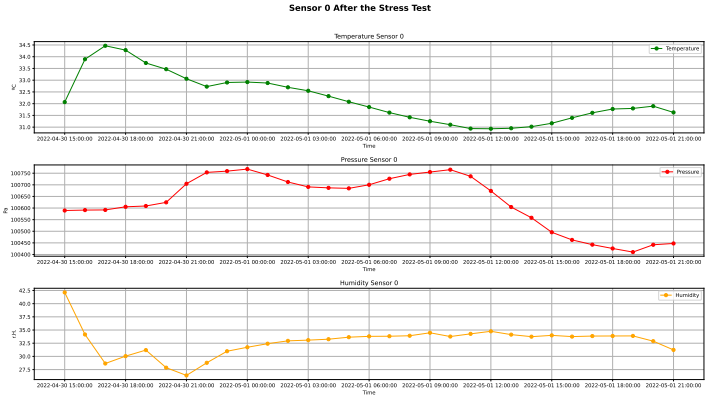


Figure 112: Test sensor 0: temperature, pressure, humidity after climatic chamber

All this having been obtained, the table 21 has been made to help draw some conclusions about the tests made.

The table shows the range of values in which the gas resistance is comprised for the different temperatures. Initially, the range of values for the test performed inside the climatic chamber is shown. Then is shown the range of values for the test performed after the placement of the sensor in the climatic chamber.

Table 21: Results of sensor 0 stress test

Test 25 Days Sensor 0	Resistance Gas Sensor
Burn In	$(3M\Omega \sim 102.4M\Omega)$ at 100°C
	$(0.5M\Omega \sim 14M\Omega)$ at 200°C
	$(0.1M\Omega \sim 0.9M\Omega)$ at 320°C
Test 24 Hours Sensor 0	**Resistance Gas Sensor**
Today	$(88M\Omega \sim 102.4M\Omega)$ at100°C
	$(5M\Omega \sim 13M\Omega)$ at200°C
	$(0.5M\Omega \sim 0.9M\Omega)$ at320°C

Based on the values, it is possible to determine that the synoptic resistance value in the burn in phase equals the synoptic resistance value taken in the test after the climatic chamber.

$$R_{\infty}(Burn\ In) \sim R_{\infty}(Today) \tag{33}$$

It is also possible to conclude that the value of gas resistance in the samples at 100°C is higher than at 200°C, and gas resistance at 200°C is higher than the resistance at 320°C.

$$R_{100°C} > R_{200°C} > R_{320°C} \tag{34}$$

It can be seen that the pressure, humidity, and temperature values in the test after the climatic chamber are within the expected values.

Since the test's purpose was to analyze the loss of the sensor's characteristics or even its malfunction, it is possible to conclude that the sensor is still working properly. The temperature, pressure, and humidity values are correct, and the gas resistance values are the expected ones.

5.2.2 Current consumption

The purpose of this test is to analyze the power consumption of the BME688 sensor for different temperatures and the power consumption of the sensor when its heating process is as described in the heater profile. For that, the measuring unit Keysight B2901A and the software QuickIV Measurement are used. The software collects all the data sent by the unit of measurement and gathers them in a table.

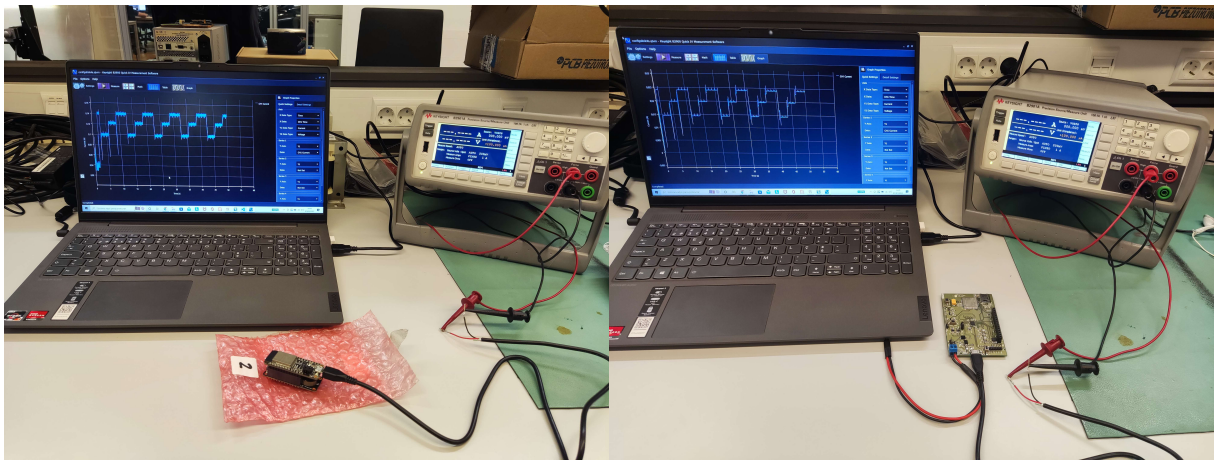
Some Python scripts are developed to compute all the data from the different tests done.

Test conditions

The tests are done first with the BME688 development kit board, with all eight sensors working simultaneously. Several tests were carried out for the sensors. Initially, tests were made on the sensor with fixed temperatures, that is, placing the sensor at 100°C, 200°C, or 320°C. In the first phase, the order in which the temperature on the sensor was placed was relevant, as well as whether the sensor was pre-heated or not. These conditions could influence the measurements, so they were taken into account. Then, some tests were made, being the sensor with the heater profile previously mentioned.

After these tests have been performed, other tests were done. This time the board developed with the Arm Cortex M0+ and with 4 sensors is used.

To compute the consumption of just one sensor, instead of the whole board, the consumption of the board was measured without starting all the sensors. Then the result consumption was averaged. Thus, in all tests, the board's consumption is subtracted from the consumption obtained. This is then divided by the number of sensors on the board so that the consumption of each sensor is obtained. Figure 113 shows the set up used for the consumption measurement.



(a) Setup *DevKit* (b) Setup Arm Cortex M0+ Board

Figure 113: Power consumption setups

The *DevKit* was used for the first test, and the board with eight BME688 sensors was tested. The table 22 presents the tests realized to the board.

Table 22: Tests power consumption *DevKit* Board

Tests	Board temperature	Temperature sequence
1	Cold	100°C-200°C-320°C
2	Cold	320°C-200°C-100°C
3	Warm	100°C-200°C-320°C
4	Warm	320°C-200°C-100°C
5	Cold	100°C-200°C-320°C
6	Warm	320°C-200°C-100°C
7	Warm	100°C-200°C-320°C
8	Warm	320°C-200°C-100°C
9	Cold	320°C-200°C-100°C
10	Cold	200°C-100°C-320°C
11	Warm	200°C-320°C-100°C

In the table, the temperature of the board varies between hot and cold. That means that if the board is cold it's because the sensor has not been heated up in the last 24 hours approximately. The hot designation appears when the sensor has recently undergone a heating process. Normally, tests were made with the board cold, and new tests were made soon afterward when the board was hot. Putting the board cold or hot could bring significant differences in current consumption, hence the development of tests with the board cold or hot.

The temperature sequence indicates how the tests were carried out, i.e., the sensor is placed over a temperature, and the power consumption is measured. Then the sensor is placed for the next temperature, and again measured the power consumption.

For the first test, the sensor is placed first to 100°C, then 200°C, and lastly to 320°C. The result will be 3 measures of consumption for each of the temperatures.

As the last test, the board is subjected to a heating profile equal to the heater profile, and the consumption of each sensor will be measured.

After all these tests, the board was placed with the heater profile used, and the consumption of each sensor was measured.

The tests performed on the sensor BME688 on the board developed are represented in the table 23. A final test was made, placing the sensor with the heater profile and measuring the consumptions in this board.

Table 23: Tests power consumption developed board

Tests	Board temperature	Temperature sequence
1	Cold	100°C-200°C-320°C
2	Cold	320°C-200°C-100°C
3	Warm	200°C-320°C-100°C
4	Warm	200°C-100°C-320°C

Tests results

The tests developed for the BME688 sensor for the *DevKit* board are presented below. As previously stated, the board consumption was subtracted from each consumption obtained, and the resulting consumption was divided by the number of sensors.

Figure 114 shows the medians of each consumption and for each test, which means the median of test 1, for the temperature at 100°C, and so on consecutively.

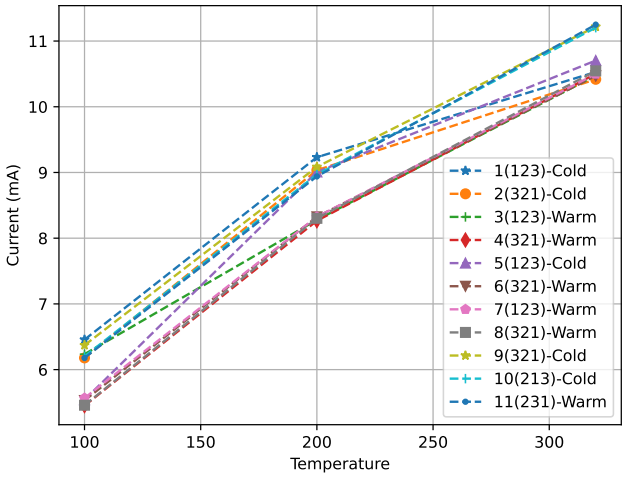


Figure 114: Consumption median board *DevKit*

Three box plots were developed with all the consumptions taken in all the tests for each temperature (100°C, 200°C, 320°C). Consumption is analyzed using boxplots because these graphs make it easier to represent the distribution of the data and compare it.

In figure 115 are represented the boxplots of the power consumption of each sensor on the *DevKit* board.

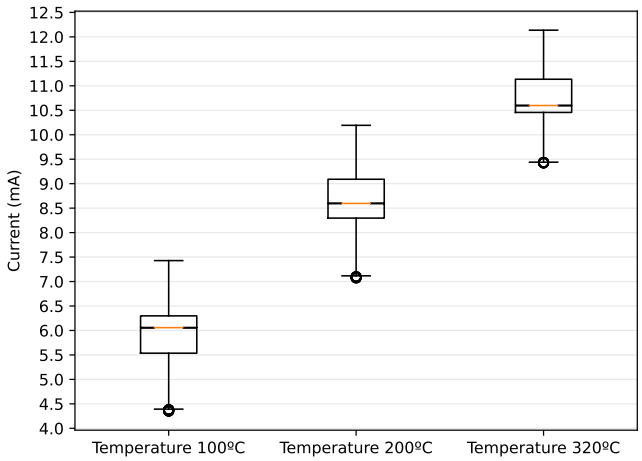


Figure 115: Consumption of one sensor box plot *DevKit*

The result of the power consumption when the sensor is heated with the heater profile is represented in figure 116.

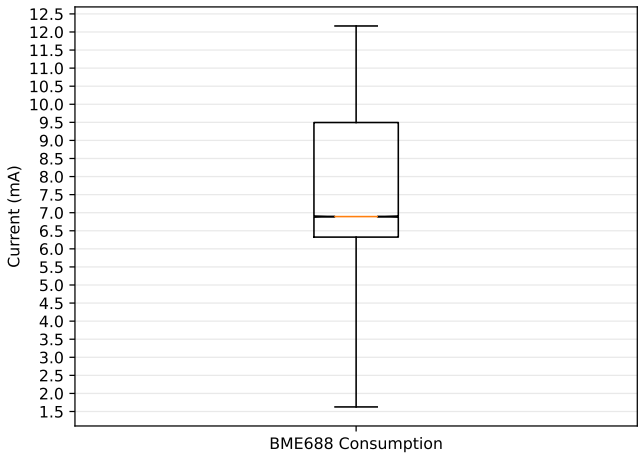


Figure 116: Consumption of one sensor box plot *DevKit* heater profile

With these results, it is possible to conclude that the current for 100°C is always lower than the current necessary for the sensor to operate at 200°C and even lower than the current necessary to operate at

320°C, even if the sensor has already been heated (warm) or not (cold).

$$I_{100°C} < I_{200°C} < I_{320°C} \tag{35}$$

It is also possible to observe the median consumption of the sensor when heated to 100°C, 200°C, and 320°C and compare this consumption to the sensor consumption when using the heater profile.

The sensor BME688 with the board developed is supplied at a voltage lower than the voltage used in the *DevKit* board. The sensor in the developed board is feed with 1.8 volts. On the DevKit, the sensor is powered at 3.3 volts. Therefore the consumption on the developed board could be lower than on the development kit board.

The median of each consumption and for each test are presented in figure 117.

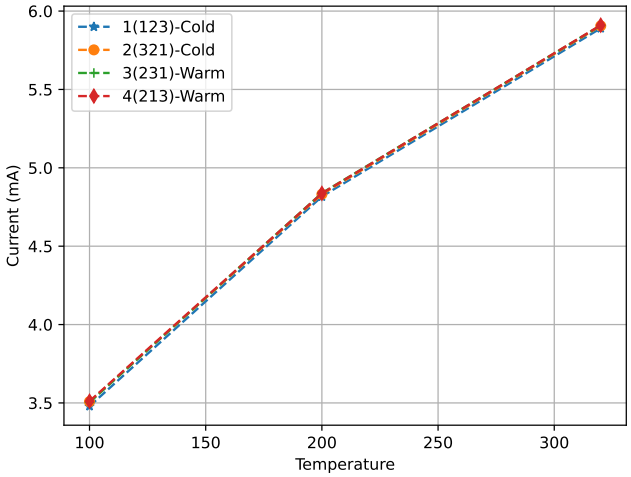


Figure 117: Median consumption of one sensor board developed

Figure 118 represents the power consumption of one sensor on the developed board.

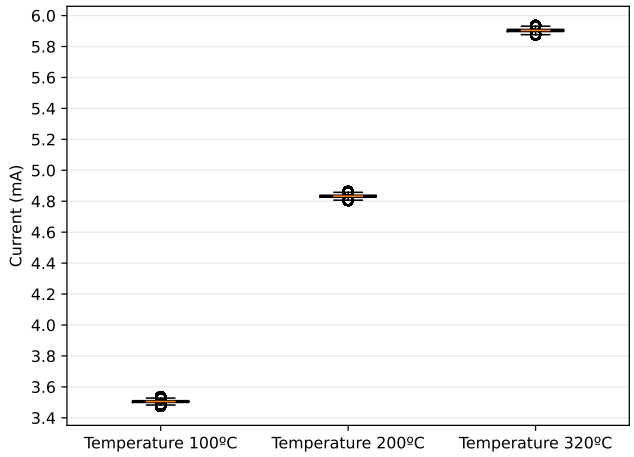


Figure 118: Consumption of one sensor box plot board developed

The power consumption of the sensor on the developed board, when it is heated with the heater profile is shown in figure 119.

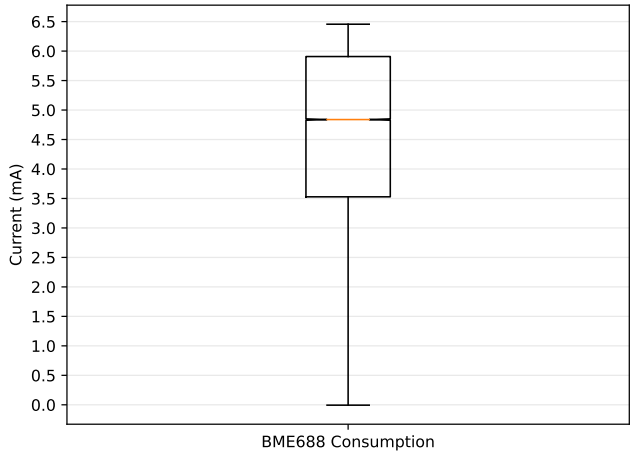


Figure 119: Consumption box plot board developed with heater profile (figure 62)

As expected, the sensor, when powered at a voltage of 1.8 volts, consumes less power than when powered at a voltage of 3.3 volts. When the heating profile of the sensor is the heater profile mentioned, the sensor on the developed board has a lower consumption distribution than on the *DevKit* board. Thus, the developed board ends up having a lower consumption since it has fewer sensors, and these consume less.

Figure 120 presents the power consumption of the entire board, and not just one sensor, as in the figures above. All the sensors are running and heated with the heater profile.

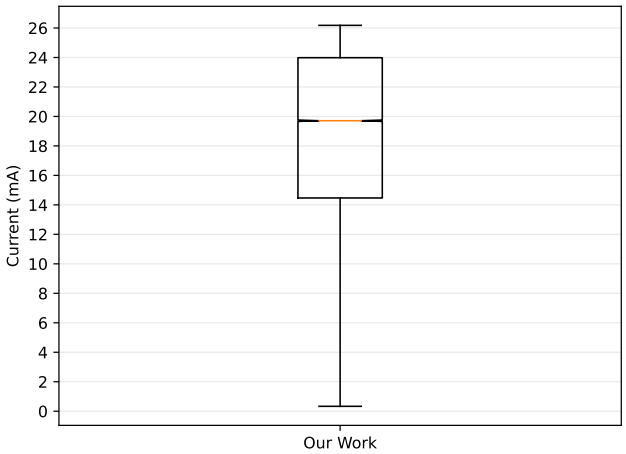


Figure 120: Consumption box plot board developed

5.3 Artificial intelligence algorithms

The training and development of the artificial intelligence algorithms are not covered in this dissertation, i.e. they have been developed by someone external. After training the different algorithms, they presented an accuracy of 90% for a decision tree, 83% for an SVM, and 72% for a neural network. Considering that the first two are classifiers and the last is a regressor, comparing the accuracy of all of them is not an appropriate approach.

In this dissertation, the algorithms were only coded and implemented. Thus tests were developed to evaluate the energy consumption and execution time of each algorithm. For the implemented regressor, the memory footprint was also analyzed.

5.3.1 Neural network

For the tests and results of the neural network implementation, the memory footprint of the algorithm implementation will be analyzed. Lastly, the algorithm will be validated by comparing actual and calculated concentrations.

Memory footprint

Two alternatives for the implementation of the LUT were previously presented. The first was to fill the LUT in run-time, and the other would be to implement the LUT already with hard-coded values.

Considering that in these alternatives, a trade-off is done between memory and processing resources, the memory footprint of the code implementing the LUT was analyzed.

Keil MDK generates the .map file that explains each function's size and address occupied. That file was checked, and it is possible to see the code memory, the RO-data, RW data, and Zi data.

The RO data is the read-only data, which holds the constants defined in the code. The RW data is the read/write data, which holds the initialized variables. The Zi data (zero initialize) is the variables that are not initialized or are initialized to zero.

On table 24 it is possible to see the amount of memory used.

Table 24: Memory footprint

	Code	RO Data	RW Data	Zi	Units
LUT Runtime	24180	708	56	4176	Bytes (B)
LUT HardCoded	21084	2908	56	1760	Bytes (B)

When the LUT is filled at runtime, the code memory is larger than when the LUT is hardcoded since the LUT initialization function is used. The RO data is smaller for the first case since the LUT is initialized with the calculated values in the second case and is, therefore, constants. The RW data remains unchanged. Zi is higher in the first case because the LUT is initialized to zero.

With this, it is obtained for the runtime LUT:

$$Total\ RO\ Size\ (Code + RO\ Data) \qquad 24888\ (\ 24.3047\ kB)$$

$$Total\ RW\ Size\ (RWData + ZI\ Data) \qquad 4232\ (\ 4.1328\ kB)$$

When the LUT is hardcoded:

$$Total\ RO\ Size\ (Code + RO\ Data) \qquad 23992\ (\ 23.4297\ kB)$$

$$Total\ RW\ Size\ (RWData + ZI\ Data) \qquad 1816\ (\ 1.7734\ kB)$$

Therefore, the amount of flash used will be $24.30kB$, and the amount of RAM used during operation will be $4.13kB$ when the LUT is initialized in runtime. When the LUT is hardcoded, the amount of Flash will be $23.42kB$, and the amount of RAM during operation will be $1.77kB$.

Real vs calculated concentrations

Data from the dataset for different values of CO_2 concentration was introduced in the neural network. The output values of the network and the expected value were compared in table 25.

Table 25: Real vs calculated concentrations

Real Concentration (ppm)	Calculated Concentration (ppm)	Gas Resistance Median ($M\Omega$)	Gas Resistance Range ($M\Omega$)
0	12258.96	5.4	4 - 7
4100	10061.60	1.7	0.6 - 4
8200	9482.89	0.9	0.4 - 3
12300	9268.56	0.5	0.25 - 2.5
16400	19512.24	14.8	12 - 22
20500	21757.76	17.4	10 - 22
24600	22326.94	18.0	10 - 22
28700	22617.08	18.7	10 - 22
32800	22668.77	17.8	10 - 22
36900	22426.52	17.8	10 - 22
41000	22629.97	17.9	10 - 22

Figure 121 represents the boxplot of the gas resistance value for each concentration. Figure 122 represents the value of the real concentration versus the computed concentration.

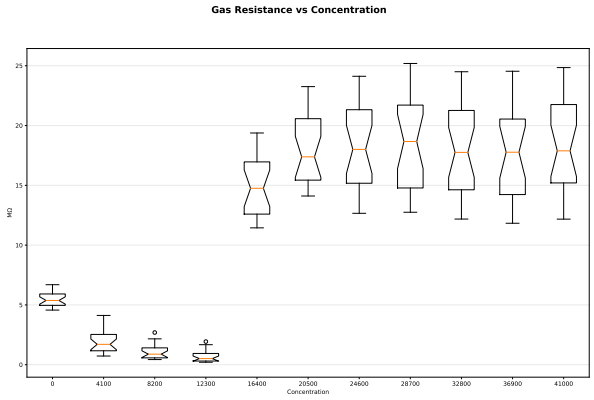


Figure 121: Boxplot of gas resistance of sensor 0 for each concentration

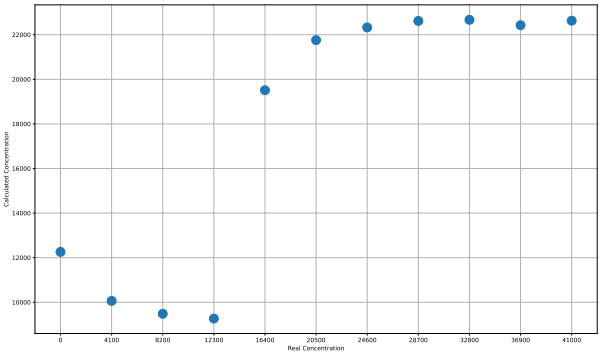


Figure 122: Real vs computed concentration

Thus, it was verified that concentrations above 28700 ppm for this sensor end up saturating. Concentrations above these do not modify the sensor's behavior, as this concentration is always in the same range of values and its curve is equal independently of the concentration. Thus, the network can not predict the amount of CO_2 in the air above this range of values.

Below the concentration of 12300 ppm, the neural network calculates a concentration very different from the real value.

Therefore, the implemented network efficiently calculates the CO_2 concentration between the mentioned values. However, it is important to determine from which (initial) value the calculation is efficient. As such, a regression is made between the values 12300ppm and 16400ppm, to discover the value for which the measured concentration is equal to the calculated one.

$$\begin{aligned} \chi_{real} &= \frac{16400 - 12300}{19512.24 - 8268.56} * \chi_{calc} + b \\ \chi_{real} &= 0.40025 * \chi_{calc} + b \\ 12300 &= 0.40025 * 99268.56 + b \Leftrightarrow b = 8590.26 \\ \chi_{real} &= 0.40025 * \chi_{calc} + 8590.26 \end{aligned} \tag{36}$$

when $\chi_{real} = \chi_{calc}$,

$$\begin{aligned} (1 - 0.40025) * \chi &= 8590.26 \\ \chi &= 14323 \end{aligned} \tag{37}$$

So when the real concentration is 14323 ppm, the calculated one will have the same value. This will be the initial value at which the network efficiently calculates the CO_2 concentration. In summary, the implemented neural network is efficient in calculating CO_2 between concentrations 1.4% and 2.3%.

5.3.2 Time duration

To test the duration time of each algorithm it was used the oscilloscope MDO3012. Every time each algorithm starts, an output port is set to a high value. Then, when the algorithm is finished, the port goes to a low level. The time that the port stays at a high level determines the duration of the algorithm.

It was analyzed the duration of the three algorithms. The decision tree, the SVM, and the neural network. A comparison can be made between the three. However, it is important to realize that the algorithms are different since two are classifiers and one a regressor. Being the first two classifiers, the duration between them can be directly compared.

The duration of the decision tree algorithm is represented in figure 123.

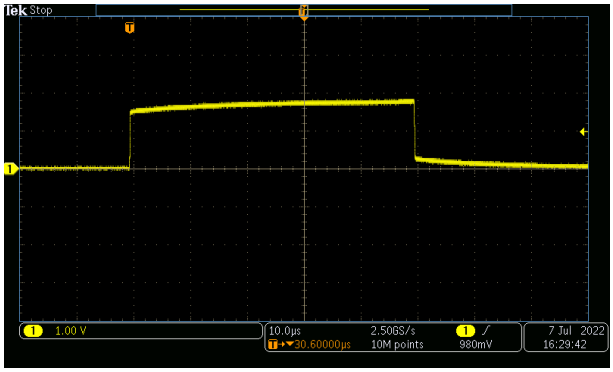


Figure 123: Decision tree algorithm duration (50uS)

The high level of the pin represents 1.8 volts, as can be seen in the figure. The duration of the high level is about 50μs.

The same process is done to the support vector machines algorithm. The duration of the algorithm is also represented in figure 124.

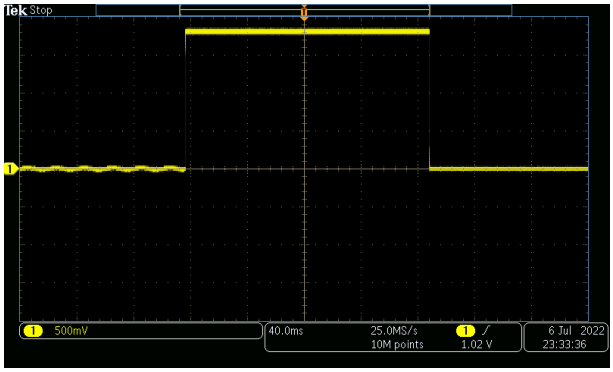


Figure 124: Support vector machine algorithm duration (160ms)

This is an algorithm that relies on more computational resources, so a longer duration would be expected. That conclusion can be seen on figures 123 and 124. The SVM algorithm takes $160ms$. The algorithm takes much longer than the decision tree.

Lastly, the duration of the implemented neural network algorithm was analyzed. The process was the same, and the result is in figure 126. The time required to initialize the LUT was analyzed first, and then the execution process of the neural network was analyzed. The initialization of the LUT is only done once in the whole program. As part of the neural network, the time consumed when it is filled was seen and represented in figure 125. Run-time initialization of the LUT is one of the possible alternatives. The other alternative is not time-consuming since the LUT is initialized externally (hard-coded).

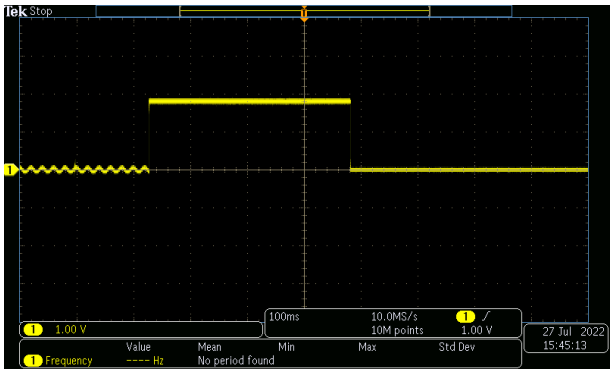


Figure 125: Time consumption of initialization of LUT (360ms)

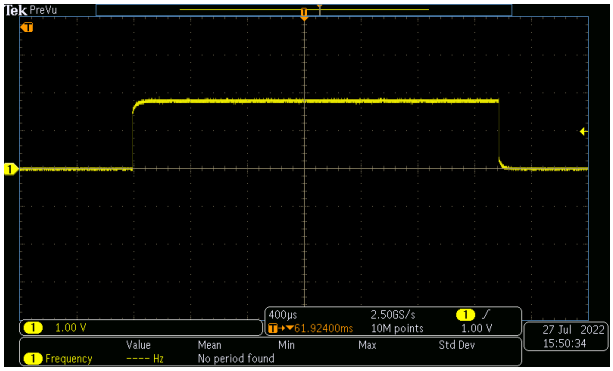


Figure 126: Time consumption of neural network (2500us)

The initialization of the LUT takes a relatively long time, approximately 360ms. However, the LUT is only executed once in the entire code duration so this time will be taken only at initialization. The time for executing the neural network is about 2500μs.

Finally, the neural network's running time without using the LUT was compared. The sigmoid function was used in the neural network. As such, the observed and estimated time for the algorithm duration is shown in the following figure 127.

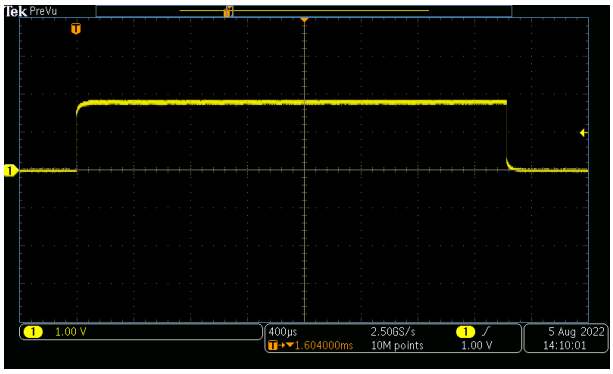


Figure 127: Time consumption of neural network without LUT (3000us)

Now, the time for the execution of the neural network is about $3000\mu s$. Therefore, it is possible to conclude that the implementation of the LUT reduces the algorithm's execution time by 16%.

5.3.3 Power consumption

Since all the algorithms are quite different in terms of duration, it would be a good approach to analyze their consumption. The digital multimeter (Keysight 34410A) was used for this purpose.

To do that test, each algorithm is looped and then measured power consumption. The first test is putting the board idle and analyzing its power consumption. The board consumption in idle was taken from the consumption obtained for each algorithm. Then the algorithms are placed in a loop and the power consumption of each one is measured.

Since the only interest is to analyze only the consumption of the algorithm, the sensors are not working.

A box plot was done to view the consumption distribution for each algorithm. Figure 128 represents the consumption of each algorithm. It is possible by visualizing the figure to estimate the algorithm that consumes the most.

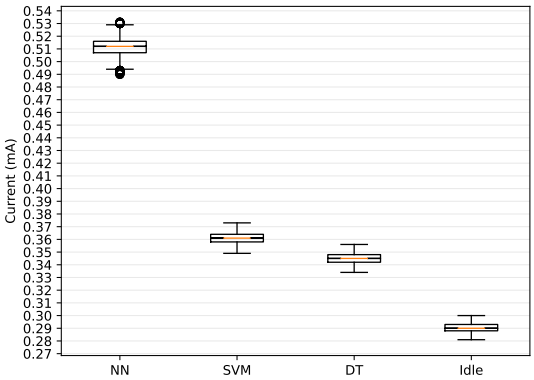


Figure 128: Algorithms box plot consumption

The most assertive comparison will be between the two classifiers. Thus it is possible to conclude that the decision tree algorithm has lower consumption and a shorter duration since it does not use so many resources. The SVM algorithm, which uses more computational resources, has higher consumption and a higher duration when compared to the decision tree. However, of all the artificial intelligence algorithms, the one that consumes the most energy is the neural network.

However, these consumptions are made continuously, and the duration times of each algorithm are neglected. Thus, taking into account the execution time of each algorithm, the consumption per cycle of each was obtained.

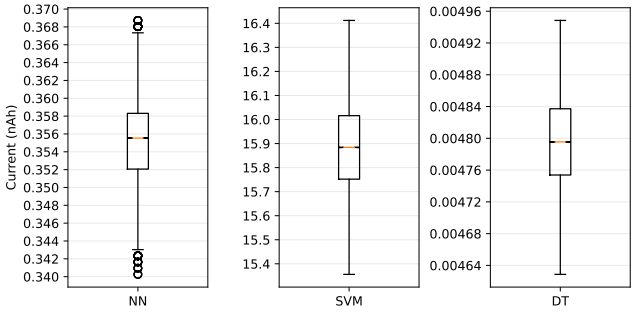


Figure 129: Algorithms consumption per cycle

So figure 129 shows the consumption per cycle for each of the algorithms. Under these conditions, it is possible to conclude that per cycle, SVM is the algorithm that consumes the most energy, followed by NN and decision tree.

5.3.4 Use case power consumption

A battery life study was carried out for the use case, in which a neural network was implemented. The life of the batteries used will be evaluated for this case. This evaluation considers the three operating modes, sleep, low-power, and stop. Tests were made in the different modes, and each mode's consumption was analyzed. Again the digital multimeter was used for this purpose. Finally, the battery life is calculated.

The calculation does not involve an hour, as taking consumption data for an hour would be unthinkable. Therefore an approximation was made, and a test was made for about 5 minutes. The average consumption taken during the burn-in of the sensor, including the rating, is taken as well as the sleep phase. The sum of the two averages taking into account the time they are subjected to, will give the average consumption per hour. In other words, the average consumption will be obtained by calculating the integral of average consumption versus time.

The data obtained when the sensor works for about 3 minutes, remaining the rest of the time in sleep mode, are shown in figure 130.

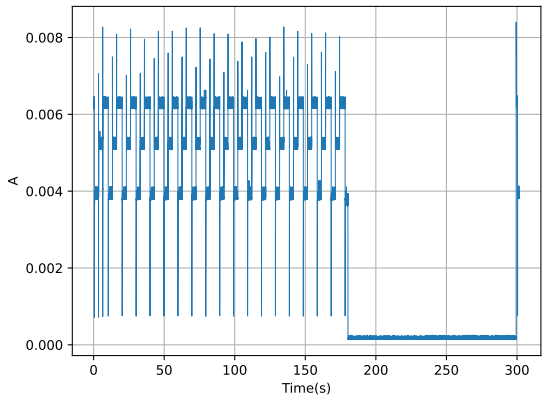


Figure 130: Sleep mode consumption

The first 180 seconds are related to the sensor burn-in, and the final part includes the calculation of the neural network. In the remaining seconds, the sensor goes into sleep mode. Finally, the sensor would restart the cycle.

The consumption during these times was averaged and obtained:

$$\begin{aligned} &Average\ current\ in\ sensor\ heating\ process\ +\ NN\ Calculation = 5.15mA \\ &Average\ current\ during\ sleep\ mode\ = 168\mu A \end{aligned} \tag{38}$$

Therefore, as a way to check if this value is correct, the microcontroller datasheet was analyzed. As such, it was verified that at a range 3 (Vcore = 1.2V) and with MSI at a frequency of 4.2MHz, the consumption should be between 150 and 240μA [16].

The measured value in this experiment is about 168μA. However, the microcontroller has its internal regulator in range 1 (Vcore = 1.8V) and a frequency of 2.09 MHz. Therefore, it is difficult to draw firm

conclusions about this consumption. As a test, the microcontroller was placed in range 3 at a frequency of 4.2MHz, and the consumption was analyzed.

The consumption in sleep mode with these conditions is around 159μA. This value is within the expected range, concluding that the results are in line with expectations [16].

Thus, the calculation for the battery lifetime, under initial conditions is done.

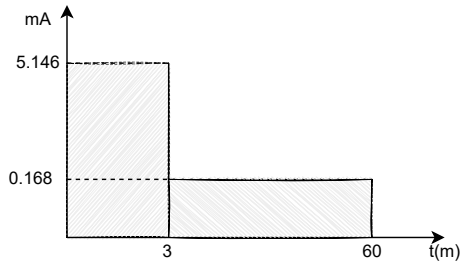


Figure 131: Example calculation current

Therefore, taking into account figure 131, to obtain the average consumption, the integral of the consumption will have to be done, that is, the area of each consumption, to obtain the consumption per hour.

$$\begin{aligned} &Consumption(Ah) = 0.005146 * (0.05) + 0.000168 * (0.95) \Leftrightarrow \\ &Consumption(Ah) = 4.169 * 10^{-4} A \end{aligned} \tag{39}$$

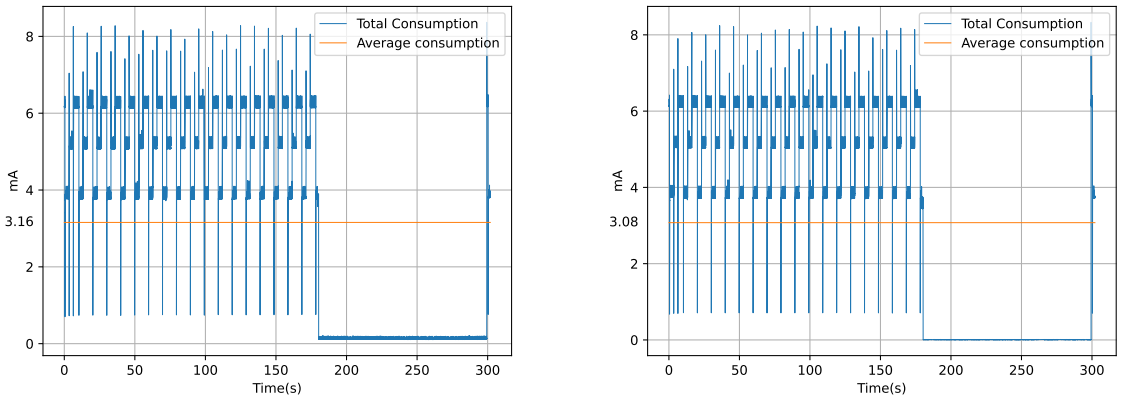
Using a $2.2Ah$, $4.8V$ battery provides:

$$\begin{aligned} &Duration(h) = \frac{2.2}{4.169 * 10^{-4}} \Leftrightarrow \\ &Duration = 5277.045\ hours\ \sim\ 220\ days \end{aligned} \tag{40}$$

It was concluded that these low performance battery would last 220 days. This time would decrease if the operating time of the sensor was reduced to two minutes, for example. Reducing the number of times the neural network runs, for example, every two hours, would also reduce this consumption. However, for different applications, this solution may be harmful.

Next, the duration in low-power sleep mode and stop mode will be analyzed. It will be verified if the consumption decreases, leading to an increase in battery life.

The microcontroller was placed in low-power sleep mode and stop mode. The process used to obtain the consumption was the same, and the results obtained are presented in figure 132.



(a) Low power sleep mode

(b) Stop mode

Figure 132: Power consumption low power sleep mode and stop mode

Again the average of each consumption was taken, and the battery life was calculated.

Table 26 shows the 3 power modes, the total average consumption, and the battery life for each mode.

Table 26: Consumption of use case in different power modes

	Average consumption Burn in and NN (mAh)	Average consumption Power Mode (mAh)	Total Consumption (mAh)	Lifetime Duration Days
Sleep	5.146	0.168	0.4169	219
Low Power Sleep	5.133	0.133	0.383	239
Stop Mode	5.082	0.007	0.261	351

As can be seen, the consumption obtained in stop mode is around $7\mu A$. This value will meet the expected value since the developed board is equipped with 4 BME688 sensors, which in sleep mode consume $1\mu A$ [6]. It has two voltage regulators with a quiescent current of $1\mu A$ [78]. It also has a microcontroller consumption of $1\mu A$ [16]. Together the predicted consumption of $7\mu A$ is obtained, being this the average measured consumption.

Therefore, it can be concluded that the use of stop mode and low-power sleep reduces the average consumption of the application. This leads to longer battery life compared to sleep and run modes.

6 | Conclusions and Future Work

This dissertation's objective was to develop a low-power printed circuit board and implement detection of H_2 and CO_2 gases using the BME688 sensor to study the behavior of this sensor for different target gases and implement algorithms for gas detection. Furthermore, the goal was to study and optimize the consumption of the system (sensor, board, and all the different algorithms).

For this purpose, the *Devkit* [61] board was initially used. On this board, the operation of the sensor was studied, as well as its behavior for different gases. With this board, the datasets for CO_2 and H_2 gases were taken. It was concluded that the sensor reacts to both gases, providing an early warning in the case of hydrogen. Different AI algorithms were developed for CO_2 detection. The consumption of the sensor on the "DevKit"[61] board was studied for future comparison with the board to be developed.

The hardware was developed, and the BME688 sensor was implemented on a new board. This board brought a hardware simplification concerning the communication between the sensor and the microcontroller. The whole board was developed with consumption reduction in mind. The components were chosen for this purpose. The firmware was developed correctly, and it is possible to get a data set from the sensors implemented on the new board. It was concluded that, as the sensors have very different characteristics between them, it would be necessary to take a dataset of the sensor to be used to implement the algorithms. The device was tested in an accelerated aging environment for 1932 hours without visible degradation.

For the developed board, it was concluded that the sensor consumes less power since it is powered at 1.8V. The sensor consumption for the heater profile used was also analyzed.

The time of execution and consumption of the algorithms developed for gas detection was analyzed. In the neural network, the use of a LUT was explored. The duration of the algorithm was analyzed with the LUT and without it. The duration would be reduced by 16% when using it, so it was used. For the LUT, two alternatives were also analyzed. The filling of the LUT in run-time or hardcoded. The second alternative is the best one since it does not require the initialization of the LUT in run-time.

The duration of the remaining algorithms was analyzed, concluding that this increases with the algorithm's complexity.

The lifetime of a battery was studied in a real context using the neural network algorithm. To do so, the existing low-power modes were used since the board would remain in one of these modes most of the time. The consumptions were analyzed, concluding that the lowest consumption is obtained using the

stop mode, with 351 days of battery life with the low performance battery used. The remaining modes have shorter durations. This consumption is because the sensor consumes a high current in the burn-in phase. During the 3-minute warm-up period, it consumes a lot of energy. By decreasing this time, longer battery life would be achieved. Putting the board with only one sensor would be beneficial since, in the sleep phase, only one sensor would be in sleep mode, reducing consumption by $3\mu A$.

6.0.1 Future Work

Several aspects and open points still require further work for the development of the project. The following aspects are considered for future work:

- Calibration of the sensor on the developed board, taking a dataset from that sensor, and implementing the algorithms for that sensor and dataset;
- Reducing the number of sensors on the board to one or implementing an algorithm that uses the differences between the different sensors for gas detection;
- Development of a new board with reduced dimensions to place it in a project-specific box. This modification could lead to the development of a 4-layer board and the placement of components on both layers. The total cost of the board would increase with these modifications.
- Review of all the HW developed and study of possible improvements on it. This includes the study of new circuits that meet the system requirements more efficiently.
- Implementation of algorithms where it is possible to include/avoid the burn-in time of the sensor, aiming at a reduction of consumption;
- For the dataset obtained, develop a more robust neural network with more hidden layers to perform the regression of the concentration;
- Verification and validation of the implementation of the modem will have to be developed since it was not used during the dissertation, although implemented;
- Use of *threadX* and implementation of all algorithms in threads. Further implementation of all code in the overall project, which uses *threadX*. This implementation requires the modem to send data to the cloud.

Thus, with all the aspects mentioned above, it is possible to make several modifications to the work developed.

Bibliography

[1] *1.4. Support Vector Machines — scikit-learn 1.1.2 documentation*. url: https://scikit-learn.org/stable/modules/svm.html.

[2] ABRACON. *ABS07 Crystal SMD*. url: https://eu.mouser.com/datasheet/2/3/ABS07-11028.pdf.

[3] B. R. Archambeault. *PCB DESIGN FOR REAL-WORLD EMICONTROL*. isbn: 978-1-4757-3642-7. doi: 10.1007/978-1-4757-3640-3.

[4] *Atmel AVR1017: XMEGA-USB Hardware Design Recommendations Features*. url: https://ww1.microchip.com/downloads/en/Appnotes/doc8388.pdf.

[5] H. Bai and G. Shi. "Gas Sensors Based on Conducting Polymers." In: *Sensors 2007, Vol. 7, Pages 267-307* 7 (3 Mar. 2007), pp. 267–307. issn: 14248220. doi: 10.3390/S7030267.

[6] *BME688 Digital Low Power gas, pressure, temperature & humidity sensor with AI*. url: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme688-ds000.pdf.

[7] V. E. Bochenkov and G. B. Sergeev. *Sensitivity, Selectivity, and Stability of Gas-Sensitive Metal-Oxide Nanostructures*. 2010. isbn: 1588831760.

[8] R. Bogue. "Detecting gases with light: A review of optical gas sensor technologies." In: *Sensor Review* 35 (2 Mar. 2015), pp. 133–140. issn: 02602288. doi: 10.1108/SR-09-2014-696/FULL/PDF.

[9] C. Bruno, A. Licciardello, G. A. M. Nastasi, F. Passaniti, C. Brigante, F. Sudano, A. Faulisi, and E. Alessi. "Embedded Artificial Intelligence Approach for Gas Recognition in Smart Agriculture Applications Using Low Cost MOX Gas Sensors." In: *2021 Smart Systems Integration, SSI 2021* (Apr. 2021). doi: 10.1109/SSI52265.2021.9467029.

[10] J. Carl and MicroChip. *Implementation Guidelines for Microchip's USB 2.0 and USB 3.1 Gen 1 and Gen 2 Hub and Hub-Combo Devices*. url: http://ww1.microchip.com/downloads/en/Appnotes/AN26.2-Application-Note-DS00001876C.pdf.

[11] Y. Chen, H. Qin, Y. Cao, H. Zhang, and J. Hu. "Acetone Sensing Properties and Mechanism of SnO 2 Thick-Films." In: (). doi: 10.3390/s18103425.

[12] Y. Choi, S. Member, N. Chang, S. Member, and T. Kim. "DC-DC Converter-Aware Power Management for Low-Power Embedded Systems." In: *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS* 26 (8 2007). doi: 10.1109/TCAD.2007.890837.

[13] J. Chou. *Hazardous gas monitors : a practical guide to selection, operation and applications*. McGraw-Hill, 2000, p. 258. isbn: 9780071358767.

[14] *Conector 1776275-2*. url: https://eu.mouser.com/datasheet/2/418/8/product_1776275_2_datasheet-2517465.pdf.

[15] M. G. Corporation. *Getting Started Getting Started in PCB Design in PCB Design ® PADS*. 2007.

[16] *Datasheet STM32L081CB, STM32L081CZ, STM32L081KZ*. 2019. url: https://www.st.com/resource/en/datasheet/stm32l081cb.pdf.

[17] A. P. Deepak and S. Chouhan. "ICATE 2013 Paper Identification Number-102 SVM Kernel Functions for Classification." In: (2013). doi: 10.1109/ICAdTE.2013.6524743.

[18] *ESD Protection Diode, ESD9L*. url: https://eu.mouser.com/datasheet/2/308/ESD9L_D-1805819.pdf.

[19] *ESD9L ESD Protection Diode*. url: https://pt.mouser.com/datasheet/2/308/ESD9L_D-1805819.pdf.

[20] S. Feng, F. Farha, Q. Li, Y. Wan, Y. Xu, T. Zhang, and H. Ning. "Review on Smart Gas Sensing Technology." In: (2019). doi: 10.3390/s19173760. url: www.mdpi.com/journal/sensors.

[21] *Gas detector - Wikipedia*. url: https://en.wikipedia.org/wiki/Gas_detector#Semiconductor.

[22] S. Ghosh, A. Dasgupta, and A. Swetapadma. "A study on support vector machine based linear and non-linear pattern classification." In: *Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2019* (Feb. 2019), pp. 24–28. doi: 10.1109/ISS1.2019.8908018.

[23] J. B. Gomes, J. J. Rodrigues, R. A. Rabêlo, N. Kumar, and S. Kozlov. "IoT-Enabled Gas Sensors: Technologies, Applications, and Opportunities." In: *Journal of Sensor and Actuator Networks 2019, Vol. 8, Page 57* 8 (4 Dec. 2019), p. 57. issn: 22242708. doi: 10.3390/JSAN8040057.

[24] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2017.

[25] M. T. Hagan and H. B. Demuth. "Neural Networks for Control." In: (1999). doi: 10.1109/ACC.1999.786109.

[26] D. Haridas, A. Chowdhuri, K. Sreenivas, and V. Gupta. "Enhanced room temperature response of SnO2 thin film sensor loaded with Pt catalyst clusters under UV radiation for LPG." In: *Sensors and Actuators, B: Chemical* 153 (1 Mar. 2011), pp. 152–157. issn: 09254005. doi: 10.1016/J.SNB.2010.10.024.

[27] T. I. Incorporated and R. Liang. “Design considerations for system-level ESD circuit protection.” In: (). url: www.ti.com/aaj.

[28] T. Instruments. *TCA6408A*. url: https://www.ti.com/lit/ds/symlink/tca6408a.pdf?ts=1663603634895&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTCA6408A%252Fpart-details%252FTCA6408ARGTR.

[29] D. Jain, S. Technology, and J. Cummins. “Understanding EMC and EMI Basics with Würth Elektronik.” In: ().

[30] Y. Javed, M. Mansoor, and I. A. Shah. “A review of principles of MEMS pressure sensing with its aerospace applications.” In: *Sensor Review* 39 (5 Aug. 2019), pp. 652–664. issn: 02602288. doi: 10.1108/SR-06-2018-0135.

[31] C. Johansson and T. Manefjord. “Analysis of a high-speed PCB design.” In: *2017 IEEE Nordic Circuits and Systems Conference, NORCAS 2017: NORCHIP and International Symposium of System-on-Chip, SoC 2017, Proceedings* 2017-January (Nov. 2017), pp. 1–4. doi: 10.1109/NORCHIP.2017.8124982.

[32] F. Karagulian, M. Barbiere, A. Kotsev, L. Spinelle, M. Gerboles, F. Lagler, N. Redon, S. Crunaire, and A. Borowiak. “Review of the Performance of Low-Cost Sensors for Air Quality Monitoring.” In: (2019). doi: 10.3390/atmos10090506. url: www.mdpi.com/journal/atmosphere.

[33] I. Kecskés, L. Székács, and P. Odry. “Lookup table based fuzzy controller implementation in low-power microcontrollers of hexapod robot Szabad(ka)-II.” In: (2015). url: https://www.researchgate.net/publication/278300786.

[34] D. Kilani, B. Mohammad, H. Saleh, and M. Ismail. *LDO Regulator versus Switched Inductor DC-DC Converter*. 2014. isbn: 9781479942428. doi: 10.1109/ICECS.2014.7050066.

[35] L. Klibanov and P. Boldt. “Preliminary analysis of Bosch BME688 4-in-1 environmental sensor with AI.” In: *Research Gate* (Nov. 2021).

[36] S. Kolokowsky, T. Davis, and C. Semiconductor. *Common USB Development Mistakes*. url: https://www.mikrocontroller.net/attachment/104429/Common_USB_Development_Mistakes_Cypress.pdf.

[37] T. R. Kuphaldt. *”Lessons In Electric Circuits, Volume III – Semiconductors”*. Fifth. Vol. 3. Design Science License, 2009.

[38] J. H. Lau and C. Chang. “Overview of microvia technology.” In: *Circuit World* 26 (2 2000), pp. 22–32. issn: 03056120. doi: 10.1108/03056120010310891/FULL/PDF.

[39] R. Lin. *Human-Centered Circuit Board Design With Flexible Levels of Abstraction and Ambiguity*. University of California, Dec. 2021. url: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-259.html.

[40] X. Liu, S. Cheng, H. Liu, S. Hu, D. Zhang, and H. Ning. "A Survey on Gas Sensing Technology." In: *Sensors (Basel, Switzerland)* 12 (7 July 2012), p. 9635. issn: 14248220. doi: 10.3390/S120709635.

[41] S. Marathe, P. Wei, S. Ze, L. Guan, and D. Pommerenke. "Scenarios of ESD discharges to USB connectors." In: *Electrical Overstress/Electrostatic Discharge Symposium Proceedings* (Oct. 2017). issn: 07395159. doi: 10.23919/EOSESD.2017.8073431.

[42] L. B. Mendes, N. W. Ogink, N. Edouard, H. J. C. van Dooren, I. de Fátima F. Tinôco, and J. Mosquera. "NDIR Gas Sensor for Spatial Monitoring of Carbon Dioxide Concentrations in Naturally Ventilated Livestock Buildings." In: *Sensors 2015, Vol. 15, Pages 11239-11257* 15 (5 May 2015), pp. 11239–11257. issn: 1424-8220. doi: 10.3390/S150511239.

[43] MicroChip. *Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers*. url: https://pt.mouser.com/datasheet/2/268/20001984g-846362.pdf.

[44] "MQ-2 Semiconductor Sensor for Combustible Gas." In: ().

[45] *Multi-sheet & Hierarchical Designs in Altium Designer | Altium Designer 18.1 User Manual | Documentation*. url: https://www.altium.com/documentation/altium-designer/multi-sheet-multi-channel-design?version=18.1.

[46] I. Nascimento, R. Jardim, and F. Morgado-Dias. "A new solution to the hyperbolic tangent implementation in hardware: polynomial modeling of the fractional exponential part." In: (). doi: 10.1007/s00521-012-0919-0.

[47] E. A. NI. *PCB Design Fundamentals: Prototyping and the PCB Design Flow - NI*. 2022. url: https://www.ni.com/pt-pt/innovations/white-papers/10/pcb-design-fundamentals--prototyping-and-the-pcb-design-flow.html.

[48] *Non-Dispersive Infrared Sensing Technology | Edinburgh Sensors*. url: https://edinburghsensors.com/non-dispersive-infrared-sensing-technology/.

[49] *Operating principle Catalytic-type gas sensor*. url: https://www.figaro.co.jp/en/technicalinfo/principle/catalytic-type.html.

[50] *Operating principle NDIR-type gas sensor*. url: https://www.figarosensor.com/technicalinfo/principle/ndir-type.html.

[51] *Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs*. url: https://www.st.com/content/ccc/resource/technical/document/application_note/c6/eb/5e/11/e3/69/43/eb/CD00221665.pdf/files/CD00221665.pdf/jcr:content/translations/en.CD00221665.pdf.

[52] M. Papadonikolakis and C.-S. Bouganis. "A Scalable FPGA Architecture for Non-Linear SVM Training." In: *IEEE International Conference on Field-Programmable Technology (FPT)* (2008). doi: 10.1109/FPT.2008.4762412.

[53] Z. Peterson. *Complete Guide to Types of PCB Vias: Designs and Routing | PCB Routing | Altium*. Jan. 2020. url: https://resources.altium.com/p/pcb-via.

[54] C. F. Poole. "Gas Chromatography - Detectors." In: *Encyclopedia of Analytical Science: Second Edition* (Jan. 2004), pp. 95–105. doi: 10.1016/B0-12-369397-7/00222-3.

[55] Quectel. *BC66-NA*. url: https://www.quectel.com/wp-content/uploads/2021/03/Quectel_BC66-NA_Hardware_Design_V1.1.pdf.

[56] G. A. Rincon-Mora and P. E. Allen. "Study and Design of Low Drop-Out Regulators." In: ().

[57] M. Robert. *An Introduction to Universal Serial Bus 2.0*. url: https://www.infineon.com/dgdl/Infineon-AN57294_USB_101_An_Introduction_to_Universal_Serial_Bus_2.0-ApplicationNotes-v09_00-EN.pdf?fileId=8ac78c8c7cdc391c017d072d8e8e5256.

[58] Saft. *Nickel-Metal Hydride VHT Cs*. url: https://cellpacksolutions.co.uk/wp-content/uploads/2015/06/saft-vhtcs-technical-data-sheet.pdf.

[59] N. Semiconductors. *PCF8523 Real-Time Clock (RTC) and calendar*.

[60] N. Sensor. "Technical Information and User Manual NE4-CO Electrochemical Carbon Monoxide (CO) Gas Sensor." In: (). url: https://www.nemoto.eu/ne4-co-carbon-monoxide-sensor.

[61] B. Sensortec. *BME688 Development Kit*. url: https://www.bosch-sensortec.com/media/boschsensortec/downloads/product_flyer/bst-bme688-fl001.pdf.

[62] S. Sensortech. "VQ5 Series Thermal Conductivity Gas Detector Elements." In: (). url: https://www.aepint.nl/wp-content/uploads/2015/01/VQ5.pdf.

[63] S. Sharma, S. Sharma, and A. Athaiya. "ACTIVATION FUNCTIONS IN NEURAL NETWORKS." In: *International Journal of Engineering Applied Sciences and Technology* 4 (2020), pp. 310–316. issn: 2455-2143.

[64] V. Shirmohammadli, A. Saberkari, H. Martínez-García, and E. Alarcón-Cot. "An efficient CMOS LDO-assisted DC/DC buck regulator; An efficient CMOS LDO-assisted DC/DC buck regulator." In: *IEEE* (Nov. 2016). doi: 10.1109/DCIS.2016.7845356.

[65] *SingIFuse SF-0603HI-F High Inrush Current Withstand Surface Mount Fuses*. url: https://www.bourns.com/docs/product-datasheets/sf-0603hi-f.pdf?sfvrsn=dc6973f6_12.

[66] *sklearn.preprocessing.MinMaxScaler — scikit-learn 1.1.2 documentation*. url: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[67] "Specification of Thermopile Sensor MTP20-A6-CH 4." In: (). url: www.eoc-inc.com.

[68] STMicroelectronics. *User manual ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32*. 2018. url: https://www.st.com/resource/en/user_manual/um1075-stlinkv2-incircuit-debuggerprogrammer-for-stm8-and-stm32-stmicroelectronics.pdf.

[69] E. Systems. *ESP32-WROOM-32E Datasheet*. url: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[70] C. Tao and A. A. Fayed. "A Low-Noise PFM-Controlled Buck Converter for Low-Power Applications." In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 59 (12 2012). doi: 10.1109/TCSI.2012.2206464. url: http://ieeexplore.ieee.org..

[71] *Technical Data MQ-2 Gas Sensor*. url: https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf.

[72] "Technical Information Sheet Ne4-CO Eletrochemical Carbon Monoxide (CO) Gas Sensor." In: (). url: https://www.nemoto.eu/ne4-co-carbon-monoxide-sensor.

[73] K. Technologies. *B2900A Series Precision Source/Measure Unit Datasheet*. url: https://www.keysight.com/us/en/assets/7018-02794/data-sheets/5990-7009.pdf.

[74] Tektronix. *Mixed Domain Oscilloscopes MD03000 Series Datasheet*. url: www.tek.com.

[75] I. Texas. *2 USB PHY Layout Guide 2.1 General Routing and Placement 2.2 Specific Guidelines for USB PHY Layout 2.2.1 Analog, PLL, and Digital Power Supply Filtering*. Dec. 2007. url: https://datasheet.octopart.com/TMS320DM365ZCE27-Texas-Instruments-datasheet-10257056.pdf.

[76] "TGS 6812-D00-for the detection of Hydrogen, Methane, and LP Gas." In: (). url: www.figaro.co.jp.

[77] N. L. Torad and M. M. Ayad. "Gas Sensors Based on Conducting Polymers." In: *Gas Sensors* (Nov. 2019). doi: 10.5772/INTECHOPEN.89888.

[78] Torex. *XC9265 Series Ultra Low Power Synchronous Step-Down PFM DC/DC Converter GreenOperation Compatible ETR05053-007*. url: https://eu.mouser.com/datasheet/2/760/TOSL_S_A0012956478_1-2575198.pdf.

[79] UL. *UL 796 STANDARD FOR SAFETY Printed Wiring Boards*. Twelfth. Oct. 2020.

[80] *What is a Gas Sensor? Construction, Types & Working of Gas Sensors*. url: https://components101.com/articles/introduction-to-gas-sensors-types-working-and-applications.

[81] W. Y. Yi, K. M. Lo, T. Mak, K. S. Leung, Y. Leung, and M. L. Meng. "A Survey of Wireless Sensor Network Based Air Pollution Monitoring Systems." In: *Sensors 2015, Vol. 15, Pages 31392-31427* 15 (12 Dec. 2015), pp. 31392–31427. issn: 14248220. doi: 10.3390/S151229859.

[82] J. Yousaf, H. Lee, and W. Nah. "System Level ESD Analysis-A Comprehensive Review II on ESD Coupling Analysis Techniques." In: *J Electr Eng Technol* 13 (5 Sept. 2018). doi: 10.5370/JEET.2018.13.5.2033.

[83] Z. Yunusa, M. N. Hamidon, A. Kaiser, and Z. Awang. "Gas sensors: A review." In: *Sensors and Transducers* 168 (4 Jan. 2014), pp. 61–75. issn: 17265479. doi: 10.13074/JENT.2015.12.153163.

A | Appendix Bill of Materials

Comment	Description	Designator	Footprint	Quantity
470 uF	Capacitor	C40	2917	1
10 uF	Capacitor	C3, C5	0603	2
22 uF	Capacitor	C4, C23	0603	2
5 pF	Capacitor C0G	C6, C8	0603	2
100 nF	Capacitor	C7, C9, C10, C12, C14, C15, C16, C17, C18, C19, C20, C21, C22, C24, C25, C26, C28, C30, C34	0603	19
100 nF	Capacitor	C27, C38	1206	2
100 uF	Capacitor	C29	1206	1
100 pF	Capacitor C0G	C31	0603	1
33 pF	Capacitor	C35, C36, C37	0603	3
22 pF	Capacitor C0G	C39	0603	1
Diode Schottky	Diode Schottky	D1, D2	DO-214AA SMB	2
2A	Fuse	F1	0603	1
HEADER 1x40	HEADER 1x40 Male	H1	HEADER 1x5 TH MALE	1
HEADER 1x16	HEADER 1x16 Female	HEADER1	Header 1x16 Female	1
XC9265A331MR-G	Voltage Regulator 3V3	IC1	SOT95P280X130-5N	1
BME688	Integrated Circuit	IC2, IC3, IC4, IC5	BME688	4
XC9265C181MR-G	Voltage Regulator 1V8	IC6	SOT95P280X130-5N	1
M95M02-DRMN6TP	Integrated Circuit	IC8	SOIC127P600X175-8N	1
1776275-2	Connector	J1	17762752	1
U.FL-R-SMT-1(10)	CONN RCPT STR 50 OHM SMD	J2	FP-U_FL-R-SMT-1_10-MFG	1
693043020611	Nano SIM Card Connector, Push & Pull	J3	693043020611	1
Inductor	Inductor 10 uH	L1, L2	0603	2
MosFet P-Channel		Q1	SOT-23-3	1
MosFet N-Channel		Q2, Q3, Q4	SOT-23-3	3
Resistor 1 kohm	Resistor	R3	0603	1
Resistor 1 ohm	Resistor	R4	0603	1
Resistor 1 Mohm	Resistor	R5, R6, R10, R11, R12, R18	1206	6
Resistor 10 kohm	Resistor	R7, R8, R9, R16	0603	4
Resistor 0 ohm	Resistor	R13	0603	1
Resistor 22 ohm	Resistor	R14, R15, R17	0603	3
SW_FSMSM	TE CONNECTIVITY Tactile Switch	SW1, SW2, SW3	SW_FSMSM	3
BC66-NA	Modem	U2	BC66	1
STM32L071K8U6	MCU	U4	UFQFPN32	1
USB MICRO	USB MICRO	USB1	USB MICRO	1
ESD9L3.3ST5G	Unidirectional Transient Voltage Suppressor	TVS2, TVS3, TVS4, TVS5, TVS6, TVS7, TVS8, TVS9, TVS10, TVS11, TVS12, TVS13, TVS14, TVS15, TVS16, TVS17, TVS18	SODFL1006X40N	17

Table 27: Bill Of Materials Board Developed

B | Appendix Schematics and Layout

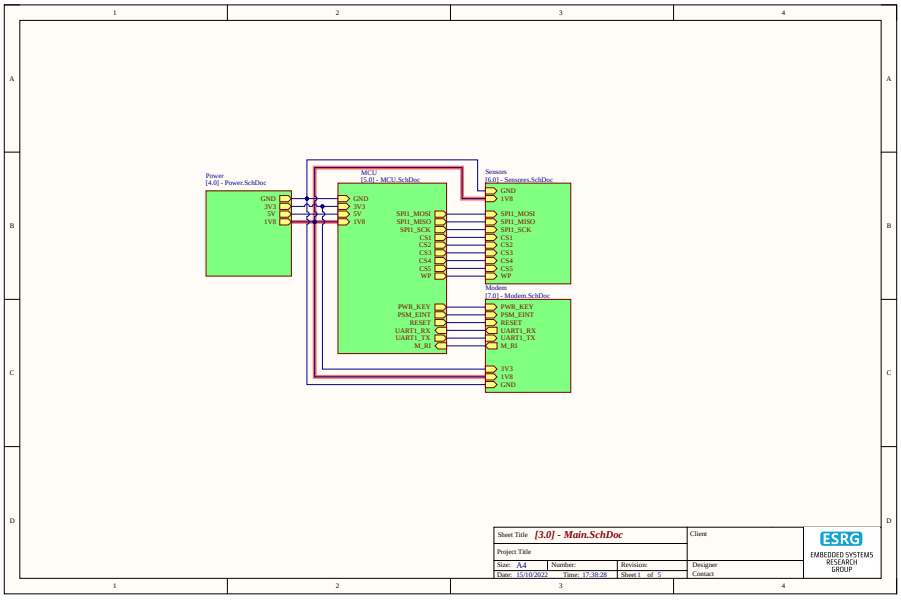


Figure 133: Main Schematic

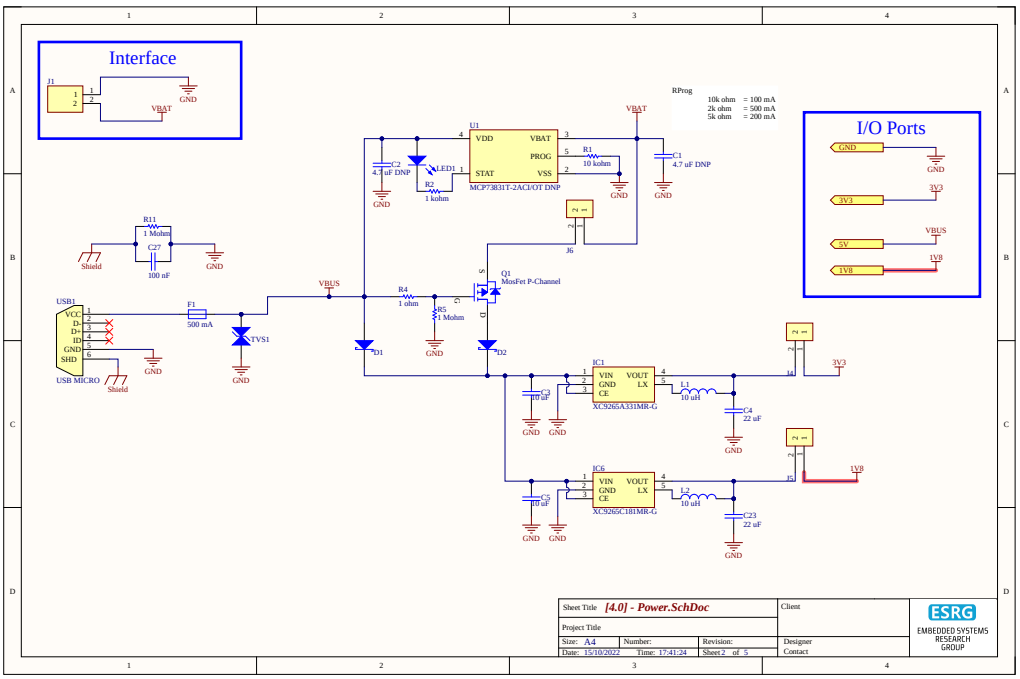


Figure 134: Power Schematic

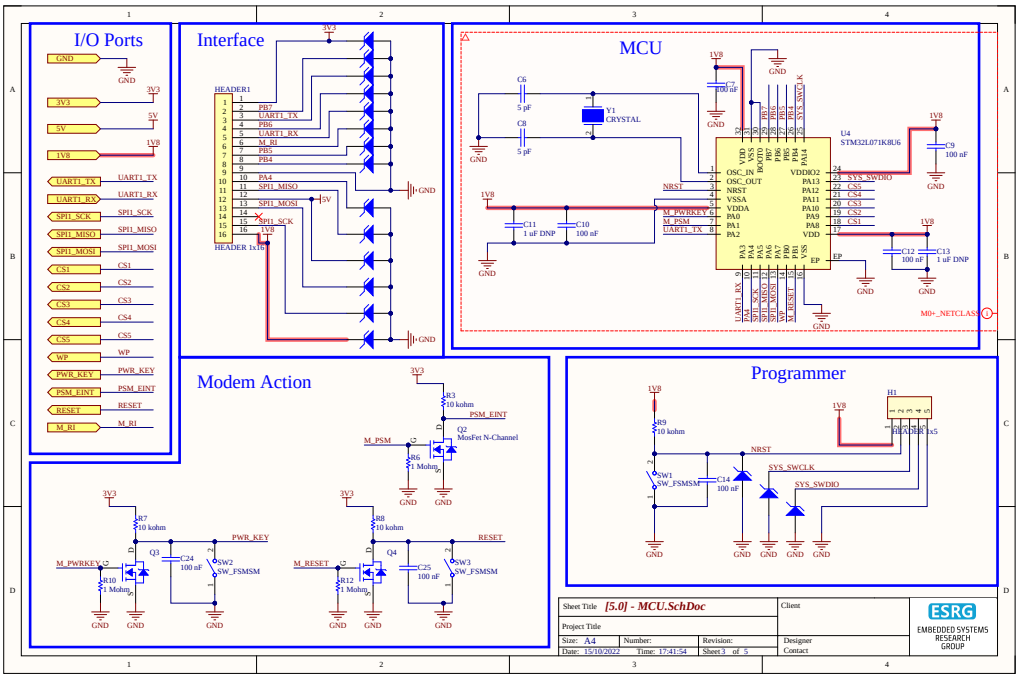


Figure 135: MCU Schematic

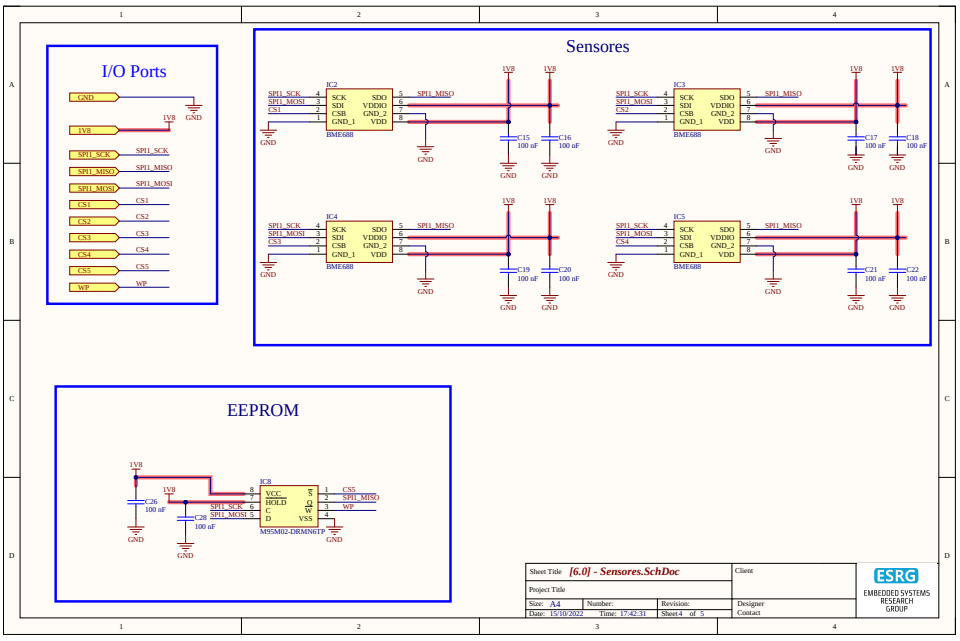


Figure 136: Sensors Schematic

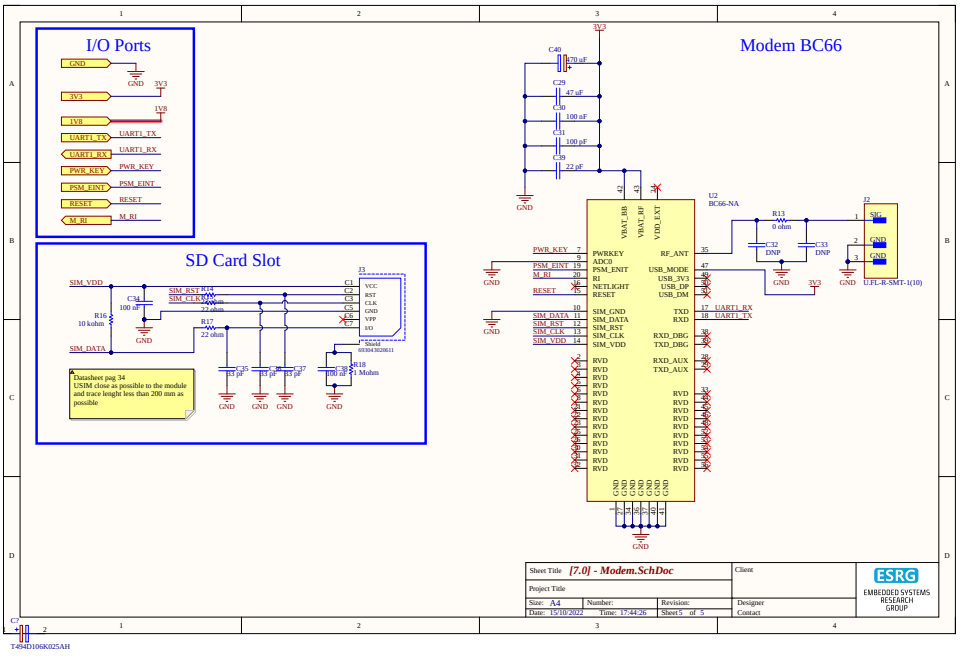


Figure 137: Modem Schematic

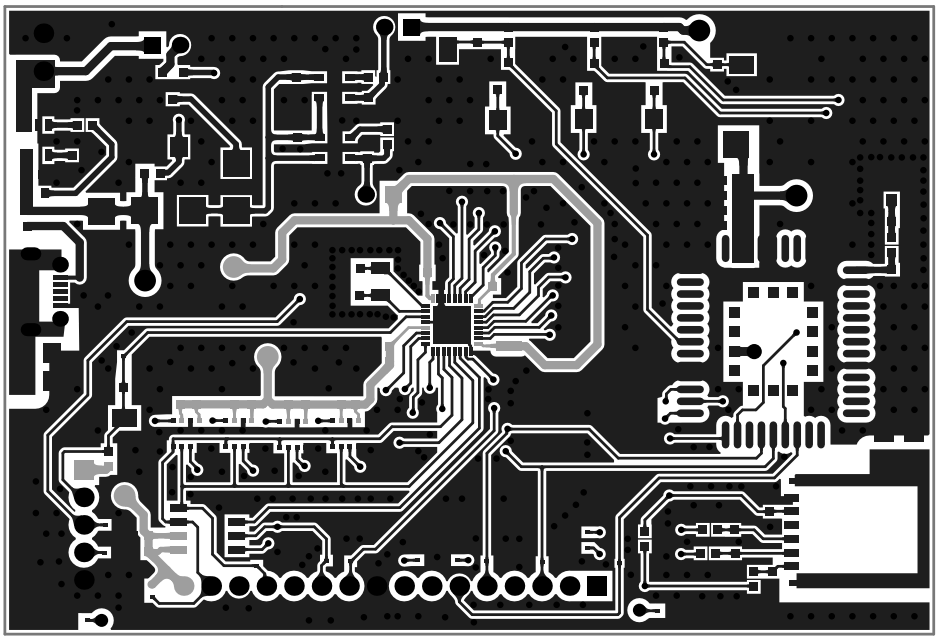


Figure 138: Top Layout

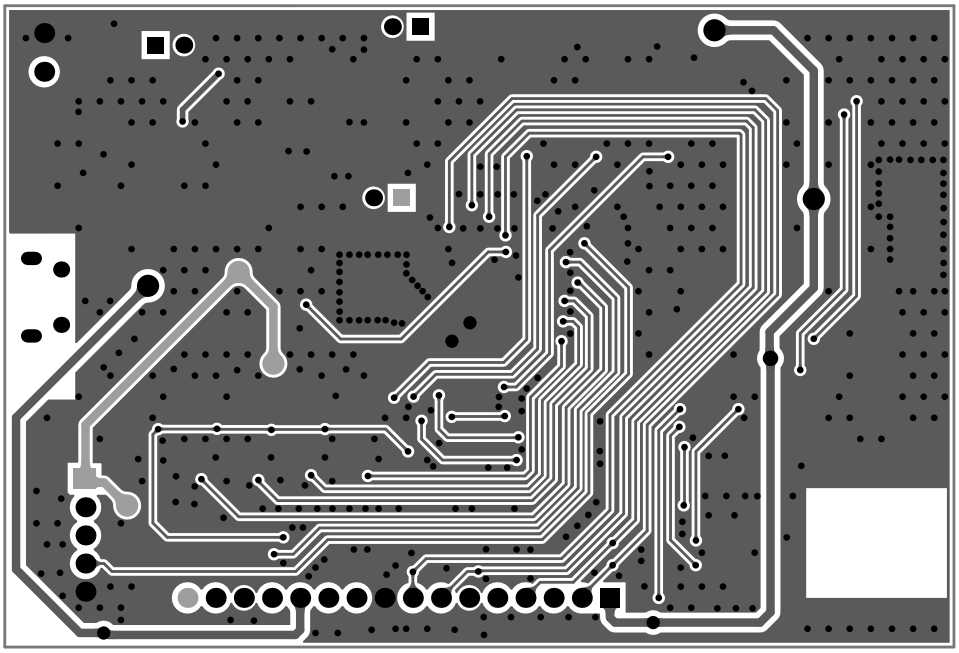


Figure 139: Bottom Layout