FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of an Ambient Air Quality Monitoring System

João Paulo Ribeiro de Sousa



Mestrado Integrado em Engenharia Mecânica

Especialização em Automação

Orientador: Prof. Joaquim Gabriel Magalhães Mendes

26 de março de 2021

Development of an Ambient Air Quality Monitoring System

João Paulo Ribeiro de Sousa

Mestrado Integrado em Engenharia Mecânica

Especialização em Automação

Esta Dissertação de Mestrado está integrada no Projeto POCI-01-0247-FEDER-041435 (Safe Cities - Inovação para Construir Cidades Seguras), no âmbito de um consórcio constituído pela Bosch Security Systems S.A. e pela Universidade do Porto (Reitoria, Faculdade de Engenharia e Faculdade de Ciências). Este projeto é cofinanciado pela União Europeia ao abrigo do Fundo Europeu de Desenvolvimento Regional, enquadrado no Programa Operacional Competitividade e Internacionalização (COMPETE 2020).

Cofinanciado por:

COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

Aprovado em provas públicas pelo Júri:

Presidente: Prof. António Manuel Ferreira Mendes Lopes

Arguente: Prof. Armando Jorge Miranda de Sousa

Vogal: Prof. Joaquim Gabriel Magalhães Mendes

26 de março de 2021

Abstract

This work aimed to develop an air quality monitoring system, both ambient and indoor. For its development, a previous analysis of the legislation and recommendations indicated the relevant parameters are temperature and relative humidity (environmental factors) and concentrations of O_3, NO_2, CO, CO_2, Total VOC, NH_3 and particulate matter (pollutants), being different for indoor and outdoor air quality assessment.

The system was assembled with three devices physically dispersed, collecting the air quality parameters. Each monitoring device includes an ESP32-based microcontroller unit and appropriate sensors. The devices are continuously acquiring and sending data to a server device, using LoRa protocol, a low-power and long-range modulation technique. This server, through Wi-Fi communication, sends the data by MQTT, a publish/subscribe protocol, to an online server where data is stored and displayed in real-time.

Since the system should be autonomous, power consumption is a concern. Therefore, sleep modes of devices were explored, along with coupling a photovoltaic cell to the system. Mechanical protections were designed for weather events, such as rain or strong wind.

Ozone sensor data was compared with data provided by nearby environmental stations for characteristic curve correction, as indicated by the manufacturer. The test results for temperature sensors indicate that they are sensitive to solar radiation, and that the two different sensors used in this work present different accuracies. For other parameters, it was not possible to validate data due to lack of a reference equipment.

Keywords: Air quality, ESP32, LoRa, MQTT, Photovoltaic solar panel

Resumo

Este trabalho teve como objetivo o desenvolvimento de um sistema de monitorização da qualidade do ar, tanto ambiente como interior. Para o desenvolvimento do mesmo, uma análise prévia de legislação e recomendações existentes indicou que os parâmetros relevantes são temperatura e humidade relativa (fatores ambientais) e concentração de O_3, NO_2, CO, CO_2, VOC, NH_3 e partículas em suspensão (poluentes), sendo que o conjuntos de parâmetros a medir no interior e no exterior é diferente.

Foi montado um sistema com três dispositivos espacialmente dispersos, que recolhem os parâmetros da qualidade do ar, usando sensores apropriados. Cada dispositivo de monitorização inclui um microcontrolador baseado no ESP32 e sensores adequados aos parâmetros que se pretende medir. Após esta recolha de dados, estes são enviados para um dispositivo central, recorrendo à técnica de modulação de ondas para longas distâncias LoRa. Este dispositivo central, através de comunicação Wi-Fi, envia os dados pelo protocolo de publicação/ subscrição MQTT para um servidor *online*, onde os dados são armazenados e mostrados em tempo real.

Uma vez que os equipamentos devem ter tanto mais autonomia quanto possível, os consumos energéticos são uma preocupação e, por isso, os *sleep modes* dos equipamentos foram abordada juntamente com a ligação de uma célula fotovoltaica ao sistema. Ainda, foram projetadas proteções mecânicas para eventos meteorológicos, tal como chuva ou vento forte.

Os dados obtidos pelo sensor de ozono foram sujeitos a comparação com dados disponibilizados por estações de medição da qualidade do ar próximas para correção da curva característica dos sensores, conforme as intruções do fabricante. O resultado dos testes realizados aos sensores de temperatura indicam que os valores dados por estes são muito sensíveis à exposição à luz solar, e que os dois modelos utilizados neste trabalho apresentam diferentes exatidões. Para os outros parâmetros, não foi possível validar os dados, devido à falta de equipamento de referência.

Keywords: Qualidade do ar, ESP32, LoRa, MQTT, Célula fotovoltaica

Acknowledgements

I would like to express the most sincere words of gratitude to all the people who guided me during this dissertation. A special word:

To Professor Doctor Joaquim Gabriel, my supervisor, who gave me great support, motivation and knowledge, helping me throughout this period;

To Joaquim Silva, Technical Assistant of the Mechanical Engineering Department of FEUP, for the technical support;

To Mehrdad Bahreini, researcher and laboratory colleague, for the companionship and friendship created. To Rúben Madureira, PhD student, for the occasional theoretical support and the revision of this document;

To my friends, especially João Pedro Silva and Guilherme Perestrelo, my special friends for the thorough revision of this document and for the support in improving its quality. Also to my other close friends for the friendship, motivation and emotional support they gave me during my course at FEUP. Also, to the friends from *Leça da Palmeira* and to those that were by my side during my journey in the NEEM FEUP association;

To my family, especially my parents Zaida Sousa and Adão Sousa, for the support and efforts made to offer me better personal and academic formation, which they never had.

Author

"O ar é de todos."
(The air belongs to everyone.)

A popular childish saying

Contents

1 Introduction **1**
- 1.1 Project context . . . 1
- 1.2 Aims and Objectives . . . 2
- 1.3 Document structure . . . 2

2 State of the Art **3**
- 2.1 Air quality . . . 3
  - 2.1.1 Human health concerns . . . 3
  - 2.1.2 Ambient (outdoor) air quality . . . 3
  - 2.1.3 Indoor air quality (IAQ) . . . 8
- 2.2 Smart cities . . . 12
  - 2.2.1 Wireless standards for Smart cities . . . 13
  - 2.2.2 MQTT – Message Queuing Telemetry Transport . . . 19
- 2.3 Sensors for ambient air quality . . . 20
- 2.4 Related work and existing systems . . . 22
- 2.5 Summary . . . 24

3 Architecture of the project **27**
- 3.1 Requirements of the project . . . 27
- 3.2 Answer to the requirements . . . 27
  - 3.2.1 The system . . . 27
  - 3.2.2 Wireless technology . . . 28
  - 3.2.3 The sensors . . . 28
  - 3.2.4 The microcontroller . . . 28
  - 3.2.5 Data management and presentation . . . 29
- 3.3 Proposed solution . . . 30
- 3.4 Summary . . . 31

4 Hardware design **33**
- 4.1 Sensor modules analysis . . . 33
  - 4.1.1 MIKROE-2767 – Ozone 2 click . . . 33
  - 4.1.2 MIKROE-3098 – NO2 click . . . 37
  - 4.1.3 MIKROE-3056 – Air Quality 5 click . . . 40
  - 4.1.4 Shinyei PPD42NS . . . 44
  - 4.1.5 Sensirion SPS30 . . . 45
  - 4.1.6 DFRobot Gravity BME680 . . . 47
  - 4.1.7 Sparkfun SEN-14348 – Environmental Combo . . . 48
- 4.2 Node assembly . . . 49

4.2.1 Node 1 – Outdoor . . . 50
4.2.2 Node 2 – Outdoor . . . 52
4.2.3 Node 3 – Indoor . . . 54
4.2.4 Development of PCB for the Nodes . . . 56
4.3 Power supply of nodes . . . 56
4.3.1 Power consumption calculations of Node 2 . . . 57
4.3.2 Design the photovoltaic solar panel for Node 2 . . . 57
4.3.3 Battery requirements of Node 2 . . . 58
4.4 Development of a case for mechanical protection . . . 58
4.5 Summary . . . 59

5 Software design **61**
5.1 Node 1 (outdoor) . . . 61
5.2 Node 2 (outdoor) . . . 61
5.3 Node 3 (indoor) . . . 61
5.4 Code implementation . . . 62
5.5 Configuration and connection to the MQTT Broker . . . 62
5.6 Creation of the dashboard . . . 63
5.7 Alerts for poor air quality . . . 64
5.8 Summary . . . 65

6 Experimental results and analysis **69**
6.1 Range tests of LoRa . . . 69
6.2 Installation of the system . . . 70
6.3 Visualisation of the results in the dashboard . . . 71
6.4 Analysis of the results . . . 71
6.5 Alert triggering . . . 72
6.6 Other tests . . . 73
6.6.1 Effect of direct sunlight exposure . . . 73
6.6.2 Comparison of temperature sensors . . . 74
6.7 Summary . . . 74

7 Conclusions and Future Work **77**
7.1 Conclusions . . . 77
7.2 Further Work . . . 78

References **79**

A Programming code **83**
A.1 C++ code uploaded in the MCU of the Node 1 . . . 83
A.1.1 `Node1.ino` . . . 83
A.1.2 `MIKROE_2767.h` . . . 85
A.1.3 `MIKROE_2767.cpp` . . . 86
A.1.4 `MIKROE_3098.h` . . . 88
A.1.5 `MIKROE_3098.cpp` . . . 90
A.1.6 `Sparkfun_CCS811_BME280.h` . . . 92
A.1.7 `Sparkfun_CCS811_BME280.cpp` . . . 93
A.1.8 `LoRa_Sender.h` . . . 95
A.1.9 `LoRa_Sender.cpp` . . . 96

A.2 C++ code uploaded in the MCU of the Node 2 98
A.2.1 Node2.ino 99
A.2.2 Gravity_BME680.h 100
A.2.3 Gravity_BME680.cpp 101
A.2.4 LoRa_Sender.h 102
A.2.5 LoRa_Sender.cpp 102
A.3 C++ code uploaded in the MCU of the Node 3 102
A.3.1 Node3.ino 102
A.3.2 MIKROE_3056.h 105
A.3.3 MIKROE_3056.cpp 105
A.3.4 SHINYEI_PPD42.h 109
A.3.5 SHINYEI_PPD42.cpp 110
A.3.6 Sparkfun_CCS811_BME280.h 111
A.3.7 Sparkfun_CCS811_BME280.cpp 111
A.3.8 LoRa_Receiver.h 111
A.3.9 LoRa_Receiver.cpp 111
A.3.10 MQTT_Client.h 114
A.3.11 MQTT_Client.cpp 115

List of Figures

2.1 Smart City Scheme. . . . 13
2.2 Zigbee mesh network . . . 15
2.3 LoRa and LoRaWAN layers represented in OSI model . . . 17
2.4 LoRaWAN architecture . . . 18
2.5 Position of LoRa technology in relation to the others wireless technologies, with regard to bandwidth and range . . . 18
2.6 MQTT, a publish/subscribing message protocol . . . 20
2.7 Different levels of data quality is required for different applications. . . . 20
2.8 Typical sensor network scheme with increasing complexity steps from a sensor element to the wireless sensor network, for air quality monitoring . . . 21
2.9 Examples of different types of air monitoring sensors. . . . 22
2.10 Example of a LoRa and LoRaWAN combination . . . 25

3.1 Examples of MCUs with LoRa capacities. . . . 30
3.2 Architecture of the proposed solution. . . . 31

4.1 MIKROE-2767 marked with key components . . . 34
4.2 SPI communication of ADC, part of MIKROE-2767 . . . 34
4.3 MQ131 characteristic curves. . . . 35
4.4 MQ131 sensor details. . . . 36
4.5 MIKROE-3098 marked with key components. . . . 38
4.6 Sensor SPEC 3SP-NO2 110-508 details. . . . 39
4.7 MIKROE-3056 marked with key components . . . 41
4.8 MiCS-6814 internal circuits. . . . 41
4.9 Curves of sensitivity of the different sensor chips of MiCS-6814. . . . 43
4.10 Shinyei PPD42NS sensor. . . . 45
4.11 Sensirion SPS30. . . . 47
4.12 DFRobot Gravity BME680 sensor details. . . . 48
4.13 Sparkfun SEN-14348 – Environmental Combo. . . . 50
4.14 Assembly of Node 1. . . . 51
4.15 Assembly of Node 2. . . . 53
4.16 Assembly of Node 3. . . . 55
4.17 PCB for Nodes. . . . 56
4.18 Photovoltaic solar panel circuit schematic. . . . 58
4.19 Snapshots of designed case for Node 2. . . . 60

5.1 Screenshot of Arduino IDE . . . 62
5.2 Screenshot of device configuration panel in Ubidots platform, presenting environmental variables. . . . 63

5.3 Screenshot of variable configuration in Ubidots platform, displaying the existing data. . . . 64
5.4 Screenshot of dashboard, displaying the existing widgets provided by Ubidots. . . 64
5.5 Screenshots of configuration of an SMS alert when CO is greater than 2000 ppb for 30 minutes. . . . 65
5.6 Flowcharts for Node 1. . . . 66
5.7 Flowcharts for Node 2. . . . 67
5.8 Flowcharts for Node 3. . . . 68

6.1 Map of the locations used to test the LoRa range . . . 70
6.2 Screenshot of the dashboard displaying results for the last 24 hours. . . . 71
6.3 Screenshot of the dashboard displaying results for the last 24 hours annotated. . . 72
6.4 Data of O_3 hourly concentrations in the Sobreiras-Lordelo do Ouro (in Porto) air quality monitoring station *Agência Portuguesa do Ambiente*. . . . 73
6.5 Screenshot of the SMS sent by Ubidots, when the programmed event triggered. . 73
6.6 Effect of sunlight exposure in temperature measurements. . . . 74
6.7 Comparison of temperature readings of the sensors. . . . 75
6.8 Configuration of the test for the Pt100 temperature sensor. . . . 76

List of Tables

2.1 Particulate matter thresholds by WHO and European Directives 5
2.2 Ozone thresholds by WHO and European Directives 6
2.3 Nitrogen dioxide thresholds by WHO and European Directives 6
2.4 Sulfur dioxide thresholds by WHO and European Directives 7
2.5 Other pollutants thresholds by European Directives 7
2.6 Level classification for some pollutants used by European Environmental Agency 8
2.7 Level classification for some pollutants used by *Agência Portuguesa do Ambiente.* 8
2.8 Indoor air pollutants thresholds by WHO and Portuguese legislation. 11
2.9 Summary about the wireless communication protocols 19
2.10 Related works and existing systems. 22

3.1 List of sensors used in the project. 29
3.2 Specifications for TTGO LORA32 V2.1.6 and TTGO T-Beam V1.1, both produced by LILYGO. 30

4.1 Comparison of O_3 concentrations measured by the sensor and by public station for air quality monitoring (n=4). 37
4.2 Most relevant functions of SPS30 library. 46
4.3 Most relevant functions of BME680 library. 48
4.4 Most relevant functions of CCS811 library. 49
4.5 Most relevant functions of BME280 library. 49
4.6 Pin connection for Node 1. 50
4.7 Pin connection for Node 2. 52
4.8 Pin connection for Node 3. 54
4.9 Worst-case scenario for a Node 2 duty cycle. 57

6.1 Locations used to test the LoRa range. 70

A.1 Organisation of programming code files for the Node 1 83
A.2 Organisation of programming code files for the Node 2 98
A.3 Organisation of programming code files for the Node 3 102

Abbreviations

3G	Third Generation of wireless mobile telecommunications technology
3GPP	Third Generation Partnership Project
4G	Fourth generation of wireless mobile telecommunications technology
AAQ	Ambient Air Quality
ADC	Analogue-Digital Converter
AFE IC	Analog Front-End Integrated Circuit
AQ	Air Quality
AQI	Air Quality Index
BLE	Bluetooth Low Energy
CCTV	Closed-Circuit TeleVision
$\overline{\text{CS}}$	Chip Select (related to SPI)
DC	Direct Current
EAQI	European Air Quality index
FEUP	Faculdade de Engenharia da Universidade do Porto
IAQ	Indoor Air Quality
I^2C	Inter-Integrated Circuit, a serial communication interface
IoT	Internet-of-Things
IP	Internet Protocol
ITS	Intelligent Transportation Systems
GDPR	General Data Protection Regulation
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications, 2G
LAN	Local Area Networks
LED	Light-Emitting Diode
LPWAN	Low-Power, Wide-Area Network
LTE	Long-Term Envolution, marketed as 4G LTE
MAC	Media Access Control
MCU	MicroController Unit
MOX	Metal OXide sensor
MQTT	Message Queuing Telemetry Transport
OPC	Optical Particle Counter
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
pcs/L	Particles per litre
PM	Particulate Matter
PM_x	Particulate Matter with less than x μm
ppb	parts-per-billion (10^{-9})
ppm	Parts-per-million (10^{-6})
PV	PhotoVoltaic solar panel
PWM	Pulse Width Modulation
RF	Radio Frequency
SCK	Serial ClocK (related to SPI)

SCL	Serial CLock (related to I^2C)
SDA	Serial DAta (related to I^2C)
SDO	Slave Data Out (related to SPI)
SMS	Short Message Service
SPI	Serial Peripheral Interface, a serial communication interface
RAM	Random Access Memory
TCP	Transmission Control Protocol
TVOC	Total Volatile Organic Compounds
USB	Universal Serial Bus
VOC	Volatile Organic Compound
WHO	World Health Organization
WSN	Wireless Sensor Network

Glossary

VOC - “Abbreviation for Volatile organic compound. Organic chemical compounds that under normal conditions are gaseous or can vaporise and enter the atmosphere. VOCs include such compounds as methane, benzene, xylene, propane and butane. Methane is primarily emitted from agriculture (from ruminants and cultivation), whereas non-methane VOCs (or NMVOCs) are mainly emitted from transportation, industrial processes and use of organic solvents” [1].

Internet-of-Things - “is the concept of basically connecting any device with an *on* and *off* switch to the Internet (and/or to each other). This includes everything from mobile phones, coffee makers, washing machines, headphones, lamps, wearable devices and almost anything else you can think of. (...) The IoT is a giant network of connected "things" (which also includes people). The relationship will be between people-people, people-things, and things-things.” – Jacob Morgan in Forbes [2].

Wireless Sensor Network - A collection of wireless sensors that may or may not be connected to the internet. "a network of nodes that work in a cooperative way to sense and control the environment surrounding them. These nodes are linked via wireless media. Nodes use this connection to communicate among each other. The architecture of a typical WSN consists of following 3 components: sensor nodes, gateway and observer (user)" [3].

[1] European Environment Agency. VOC — European Environment Agency. URL: https://www.eea.europa.eu/themes/air/air-quality/resources/glossary/voc.

[2] Jacob Morgan. A Simple Explanation Of ’The Internet Of Things’. Forbes, may 2014. URL: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/.

[3] Andrew Meola. What is the Internet of Things How Does IoT Work? - Business Insider, may 2018. URL: https://www.businessinsider.com/internet-of-things-definition.

Chapter 1

Introduction

Concerns about air quality are expanding as more data is available and more studies reinforcing the relationship between air quality and quality of life. This increasing awareness of the problems that can arise from poor air quality is forcing urban managers to create conditions for monitoring air quality, facilitating the identification of sources of air contamination and allowing citizens to be informed, in real-time, about air quality.

> "To date, air pollution – both ambient (outdoor) and household (indoor) – is the biggest environmental risk to health, carrying responsibility for about one in every nine deaths annually. (...) it is a public health emergency" – WHO [1].

This problem, combined with the motivation to better understand smart cities, Internet-of-Things and emerging communication technologies, has led to the development of an air quality monitoring system based in IoT and suitable for urban environments.

1.1 Project context

The development of a system for air quality monitoring was purposed as a Master's Dissertation, included in the Project POCI-01-0247-FEDER-041435 (Safe Cities - *Inovação para Construir Cidades Seguras*), a consortium by Bosch Security Systems S.A. and the University of Porto (Rectorate, the Faculty of Engineering and the Faculty of Sciences). This project is co-financed by the European Union under the European Regional Development Fund, under the Operational Programme for Competitiveness and Internationalisation, COMPETE 2020. The overall objective of the Safe Cities project is to research and develop innovative products and services that contribute to the safety of citizens and the systems and subsystems that interact with them. The potential for data acquisition and storage, intelligent data, intelligence and communications are exploited in multiple ways to create numerous services through the same ecosystem [1].

[1] Faculty of Engineering of the University of Porto. FEUP - Projeto: POCI-01-0247-FEDER-041435 (Safe Cities). URL: https://sigarra.up.pt/feup/en/PROJECTOS_GERAL.FICHA_PROJECTO?P_ID=74436.

1.2 Aims and Objectives

The main objective of this dissertation is to develop a microcontroller-based network system for air quality data acquisition, along with a real-time presentation of this data.

The network should support various devices measuring different air quality parameters, both environmental factors and pollution levels. For a comprehensive understanding of air quality, data should be collected at remote points. As the data collection points are far apart, particular attention should be paid to energy consumption. Furthermore, for real-time data visualisation to be possible, data must be sent to a central network using a wireless network for this purpose.

To achieve this objective, several intermediate tasks were defined:

- Find air quality parameters and sensors to measure them;
- Study wireless communication for data exchange between nodes;
- Select the components for the project, microcontroller and sensors for environmental assessment;
- Programming the microcontroller;
- Design a mechanical case to protect the system against weather events;
- Evaluate power requirements of the system;
- Assembly the prototype and test it.

1.3 Document structure

This document describes the steps performed in this dissertation project and the respective support in developing a system for air quality monitoring. The theoretical content necessary to understand all the concepts used is exposed, and the whole process of development and assembly of the mechanism is presented. Finally, the results of the tests performed are presented along with the respective analysis and improvement proposals.

Chapter 1 exposes the problem and exhibit motivations, aims and objectives of this project.

Chapter 2 introduces some theoretical concepts, evaluating what the system should measure and how it should be done, addressing air quality parameters, smart cities technologies and low-cost sensors for air quality.

Chapter 3 approaches the resolution of the problem, the studied solutions and the proposed architecture for the system.

Chapter 4 study the hardware components used and how they are coupled with the microcontroller.

Chapter 5 presents the working logic behind the programming code.

Chapter 6 analyses the results of the measurements made.

Chapter 7 resumes the conclusions and proposes actions to be developed after this work to improve the quality of the system.

Chapter 2

State of the Art

This state of the art presents a literature review of the most relevant issues concerning this project, air quality monitoring.

2.1 Air quality

In the last years, numerous European organisations declared the right to breathe clean air and air pollution monitoring at the most polluted places [2]. World Health Organization (WHO) also establishes obligations to decision-makers of public health [3] and posted several ambient and indoor air quality guidelines.

2.1.1 Human health concerns

According to WHO reports, every year, 7 million deaths (one in eight deaths) are attributable to the combined effects of indoor and ambient air pollution (3.8 million related to indoor and 4.2 million to ambient air pollution). According to *State of Global Air 2020* report, in 2019, 500 thousand deaths among infants in their first month of life are attributable to air pollution. Furthermore, 90% of the world's population breathes polluted air everyday [4, 5]. Moreover, the WHO's International Agency for Research on Cancer found a correlation between air pollution and cancer of the urinary bladder and lung cancer [6]. Therefore, WHO identifies that air pollution is a critical risk factor for non-communicable diseases, causing about 24% of all adult deaths from heart disease, 25% from stroke, 43% from chronic obstructive pulmonary disease and 29% from lung cancer [1].

2.1.2 Ambient (outdoor) air quality

Due to the advance in the scientific knowledge of sources and the adverse effects of air pollution on human health, it has become increasingly essential to create legislation to control air quality.

[1] World Health Organization. 9 out of 10 people worldwide breathe polluted air, but more countries are taking action, may 2018. URL: https://www.who.int/health-topics/air-pollution.

World Health Organization publications summarise the available research and create thresholds for measurable pollutants to give a base to national and local authorities and to alert the citizens. In 1987, WHO launched first guidelines about air quality, air pollutants and related concerns [7], updated in 1995 [8] and replaced in 2000 by *Second Edition of Air Quality Guidelines* [9], updated in 2005 [10] which indicates reference limits for some pollutants.

Generally, multiple factors are associated with ambient air pollution: dust, industry and energy supply, transport, agricultural practices, household energy and waste management[2].

2.1.2.1 Ambient air quality parameters and standards

The ambient air quality could be assessed by the presence of harmful substances in the atmosphere: gases (ammonia, CO, SO_2, NO_x, methane and chlorofluorocarbons), particulates (organic and inorganic) and biological molecules, both natural and human made.

Human exposure is defined as "the event when a person comes into contact with a pollutant of a certain concentration during a certain period of time". It is determined by pollutants concentrations in environments where people spend their time and, the dose is defined as "the amount of pollution that crosses one of the body boundaries" [11]. There is an established pathway between them:

$$\text{Source} \rightarrow \text{Emissions} \rightarrow \text{Concentrations} \rightarrow \text{Exposure} \rightarrow \text{Dose} \rightarrow \text{Health effects.}$$

Thus, legislated and recommended thresholds are not defined as an instantaneous measured value. Instead, they are defined as an average value for a time interval, both short-term (hourly or daily) and long-term (annual). Generally, long-term levels are stricter since serious health effects are connected to long-term exposure[3,4,5] [10]. The next paragraphs summarise the WHO recommendations [10], and the thresholds defined in *European Directive 2008/50/EC* from European Commission[3]. In Portugal, national legislation *Decreto-Lei n.º 102/2010*, from 23 September, transposes the European Directive and was amended by *Decreto-Lei nº 43/2015*, from 27 March, creating judicial support and determining evaluation method[6,7,8].

Particulate matter (PM) is a complex airborne mixture with diverse components having different chemical and physical proprieties, mostly sulfate, nitrates, ammonia, sodium chloride, black

[2]World Health Organization. What are the sources of air pollution? (infographics). URL: https://www.who.int/health-topics/air-pollution.

[3]European Parliament. Directive 2008/50/EC of the European Parliament and of the Council of 21 May 2008 on ambient air quality and cleaner air for Europe, may 2008. URL: http://data.europa.eu/eli/dir/2008/50/2015-09-18.

[4]United States Environmental Protection Agency. NAAQS Table | Criteria Air Pollutants | US EPA. URL: https://www.epa.gov/criteria-air-pollutants/naaqs-table.

[5]European Environment Agency. European Air Quality Index. URL: https://airindex.eea.europa.eu/Map/AQI/Viewer/#.

[6]Ministério do Ambiente e do Ordenamento de Território. Decreto-Lei n.º 102/2010 de 23 de setembro. *Diário da República*, sep 2010. URL:https://data.dre.pt/eli/dec-lei/102/2010/09/23/p/dre/pt/html

[7]Ministério do Ambiente, Ordenamento de Território e Energia. Decreto-Lei nº 43/2015 de 27 de março. *Diário da República*, mar 2015. URL: https://dre.pt/application/conteudo/66863795.

[8]Agência Portuguesa do Ambiente. APA - Políticas > Ar > Qualidade do Ar Ambiente. URL: https://apambiente.pt/index.php?ref=16&subref=82&sub2ref=316.

carbon, mineral dust, sea salt and water. These components predominantly come from traffic, oceans, dust suspension, industry, biomass burning (large-scale vegetation fires, agricultural burning, residential heating and cooking) and coal/oil burning (coal-fired power plants, marine diesel engines, heating and lighting)[9] [12].

For regulatory and measurement purposes, PM has been classified by aerodynamic diameter once evidence has become available on the risk associated with specific size groups. While PM_{10} - fine particulate matter - are particles with 10 µm or less and can penetrate and lodge deep inside the lungs, $PM_{2.5}$ - coarse particulate matter - are particles with 2.5 µm or less and can penetrate lung barrier and enter the blood system. Prolonged exposure to these particles contributes to the risk of developing cardiovascular and respiratory diseases, as well as lung cancer[9] [10].

Table 2.1 presents the recommended thresholds of particulate matter by the World Health Organization and Directive 2008/50/EU from the European Commission.

Table 2.1: Particulate matter thresholds by WHO [10] and European Directives[3].

	annual mean (µg/m^3)		**24-hour mean (µg/m^3)**	
	WHO*	European Union	WHO*	European Union
PM_{10}	20	40	50	50**
$PM_{2.5}$	10	25	25	N/D

* WHO defined provide interim targets for these values as a transition for recommended values. Shown value are the recommended values;
** this value should not be exceeded more than 35 times a year.

Ozone (O_3) is not directly emitted by primary sources. However, it results from the reaction of sunlight (photochemical reaction) with other pollutants, such as NO_2 from vehicles and industrial emissions and volatile organic compounds (VOC) emitted by vehicles, solvents and industry. It is one of the most significant components of photochemical smog at ground level and, ozone reaches high levels in sunny weather occurring from the nature of the reaction.

Ozone is a harmful component for human health and can cause muscles in the airways to constrict, which results in wheezing and shortness of breath. Besides breathing difficulties, it could cause lung diseases such as asthma, emphysema, chronic bronchitis and a reduction in life expectancy[10] [10].

Table 2.2 presents the recommended thresholds by World Health Organization and European Commission.

Nitrogen dioxide (NO_2) is the fundamental source of nitrate aerosols, a substantial fraction of $PM_{2.5}$ and, in the presence of ultraviolet light, triggers ozone formation. Emission of NO_2

[9] World Health Organization. Ambient (outdoor) air pollution, may 2018. URL: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-qualityand-health.

[10] United States Environmental Protection Agency. Ground-level Ozone Basics | Ground-level Ozone Pollution | US EPA. URL: https://www.epa.gov/ground-level-ozone-pollution/ground-level-ozone-basics.

Table 2.2: Ozone thresholds by WHO [10] and European Directives[3].

8-hour mean ($\mu g/m^3$)	
WHO	European Union
100	120*

* this value should not be exceeded 25 days averaged over 3 years

results of natural and human-based activities, mainly combustion processes from heating, power generation and engines in vehicles and ships[9].

Epidemiological studies have shown a clear relation between long-term exposition to NO_2 and asthma in children [13], lung function decline [14] and even identified long-term exposure to NO_2 as the most significant factor to fatality caused by COVID-19, an infectious disease caused by SARS-CoV-2 [15].

Table 2.3 presents the recommended thresholds of NO_2 by World Health Organization and European Commission.

Table 2.3: Nitrogen dioxide thresholds by WHO [10] and European Directives[3].

annual mean ($\mu g/m^3$)		1-hour mean ($\mu g/m^3$)	
WHO	European Union	WHO	European Union
40	40	200	200*

* this value should not be exceeded more than 18 times a year.

Sulfur dioxide (SO_2) is a colourless gas with a sharp odour and results from volcanoes eruptions, burning of fossil fuels (coal and oil) and the smelting of mineral ores containing sulfur.

Sulfur dioxide exposure affects the respiratory system and the function of the lungs. It causes eye irritation, coughing due to the inflammation of the respiratory tract, mucus secretion, aggravation of asthma and bronchitis and makes people prone to infections of the respiratory tract[9]. Hospital admissions related to cardiovascular diseases increase on days with higher SO_2 levels [16]. Furthermore, SO_2 combined with water leads to the formation of sulfuric acid, Equation 2.1, "acid rains", a cause of deforestation and ecological disturbance[11].

$$SO_2\,(g) \xrightarrow{O_2} SO_3\,(g) \xrightarrow{H_2O} H_2SO_4 \tag{2.1}$$

Table 2.4 presents the recommended thresholds of SO_2 by World Health Organization and European Commission.

[11] United States - Environmental Protection Agency. What is Acid Rain? | Acid Rain | US EPA. URL: https://www.epa.gov/acidrain/what-acid-rain.

Table 2.4: Sulfur dioxide thresholds by WHO [10] and European Directives[3].

24-hour mean (µg/m^3)		1-hour mean (µg/m^3)	10-minute mean (µg/m^3)
WHO	European Union	European Union	WHO
20	125*	350**	500

* this value should not be exceeded more than 3 times a year;

** this value should not be exceeded more than 24 times a year.

Other components European Union's Directive 2008/50/EU applied to all Member States, restrict some other components[3], as seen in Table 2.5.

Table 2.5: Other pollutants thresholds by European Directives[3].

	annual mean	8-hour mean
Lead (Pb)	0.5 µg/m^3	N/D
Carbon monoxide (CO)	N/D	10 mg/m^3
Benzene	5 µg/m^3	N/D
Arsenic (As)	6 ng/m^3	N/D
Cadmium (Cd)	5 ng/m^3	N/D
Nickel (Ni)	20 ng/m^3	N/D
Polycyclic Aromatic Hydrocarbons	1 ng/m^3	N/D

2.1.2.2 Air Quality Index (AQI)

Government agencies use air quality indices (AQI) to provide simple air quality information to the general public. Different indices are used by countries, according to national air quality standards and using different calculation methods, e.g., European Air Quality Index (Europe), *Índice QualAR* (Portugal), Air Quality Index (United States), Air Quality Health Index (Canada)[5,12,13,14].

Europe - European Air Quality Index European Air Quality Index (EAQI), provided by European Environmental Agency, is based on concentration values of key pollutants - PM_{10}, $PM_{2.5}$, O_3, NO_2 and SO_2. This index is calculated hourly from more than 2000 air quality monitoring stations across Europe, reflecting the short-term air quality situation for each monitoring station. It was not designed as a tool to check compliance with air quality standards.

Traffic stations collect data of NO_2 and PM (PM_{10}, $PM_{2.5}$ or both), once SO_2 could be high in localised areas distorting evaluation and O_3 is, generally, low there. In industrial or background stations, EAQI is calculated using at least NO_2, O_3 and PM (PM_{10}, $PM_{2.5}$ or both) data. For

[12] Agência Portuguesa do Ambiente. QualAR - Qualidade do AR. URL: https://qualar.apambiente.pt/node/metodo-calculo-indices.

[13] AQI Basics | AirNow.gov. URL: https://www.airnow.gov/aqi/aqi-basics/.

[14] Air Quality Health Index - Canada.ca. URL: https://www.canada.ca/en/environment-climate-change/services/air-quality-health-index.html.

O_3, NO_2 and SO_2, it is used 1-hour mean concentrations and, for PM it is used 24-hour mean concentrations[5]. The index corresponds to the worst level for any of five pollutants, according to the Table 2.6.

Table 2.6: Level classification for some pollutants used by European Environmental Agency[5].

Classification	**PM_{10}** µg/m^3	**$PM_{2.5}$** µg/m^3	**NO_2** µg/m^3	**O_3** µg/m^3	**SO_2** µg/m^3
Good	0-20	0-10	0-40	0-50	0-100
Fair	21-40	11-20	41-90	51-100	101-200
Moderate	41-51	21-25	91-120	101-130	201-350
Poor	51-100	26-50	121-230	131-240	351-500
Very poor	101-150	51-75	231-340	241-380	501-750
Extremely poor	151-1200	76-800	341-1000	381-800	751-1250

Portugal - *Índice QualAR* *Agência Portuguesa do Ambiente* (Portuguese Agency for Environment) uses a level classification for measured values, updated in 2019, classifying values of different pollutants in a colour-based scale varying from "Very Good" to "Bad", updated with Portuguese and European legislation and WHO recommendations[12]. The index corresponds to the worst level for any of five pollutants, according to Table 2.7.

Table 2.7: Level classification for some pollutants used by *Agência Portuguesa do Ambientes*[12].

Classification	**PM_{10}** µg/m^3	**$PM_{2.5}$** µg/m^3	**NO_2** µg/m^3	**O_3** µg/m^3	**SO_2** µg/m^3
Muito Bom (Very Good)	0-20	0-10	0-40	0-80	0-100
Bom (Good)	21-35	11-20	41-100	81-100	101-200
Médio (Medium)	36-50	21-25	101-200	101-180	201-350
Fraco (Weak)	51-100	26-50	201-400	181-240	351-500
Mau (Bad)	101-1200	51-800	401-1000	241-600	501-1250

2.1.3 Indoor air quality (IAQ)

Indoor air quality has not always been a relevant topic for society. Only 20 years ago, WHO explored this and launched a series of guidelines related to air quality enunciating the most dangerous substances based on toxicological and epidemiological data availability. It is essential because activity pattern data studies estimate people spend almost 90% of their time indoors [17], which means there is a massive exposure to indoor pollutants.

The first editions (1987 and 2000) of *Air Quality Guidelines* from WHO [7, 9] were generic about outdoor and indoor air quality parameters and only specifies tobacco smoke, synthetic vitreous fibres and radon as significant indoor pollutants. *The Air Quality Guidelines: Global Update 2005* incorporates a complete chapter about indoor air quality, including particulate matter ($PM_{2.5}$ and PM_{10}) limit values without specifying any values for other air compounds due to weak scientific evidence [10]. In 2009, WHO launched *Guidelines for Indoor Air Quality: dampness and*

mould, identifying the excess of moisture on almost all indoor materials lead to the growth of microbes, such as mould, fungi and bacteria that emit spores, cells, fragments and volatile organic compounds into indoor air which increases the risk of respiratory symptoms, respiratory infections and exacerbation of asthma [18]. The year after, *Guidelines for Indoor Air Quality: Selected Pollutants* provide a review of the latest scientific evidence of sources and effects of several chemicals (benzene, carbon monoxide, formaldehyde, naphthalene, nitrogen dioxide, polycyclic aromatic hydrocarbons, radon, trichloroethylene, tetrachloroethylene), providing reference values for them [19]. In 2014, the *Indoor Air Quality Guidelines: Household Fuel Combustion* focused on pollutants created by the burn of solid fuels (wood, animal dung, charcoal and coal), in inefficient stoves for cooking and heating, and of kerosene for heaters and lamps, a problem affecting 3 billion people, mostly in low- and middle-income countries [20].

2.1.3.1 Sources of indoor air pollution

As mentioned at the beginning of this section, people spend most of their time indoors, representing the necessity of identifying indoor air pollution sources to tackle them. The next paragraphs present some sources of indoor air pollution.

Chemicals Harmful and synthetic chemicals used in cleaning products, carpets and furnishings can damage the liver, kidneys and nervous system, cause cancer, headaches and nausea and irritate the eyes, nose and throat [21].

Radon Radon occurs naturally in soil and rocks by the decay of radium and can lead to lung cancer [19].

Particulate matter Coarse, fine and ultrafine suspended particles cause adverse health effects, including respiratory and cardiovascular systems. Particles may come from outdoor pollution, but can also form indoors by the burning of fuels for heating, cooking and lighting [10, 4].

Microbes Bacteria, fungi and moulds can grow indoor when moisture is available. Exposure can cause respiratory problems, allergies and asthma and affect the immune system [18].

Allergens Pets, pests, house dust mites, cockroaches and, in inner cities, in particular, mice are essential sources of allergens, leading to diseases of the airways, rhinitis and asthma. Contact with these sources is dependent on the environment type, and cultural habits [21].

Tobacco smoke Tobacco smoke exposure can exacerbate respiratory problems, cause lung cancer, headaches, coughs and sore throats [7], but smoking bans almost eliminated this source of indoor air pollution[15].

[15] European Environment Agency. Indoor air quality — European Environment Agency, dec 2019. URL: https://www.eea.europa.eu/signals/signals-2013/articles/indoor-air-quality.

Ventilation and Ambient Air Quality Natural or artificial ventilation is one of the most significant factors determining IAQ. Poorly aired offices can affect work productivity [22] and have a drastic impact on students' cognitive abilities in schools [23]. Furthermore, ambient air quality influences IAQ through ventilation, bringing pollutants into indoor spaces that could be fatal in high doses [24].

Humidity There is an optimal level of humidity. Low humidity can cause eye irritation, dry skin, and rashes, while high humidity results in water damage and mould problems and favours the growth of dust mites [21, 18].

Temperature The temperature could cause discomfort, very cold or hot temperatures affect respiratory health, diabetes management, and symptoms of mental health disorders [25]. Besides, high temperatures aggravate the effects of insufficient humidity.

2.1.3.2 Indoor air quality parameters and standards

The knowledge about air pollution sources, Section 2.1.3.1, provides insight into the harmful effects of pollutants on human health. Epidemiological research supports WHO recommendations and the legislation across the world about indoor air quality defining thresholds for some pollutants based on the average value for a time interval.

In 2006, the Portuguese Government created a law that made indoor air quality inspections mandatory for commerce and service buildings, according to *Decreto-Lei n.º 79/2006*, from 4 April[16].

In 2010, the European Parlament and the Council of European Union approved *Directive 2010/31/EU*, about the energy performance of buildings, giving liberty to national authorities to use other factors besides thermal characteristics, including indoor air quality, to calculate the energy performance of buildings[17]. Although most European countries focused on meeting the Directive's energy requirements, Portugal has chosen to incorporate IAQ requirements[18] [26], publishing *Decreto-Lei n.º 118/2013*, from 20 August, about buildings' energy efficiency and supporting *Portaria n.º 353-A/2013*, from 4 December[19]. That ordinance stipulated protective thresholds for indoor air pollutant concentrations and delegated supervision issues to *Inspeção-Geral da Agricultura, do Mar, do Ambiente e do Ordenamento do Território* (General Inspection of Agriculture,

[16] Ministério das Obras Públicas, Transportes e Comunicações. Decreto-Lei n.º 79/2006, de 4 de abril, apr 2006. URL: https://data.dre.pt/eli/dec-lei/79/2006/04/04/p/dre/pt/html.

[17] European Parliament and Council of the European Union. Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings, jun 2010. URL: http://data.europa.eu/eli/dir/2010/31/oj.

[18] Agência Portuguesa do Ambiente. APA - Políticas > Ar > Qualidade do Ar Interior > Enquadramento legal. URL: https://apambiente.pt/index.php?ref=16&subref=82&sub2ref=319&sub3ref=339.

[19] Ministério da Economia e do Emprego. Decreto-Lei n.º 118/2013, de 20 de agosto. *Diário da República*, 159:4988–5005, aug 2013. URL: https://dre.pt/application/conteudo/499237.

Sea, Environment and Territorial Planning) and assessment methodology to *Agência Portuguesa do Ambiente* and *Direção-Geral de Saúde* (Directorate-General for Health)[20] [27].

In the European Union, only Belgium, Finland, Lithuania, Portugal and France (for benzene, formaldehyde, carbon dioxide and tetrachloroethylene) have recommendations with legal value for indoor air quality. Furthermore, there is no Directive from the European Union defining thresholds for indoor air pollutants [26].

Table 2.8 exposes reference maximum values for PMs, carbon dioxide, carbon monoxide, ozone, formaldehyde, total VOC, according to WHO guidelines and Portuguese legislation.

Table 2.8: Indoor air pollutants thresholds by WHO and Portuguese legislation.

	Portuguese legislation[20]	**WHO guidelines**
PM_{10} $\mu g/m^3$	50 (8-hour mean)	50 (8-hour mean) 20 (1-year mean) [10, 20]
$PM_{2.5}$ $\mu g/m^3$	25 (8-hour mean)	25 (24-hour mean) 10 (1-year mean) [10, 20]
Benzene	5 $\mu g/m^3$ (8-hour mean)	0.17 $\mu g/m^3$ (UR/lt) 10^{-6} [19]
Formaldehyde $\mu g/m^3$	100 (8-hour mean)	100 (30-minute mean) [19]
CO mg/m^3	10 (8-hour mean)	100 (15-minute mean) 35 (1-hour mean) 10 (8-hour mean) 7 (24-hour mean) [19]
CO_2 mg/m^3	2250 (8-hour mean)	N/D
NO_2 $\mu g/m^3$	N/D	200 (1-hour mean) 40 (1-year mean) [19]
Radon Bq/m^3	1	100 [19, 28]
Naphthalene $\mu g/m^3$	N/D	10 (1-year mean) [19]
PAHs (as BaP)	N/D	0.012 ng/m^3 (UR/lt) 10^{-6} [19]
Styrene $\mu g/m^3$	260 (8-hour mean)	N/D
Tetrachloroethylene $\mu g/m^3$	250 (8-hour mean)	250 (1-year mean) [19]
Trichloroethylene	25 $\mu g/m^3$ (8-hour mean)	2.3 $\mu g/m^3$ (UR/lt) 10^{-6} [19]

[20] Ministérios do Ambiente, Ordenamento do Território e Energia, da Saúde e da Solidariedade, Emprego e Segurança Social. Portaria n.º 353-A/2013, de 4 de dezembro. *Diário da República*, dec 2013. URL: https://data.dre.pt/eli/port/353-a/2013/12/04/p/dre/pt/html.

Table 2.8 continued from previous page

	Portuguese legislation[20]	WHO recommendations
Toluene µg/m^3	250 (8-hour mean)	N/D
Total VOC µg/m^3	600 (8-hour mean)	N/D

Abbreviations: UR unit-risk, lt - lifetime, PAHs - Polycyclic Aromatic Hydrocarbons; BaP - Benzo[a]pyrene; VOC - Volatile Organic Compounds; N/D - Not defined.

2.2 Smart cities

A smart city is a city that uses technology (read about IoT in Glossary at the page xxi) to monitor city data (e.g., traffic, lighting, parking, public transportation, **air quality**, fire detection, smart farming), communicating and exchanging data with other devices or a global system, and storing data in a cloud or a server, as illustrated in Figure 2.1.

A city can use the collected data to provide services and solve community problems, such as improving transportation, accessibility, social services and promoting sustainability. Real-time monitoring enables immediate action to detect and solve city problems (as a fire or leak detection on water distribution[21] [29]) by the authorities or could be used for daily planning by the citizens. Data management of a smart city, using IoT devices, consists of four steps:

1. Collection – data is gathered in real-time through sensors.
2. Analysis – collected data is assessed to draw meaningful insights.
3. Communication – the resulting insights are communicated with the decision-makers through communication networks.
4. Action – Cities use the communicated insights to create solutions, optimise operations, evaluate management, and improve citizens' quality of life [22].

Additionally, city managers should guarantee data privacy and security by respecting privacy laws such as the General Data Protection Regulation (GDPR), in force in the European Union. This legislation guarantee that citizens are clarified about the purpose of data collection, how data is used and how long the data will be retained for[23] [30].

[21] Echologics. Smart City connectivity for fixed leak detection, may 2018. URL: https://www.echologics.com/resources/articles/smart-city-connectivity-fixed-leak-detection/.

[22] Margaret Rouse. What is a Smart City?, jul 2020. URL: https://internetofthingsagenda.techtarget.com/definition/smart-city.

[23] Jonathan Andrews. How the EU's new data laws will affect Smart City development, aug 2018. URL: https://news.itu.int/eu-gdpr-smart-cities/.

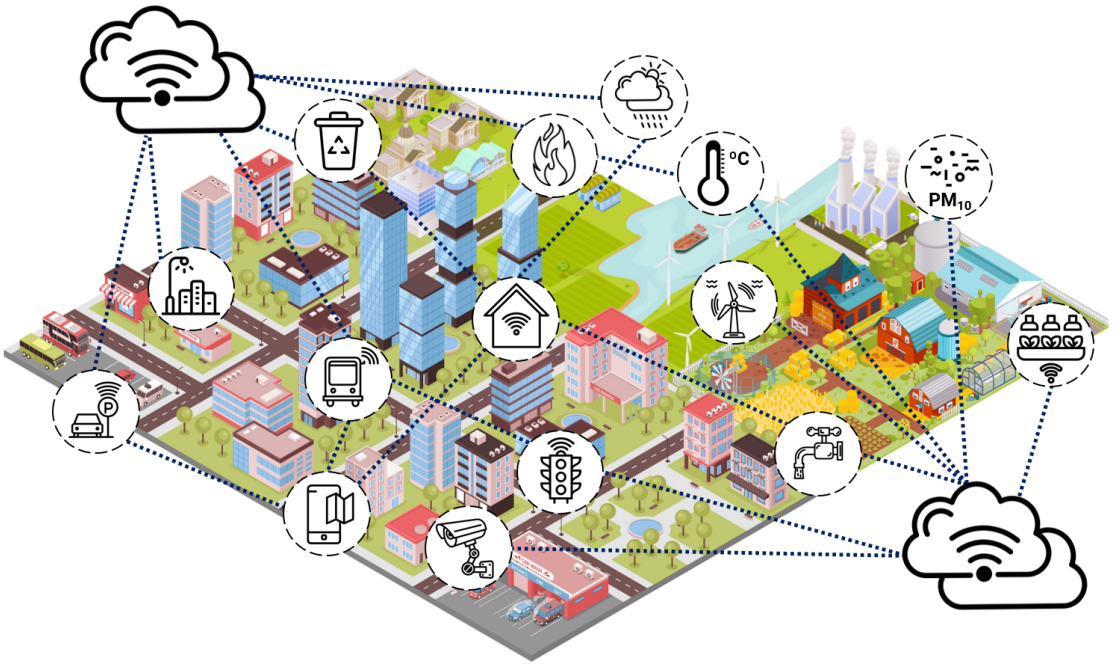


Figure 2.1: Smart City Scheme.

2.2.1 Wireless standards for Smart cities

Communication is an essential part of a Smart City and, without a reliable connection structure, thousands or millions of IoT devices scattered across the city would be defunct. Given that cities comprise several different sectors (e.g., transport, resource management, air quality), different protocols could be combined[23], which could be separated based on the range of the communication: short and wide-range communication protocols. The most prominent short-range technologies include Zigbee, Bluetooth, and Wi-Fi, whereas for long-range communication, protocols such as Global System for Mobile Communication (GSM), Long-Term Evolution (LTE, marketed as 4G LTE), and LoRaWAN are the most well-known[24].

2.2.1.1 Cellular - GSM, 3G, 4G LTE, 5G NR

3GPP (Third Generation Partnership Project) is the global standardisation forum behind the evolution and maintenance the cellular systems: GSM (Global System for Mobile Communications), UMTS (Universal Mobile Telecommunications System), LTE (Long-term Evolution, marketed as 4G LTE) and the 5th generation of cellular radio access technology (5G), described as *New Radio* (NR).

The last releases of the 3GPP (in particular for 4G LTE and 5G NR) cover massification of IoT devices (smart metering, asset management), broadband IoT applications (fleet management,

[24]Prasanth Aby Thomas. Wireless protocol for smart cities: Is one better than the other? *asmag.com*, may 2018. URL: https://www.asmag.com/showpost/25368.aspx.

drones, Augmented Reality/Virtual Reality (AR/VR)), Critical IoT applications (Intelligent Transportation Systems, traffic safety and control, and smart grid), and Industrial Automation IoT (collaborative robotics). All these applications are possible due to extremely low latency, ultra-high reliability, wide-area coverage characteristics[25] [26].

However, power consumption is high when compared with LPWAN (Low Power, wide area networks) and there are costs associated with connecting the devices to RAN (Radio access networks), contracted through data plans with operators [31].

2.2.1.2 Zigbee

Zigbee protocol was created and maintained by the Zigbee Alliance under the IEEE 802.15.4 standard. The Alliance designed this protocol for low-cost and small personal networks and applications that require low data-rate, longer battery life, and secure networking devices. Zigbee is based on a flexible self-organising (self-forming and self-healing) mesh network and supports ultra-low-power devices and transmissions.

Zigbee is mainly used in commercial applications and is a widely used solution for major smart home ecosystem providers such as Amazon Echo Plus, Samsung SmartThings, and IKEA[27].

The constituent devices classify into three types, represented in Figure 2.2:

- Coordinator (ZC) - the network's source and the interface with other networks. The ZC could store information of the network and manage security keys and access (e.g., Amazon Echo);
- Router (ZR) - receives and forwards data to other devices on the network (example: water meter, reading usage and then transmitting data to a field engineer's handheld device);
- End Device (ZED) - performs a function for an application and could receive and respond to its parent node (ZC or ZR) but cannot talk with other devices. This relationship enables the node to rest and preserve battery life (e.g., light switch)[28].

[25] Ericsson. Cellular IoT Evolution for Industry Digitalization. Technical Report January, Ericsson, jan 2019. URL: https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-iot-evolution-for-industry-digitalization.

[26] Brian Ray. Examining Cellular IoT: Cost, Battery and Data, aug 2015. URL: https://www.link-labs.com/blog/cellular-iot.

[27] Zigbee Alliance. Zigbee - Zigbee Alliance. URL: https://zigbeealliance.org/solution/zigbee/.

[28] Assured Systems. What is a ZigBee Network? A Concise Overview. URL: https://www.assured-systems.com/uk/news/article/what-is-a-zigbee-network/.

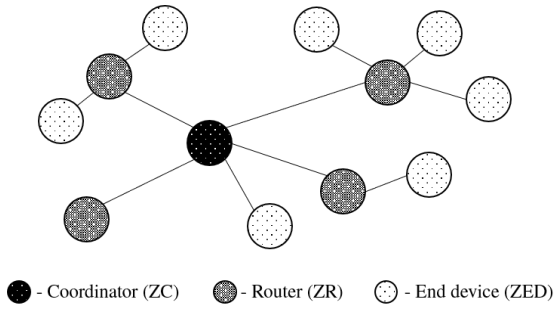


Figure 2.2: Zigbee mesh network[28].

2.2.1.3 Bluetooth

Bluetooth is a communication protocol regulated by Bluetooth SIG and under the IEEE 802.15.1 standard. Bluetooth SIG calculated that 4 billion Bluetooth devices were shipped in 2019[29].

This protocol was designed for a wide range of applications, such as low-energy or mesh applications. Bluetooth represents a solution for phones, tablets and PCs, audio, automotive, smart building, smart industry (powering the Industry 4.0), smart cities, smart homes and Indoor Positioning Systems (using beacons). There are applications in smart cities context in lightning, security, traffics sensors[30,31].

Bluetooth Classic *Classic* Bluetooth uses basic rate/enhanced data rate (BR/EDR), intended for high-throughput and high-duty-cycle applications. Classic Bluetooth represents a good solution for audio and video streaming, for example. Latest versions of Bluetooth, Bluetooth 5.0 and 5.2 technologies enables data rates up to 54 Mb/s and a range of 40 metres and use techniques that reduce the risk of interference in Wi-Fi and LTE networks[32].

Bluetooth Low Energy - BLE Bluetooth SIG introduced Bluetooth Low Energy (BLE) as part of the Bluetooth 4.0 Core Specification in 2010. BLE has a low data rate, lowest possible power consumption and optimised for low-complexity, low-cost and low-bandwidth wireless applications. Adding to this, BLE is compatible with most personal devices, such as smartphones and computers[32].

[29] Bluetooth SIG. 2020 Bluetooth Market Update . Technical report, Bluetooth SIG, Kirkland, Washington, United States of America, mar 2020. URL: https://www.bluetooth.com/bluetooth-resources/2020-bmu/.

[30] Bluetooth SIG. Smart City. URL: https://www.bluetooth.com/learn-about-bluetooth/markets/smart-city/.

[31] Quuppa. Bluetooth AoD as the Technology of Choice for Indoor Positioning Systems (IPS) | RealTime Locating System (RTLS), may 2020. URL: https://quuppa.com/bluetooth-aod-as-the-technology-of-choice-for-indoor-positioning-systems-ips/.

[32] Bluetooth SIG. Bluetooth Core Specification Version 5.2. Technical report, Bluetooth SIG, dec 2019. URL: https://www.bluetooth.com/specifications/bluetooth-corespecification/.

Bluetooth Mesh Networking Bluetooth SIG developed specifications enabling an interoperable many-to-many mesh networking solution for Bluetooth Low Energy, designed to control, monitor, and automation systems with tens, hundreds or more devices to ensure a reliable and secure communication[33].

2.2.1.4 Wi-Fi

Wi-Fi, a trademark of Wi-Fi Alliance, is a family of wireless network protocols, based on IEEE 802.11 standards, commonly used for Local Area Networks (LAN) and Internet access. In 2019, 3.05 billion Wi-Fi devices were shipped, and there is a prevision of increasing this number every year [32]. Also, Wi-Fi technology carries about half the Internet traffic and ten times more than cellular[34].

Over the years, new IEEE standards – IEEE 802.11 (b/a/g/n/ac/ax) – were developed allowing higher maximum data rates and new working frequencies. Wi-Fi technologies using frequencies bellow 1GHz are being developed to provide better ranges and better performance with different materials and are well-suited for low data payloads, such as sensors and control devices. Other frequencies (2.4 and 5 GHz) are being specially used for multimedia and Internet connectivity[34].

2.2.1.5 LoRa and LoRaWAN

LoRa (the abbreviation of Long Range) is a spread spectrum modulation technique derived from chirp spread spectrum (CSS) for Low-power, wide-area networks (LPWAN). The patent was published in 2014[35] and is owned by the American company Semtech Corporation. LoRa is a physical layer and Lora-based devices translate data into RF signals and enable long-range over the air communications in the unlicensed radio spectrum in the Industrial, Scientific and Medical (ISM) sub-gigahertz frequency band, 433 and 868 MHz for Europe, 915 MHz for North America, achieving communications up to 5 km in urban areas and up to 15 km or more in rural areas and in line of sight (the current record is 832 km[36]). Due to the modulation technique and small amounts of data, LoRa requires low power which allows the batteries to last 10 years or even more[37].

LoRaWAN is being developed and promoted by LoRa Alliance, a nonprofit association founded in 2015, and it is an LPWAN protocol that uses LoRa physical layer. LoRaWAN is deployed in a star topology where gateways relay messages between end-devices and a central network server.

[33] Bluetooth SIG. Mesh Networking Specifications. Technical report, Bluetooth SIG, jan 2019. URL: https://www.bluetooth.com/specifications/mesh-specifications/.

[34] Kelly Davis-Felner. Wi-Fi and the Rise of Smart Cities. TMCnet, oct 2015. URL: https://www.tmcnet.com/voip/departments/articles/411301-wi-fi-the-rise-smart-cities.htm.

[35] Olivier B. A. Seller and Nicolas Sornin. Low power long range transmitter, feb 2014. URL: https://patents.google.com/patent/EP2763321A1/en.

[36] The Things Network Global Team. LoRa World Record Broken: 832km/517mi using 25mW, apr 2020. URL: https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1.

[37] Semtech Corporation. LoRa and LoRaWAN: A Technical Overview. Technical report, dec 2019. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/.

The gateways are connected to the network server via standard IP connections and act as a transparent bridge, converting RF packets to IP packets and vice versa. *LoRaWAN Specification* defines the (LoRa) device-to-structure parameters, network architecture and the communication protocol (LoRaWAN) enabling interoperability between different manufacturers [38]. LoRa and LoRaWAN layers are represented in OSI model in Figure 2.3 and LoRaWAN architecture is represented in Figure 2.4.

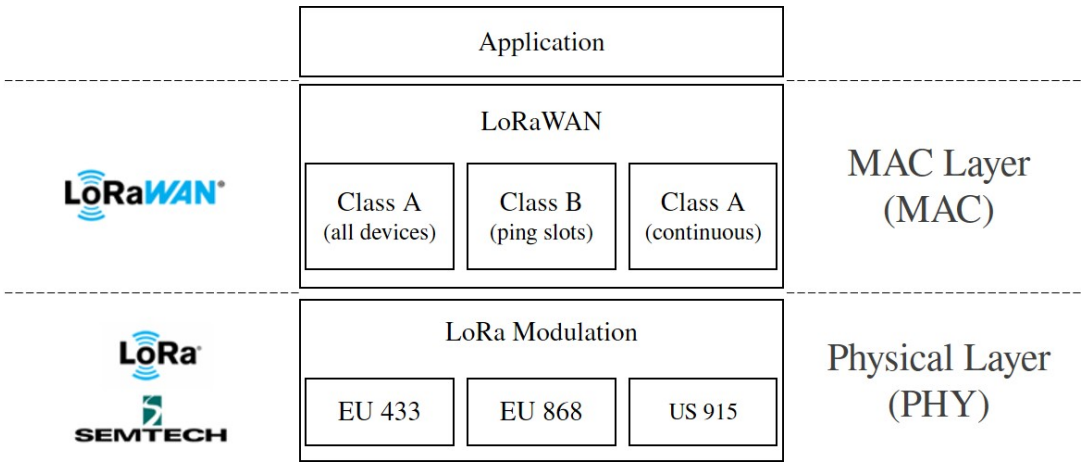


Figure 2.3: LoRa and LoRaWAN layers represented in OSI model[38].

LoRaWAN is composed of:

- End device/node - an object with an embedded low-power communication device;
- Gateway - antennas that receive broadcast from end devices and send back messages;
- Network server - servers that route messages from end devices to applications and back;
- Application - a piece of software, running on a server. For example, to view data, control end nodes, create notifications[39].

LoRa devices and LoRaWAN protocol enable a solution for applications that require long-range or deep in-building communication among many devices with low power requirements and that collects small amounts of data [37,38,40]. The Figure 2.5 shows the position of LoRa in relation to bandwidth and range. Some known projects of LoRa/LoRaWAN are:

- environmental monitoring and pollution control [33, 34];
- water leak detection and monitoring water distribution networks, using LoRaWAN[21];
- helping to fight COVID-19, monitoring medical devices and verifying safety measures (such as social distancing), with LoRaWAN[41];

[38] Semtech Corporation. SEMTECH SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver, 2019

[39] The Things Industries. LoRaWAN Architecture | The Things Network. URL: https://www.thethingsnetwork.org/docs/lorawan/architecture.html.

[40] Semtech Corporation. What is LoRa? | DEVELOPER PORTAL. URL: https://lora-developers.semtech.com/get-started/what-is-lora.

[41] LoRa Alliance. How LoRaWAN® Can Help Fight COVID-19, 2020. URL: http://pages.services/pages.lora-alliance.org/covid-19-lorawan-solutions/.

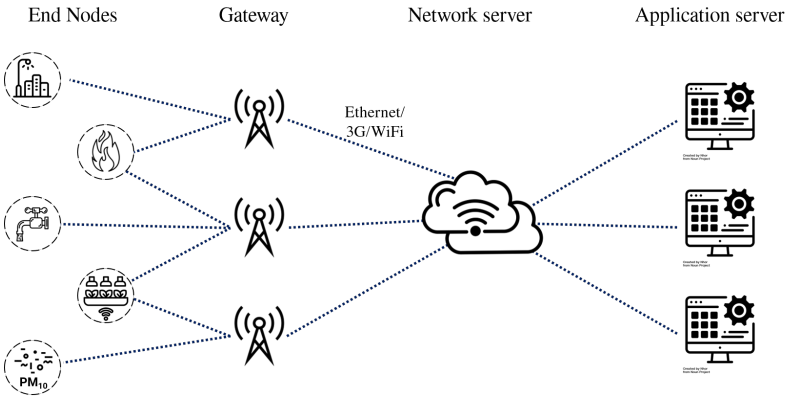


Figure 2.4: LoRaWAN architecture[39].

- disaster prevention using an open-source mesh LoRa network – disaster.radio[42];
- outdoor sports (hiking, pilot, skiing) and secure long-range communication within groups without depending on cellular providers, using LoRa[43];
- geolocation using trilatering positions and timestamps from gateways, using LoRa [35].

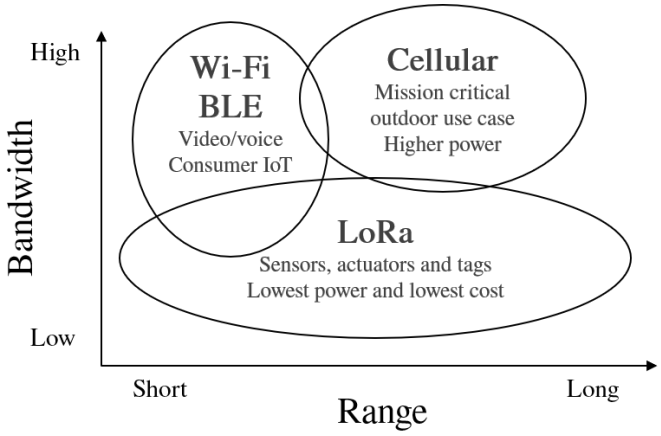


Figure 2.5: Position of LoRa technology in relation to the others, with regard to bandwidth and range[40].

2.2.1.6 Resume of Wireless standards

In the Table 2.9 there are presented the characteristics about the main wireless communication protocols.

[42]Disaster.radio. disaster.radio | a disaster-resilient communications network powered by the sun. URL: https://disaster.radio/.

[43]Meshtastic. What is Meshtastic? | Meshtastic. URL: https://www.meshtastic.org/.

Table 2.9: Summary about the wireless communication protocols[32] [36].

Technology	Operating frequency	Date rate	Coverage	Latency (ms)	Power usage	Use cases
Wi-Fi (IEEE 802.11a/b/g/n)	2.4 and 5GHz	54 Mb/s 6.75 Gb/s	140 m 100 m	46	Medium	Metering, waste management, energy automation, CCTV, personal use (internet access)
Bluetooth (IEEE 802.15.1)	2.4 GHz	54 Mb/s	10 m	100	Low	Smart industry, personal use (audio, mobile phones)
Zigbee (IEEE 802.15.4)	2.4 GHz 868 and 915 MHz	250 kb/s	100 m	16	Low	Smart metering, indoor e-healthcare, home automation
IEEE 802.11p	5.85-5.925 GHz	6 Mb/s	1000 m		Low	Vehicular communication, infotainment, ITS (V2V/V2I)
6LoWPAN	2.4 GHz 868 and 915 MHz	250 kb/s	100 m		Low	ITS, smart metering, logistics
LoRa LoRaWAN	433, 868 and 915 MHz	50 kb/s	2-5 km		Low	ITS, smart metering, waste management
GSM/GPRS	850, 900, 1800 and 1900 MHz	80-384 kb/s	5-30 km	1500-3000	High	ITS, smart metering, energy management, logistics, personal use (mobile communication)
3G	850 MHz	3 Mb/s	5-30 km	100	High	ITS, smart metering, energy management, logistics, personal use (mobile communication)
(4G) LTE LTE Advanced	700, 750, 800, 1900 and 2500 MHz	1 Gb/s 500 Mb/s	5-30 km	25	High	ITS, smart metering, energy management, logistics, personal use (mobile communication, internet access)
5G	700 and 900 MHz, 1.8, 2.1, 2.6 and 3.6 GHz	7.5 Gb/s	500 m	8-12	High	ITS, smart metering, critical and medical procedures, industry, personal use (internet access)

Abbreviations: ITS - Intelligent Transportation Systems; V2I - Vehicle to Infrastructure.

2.2.2 MQTT – Message Queuing Telemetry Transport

MQTT is a standard messaging protocol running over TCP/IP based on publish/subscribe method for transporting messages between devices. Due to being lightweight and small network bandwidth, it is ideal for the Internet-of-Things and smart cities applications, connecting remote devices.

This protocol implies two different types of identities: the message broker and the client. The broker is responsible for receiving, filtering and sending the messages from and to the clients. The broker could run locally or in a server, with existing open-source options available at the market.

The clients are any device that runs an MQTT library and connects to an MQTT broker over a network.

The information is organised into topics. When a client has new data available to distribute, it publishes (sends) this data to the broker on a specific topic. The broker sends this data to other devices that subscribed to that topic[44,45]. Figure 2.6 presents a flowchart of this publishing/subscribing process.

[44]MQTT. MQTT - The Standard for IoT Messaging. URL: https://mqtt.org/

[45]The HiveMQ Team. Client, Broker / Server and Connection Establishment - MQTT Essentials: Part 3, jul 2019. URL: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/.

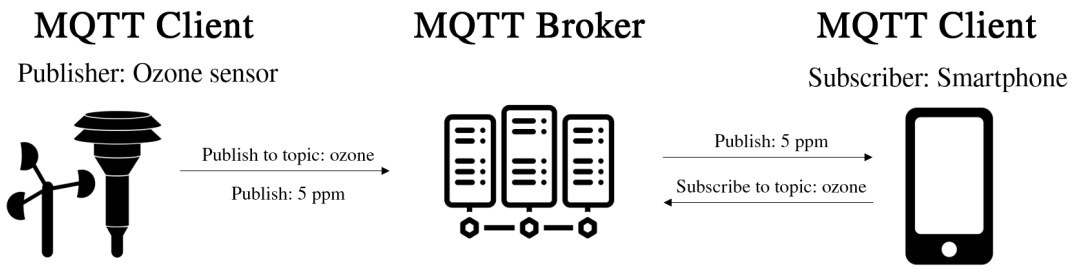


Figure 2.6: MQTT, an publish/subscribing message protocol[44].

2.3 Sensors for ambient air quality

Low-cost air pollution sensors are increasingly attracting more attention by offering air pollution control at a lower cost making possible denser networks at a reasonable cost[46]. However, low-cost sensors typically present worse results than more expensive sensors, but they match applications with lower requirements about data quality[47], represented in Figure 2.7.

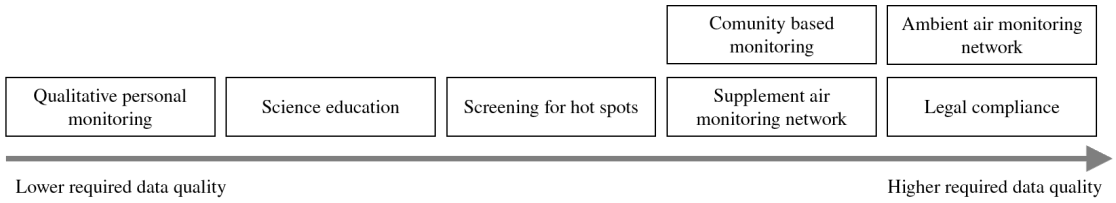


Figure 2.7: Different levels of data quality is required for different applications[47].

A sensor system is constituted by the sensor element consisting of active material absorbing dispersed gases in the air, the transducer that converts sensor response into an output electrical signal, the data logger connected to a communication device, and a power source. From an approach based on the network, the air quality monitoring using low-cost sensors is based on different building blocks with different complexity: sensor element, sensor module, multiple sensor nodes, and a sensor network [37], as represented in Figure 2.8.

Air quality sensors can be classified into two main categories: sensors measuring gas concentrations and sensors for mass concentration of particulate matter. The chemical sensors comprise an appropriate chemically sensitive material interfaced to a transducer. The interaction between material and the pollutant will result in physical changes measured sensed by a transducer and converted into an output signal [37].

Below, operation principles for transducers in air quality monitoring are presented.

[46]European Commission. Measuring air pollution with low-cost sensors, 2018.

[47]US EPA. Draft Roadmap for Next Generation Air Monitoring. Technical report, US EPA, mar 2013. URL: https://www.epa.gov/sites/production/files/2014-09/documents/roadmap-20130308.pdf.

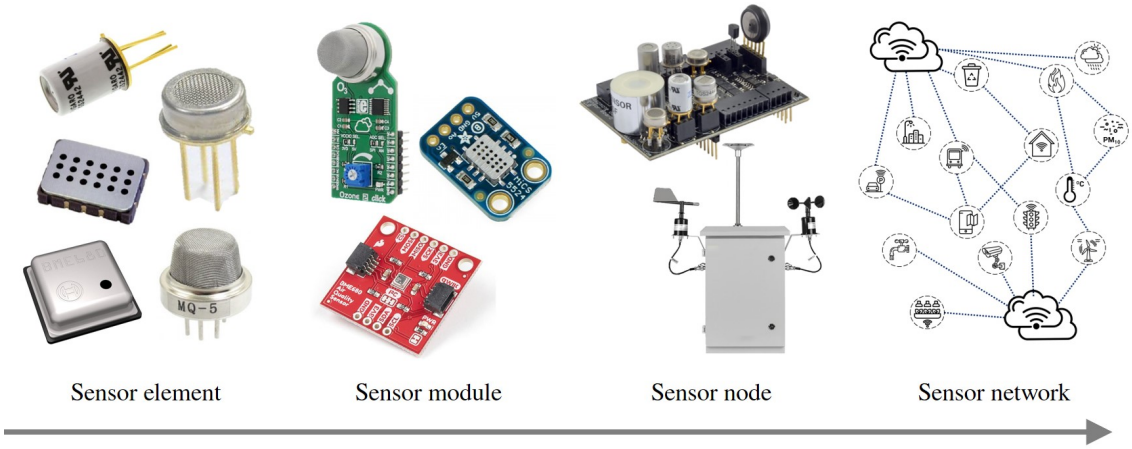


Figure 2.8: Typical sensor network scheme with increasing complexity steps from a sensor element to the wireless sensor network, for air quality monitoring. Based on [37].

Electrochemical sensors (EC) are based on a chemical reaction between gases in the air and the electrode in a liquid inside the sensor. An example could be found in Figure 2.9a.

These sensors are generally used to measure NO_2, O_3 and CO, have a good sensibility, from mg/m^3 to µg/m^3, and long response time (5 - 50 minutes). These sensors, typically, have a signal output current that is linearly proportional to the gas concentration. Results are affected by humidity and temperature and the outputs depend on past inputs' history. Some instability can be observed.

Metal oxide sensors (MOX) are resistive sensors and semiconductors, based on reaction on resistance modification caused by the reaction between air and sensor surface. An example could be found in Figure 2.9b.

Generally, MOX sensors are used to measure NO_2, O_3, SO_2, NO and CO, have a good sensibility, from mg/m^3 to µg/m^3 and a fast response time (30 - 200 seconds). Results are susceptible to temperature and humidity, depending on electrolytes and, there is cross-selectivity with similar molecule types. For these sensors, recalibration is necessary after a specific operation time.

Photoionisation detector (PID) measures electrical current resulting ionisation with UV light of volatile organic compounds. An example could be found in Figure 2.9c.

Generally, these sensors are used to measure VOC's concentration, and the electrical output is proportional to the gas concentration. There is a good sensibility, down to mg/m^3 and a very fast response time (few seconds). Significant signal drift can be observed.

Optical particle counters (OPC) measure the light scattered by particles, giving an indicator of particulate pollution. An example could be found in Figure 2.9d.

These sensors are used to measure PM and can identify their size. There is sensibility in the range of 1 µg/m^3 and a very fast response time (1 second).

Optical sensors detect gases like CO and CO_2 by measuring infrared light absorption.

There is a good sensibility for CO_2 (320 - 2000 ppm), and selectivity is good through characteristic CO_2 IR spectra. Response time varies from 20 and 120 seconds. This sensor requires a correction for the effects of temperature, humidity and pressure[46] [37].



(a) EC sensor. (b) MOX sensor. (c) PID sensor. (d) OPC sensor.

Figure 2.9: Examples of different types of air monitoring sensors.

2.4 Related work and existing systems

The market and existing literature provide some (academic and commercial) solutions to solve similar problems, found in search engines such as ScienceDirect, ResearchGate and IEEE Xplore.

Table 2.10 presents some relevant papers supporting the aim of this dissertation: to develop an ambient air quality monitoring system. Each project includes information on technologies used and other relevant details.

Table 2.10: Related works and existing systems.

Citation	Topics	Study focus
Development of a Microcontroller Based System for Measuring Environmental Conditions [48]	IAQ, Monitoring Solution, Wi-Fi, ESP8266, BME680, ThingSpeak	Study for the integration of an air quality sensor in a fire detectors network.
Artificial intelligence-enabled context-aware air quality prediction for smart cities [38]	AQ, Smart cities, AQ prediction, Artificial intelligence	Construction of a model for accurate air quality prediction algorithm, based on Long Short-Term Memory Deep Neural Network, with information of surrounding pollution sources (e.g., bush fires, traffic volumes) and users' health profile.
AirCloud: A Cloud-based Air-Quality Monitoring System for Everyone [39]	AQ, Monitoring solution, cloud, BLE, 3G, PPD42NJ, $PM_{2.5}$	Development of a system measuring $PM_{2.5}$ sending it data for the online database.

Continued on next page

Table 2.10 – continued from previous page

Citation	Topics	Study focus
Low-cost air pollution monitoring systems: A review of protocols and enabling technologies [40]	AQ, Low-cost sensors, $PM_{2.5}$, IoT, WSN	State of the Art of low-cost air pollution monitoring technologies.
Compliance of indoor air quality during sleep with legislation and guidelines. A case study of Lisbon dwellings [41]	IAQ, Monitoring solution	Using professional IAQ meters and governed by Portuguese AQ legislation, the project's members monitored IAQ in Lisbon dwellings.
Air quality monitoring using mobile low-cost sensors mounted on trash-trucks: Methods development and lessons learned [42]	AQ, Monitoring solution, Mobile sensing	The team of the project mounted two OPC on two trash-trucks with the purpose of identifying aerosol hotspots, mapping the pollution across the city (Cambridge) and understanding aerosol sources.
Evaluation of low-cost sensors for quantitative personal exposure monitoring [43]	AQ, PM, Low-cost sensors	This study uses various methods to compare low-cost sensor with industry-grade instruments. The results indicate a strong relation (R^2=0.78) between low-cost and industrial sensors employing a calibration model based on Support vector regression, a machine learning method.
Battery-free smart-sensor system for real-time indoor air quality monitoring [44]	AQ, Low-energy	Design of a PCB collecting available RF (radio frequency) waves to power the sensing operation. That energy is temporarily stored in a supercapacitor.
A Modular IoT Platform for Real-Time Indoor Air Quality Monitoring [45]	IAQ, Monitoring solution, Zigbee	Development of a system to control IAQ. The nodes send the sensor data via Zigbee to gateways, connected to a PC, via Wi-fi.
Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation [46]	LoRa, LoRaWAN, Data collection, Sensors	Creation and implementation of a LoRa custom mesh network.

Continued on next page

Table 2.10 – continued from previous page

Citation	Topics	Study focus
LoRaWAN Mesh Networks: A Review and Classification of Multihop Communication [47]	LoRa, LoRaWAN, Mesh networks	This article is the result of an extensive literature review about LoRa networks' topologies, including LoRaWAN. The authors considered four types of nodes: main-gateway (connection with the server), router-gateway (routes data from other routers to the main-gateway), relay-gateway (receive data from sensor nodes, relay-devices or end-devices, and send that for router-gateways), relay-device (it is a sensor node that sends its data and, also, redirects end-device data for router-gateway and end-device (sensor nodes sending data for relay-devices or relay-gateways). An architecture example is present in Figure 2.10.
Environmental Monitoring System Based on LoRa Technology in Island [33]	Monitoring system, LoRa	Simple LoRa communication between a remote place (an island), 23 km apart of the receptor.
A New Smart Sensing System Using LoRaWAN for Environmental Monitoring [34]	AQ, Monitoring System, LoRaWAN, Cloud	Solution for AQ monitoring using LoRaWAN, recurring to sleep modes for less energy consumption. There are graphics displaying the collected data.

2.5 Summary

This chapter presents a review of the relevant literature to pursue the proposals of this Master's dissertation.

Section 2.1 presents a review of the legislation and recommendations from international and national entities, such as Governments or the World's Health Organization. The different laws and recommendations determine a maximum acceptable value for the pollutants. Other institutions, normally dedicated to air quality monitoring, determine a level-based scale to communicate easily air quality information with the public.

Section 2.2 presents an overview of smart cities and the technologies used that could be applied to this project. Wireless communications assume a very relevant role in the context of smart cities due to their great flexibility and reliability. Technologies such as LoRa, ideal for low-power and long-range applications, show exponential growth and are expected to become prevalent in the market in a few years.

Section 2.3 includes an analysis of existing sensor solutions for low-cost environmental and pollution monitoring. Briefly, there are sensors based on chemical reactions with gases, resulting in a voltage or current variance and, there are sensors that measure light scattering by particles.

[48] Oleksandra Sidlovska. *Development of a Microcontroller Based System for Measuring Environmental Conditions*. Master's dissertation, University of Porto - Faculty of Engineering, Porto, feb 2018. URL: https://hdl.handle.net/10216/113512.

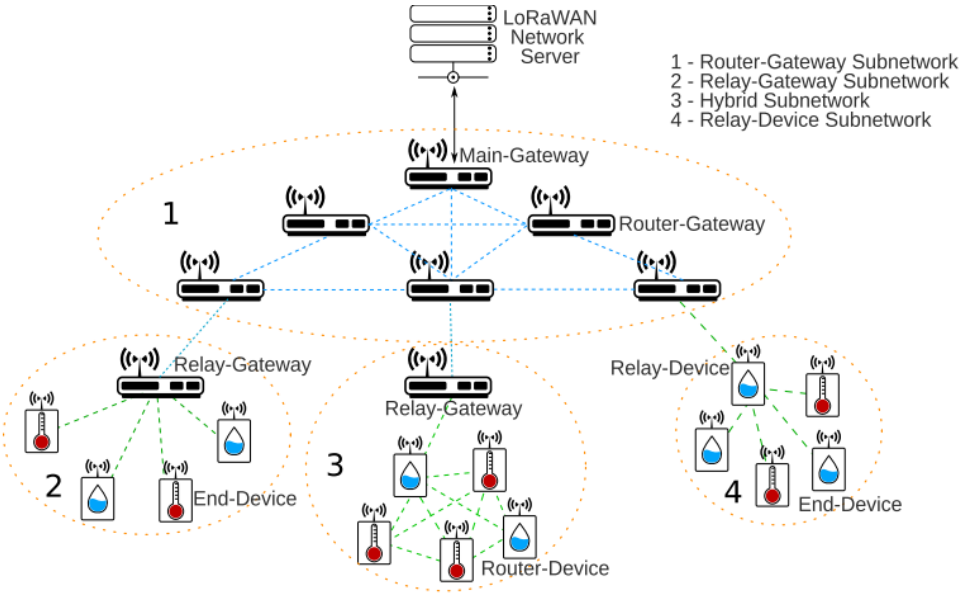


Figure 2.10: Example of a LoRa and LoRaWAN combination. Extracted from [47].

In the last section, Section 2.4, a brief listing of selected relevant articles to the development of this project is presented.

Chapter 3

Architecture of the project

The previous chapters present the problem of this dissertation and the existing solutions. This chapter proposes a solution for the air quality monitoring system based on an analysis of the requirements and the known technologies.

3.1 Requirements of the project

As the title of this dissertation suggests, the main objective of this project is to create a system to monitor air quality (for indoor and outdoor), designed for a city environment.

The context of the installation requires the collecting points to be located in not closely areas, measuring air quality in critical points as typical traffic or people concentration points. Therefore, **various data collection points** will be needed around the city, **sending low quantities of data** through the **network**. It will be necessary to **attach sensors**, capable of measuring relevant air quality parameters and environmental factors, for each point. And, **present the collected data** to ease the monitoring work of decision-makers and support it.

3.2 Answer to the requirements

3.2.1 The system

The requirements set different parts for the project:

- **Collecting Points/End Nodes** – points that will collect environment data;
  - **Microcontroller** – a board where the sensors could be coupled and send data to a central system;
  - **Sensors** – to collect data from the environment and convert it in a "digital" value. Different sensors should be used for indoor and outdoor air quality;
- **Gateway** – to receive data from all the collecting points and send it to the server;
- **Network server** – hosts collected data and send it to applications;
- **Applications** – application to monitor data (e.g.: web application, notifications).

3.2.2 Wireless technology

To connect the nodes to the gateway, which are physically distant from each other, it will be necessary to use a long-range wireless protocol. In a city context, a reasonable value for the distance between collecting points could be between one and five kilometres.

Section 2.2.1 presented the leading wireless technologies and standards for wireless communication. For these values in the order of kilometres and outdoor environments, the available solutions are LPWAN, a low-power wide-area network, such as LoRa, or a cellular technology, such as GSM/GPRS, 3G or 4G. Although all those technologies can be applied to the system, some priorities were established in the project. These technologies are comparable in regard to implementation costs, operation costs charged by service providers, availability, state of development of the standards, and relevance in the market.

This project's priority was low prices, lower energy consumption, independence from external and non-controllable services, and growth. For these main reasons, LoRa is the analysed technology that fits better. Moreover, LoRa is projected to be the leading non-cellular LPWAN technology, representing more than 1/4 of this market[1] [48], making it a more appealing technology to explore.

As is written in Section 2.2.1.5, a LoRa-based system could be mounted with LoRaWAN protocol or not. In other words, it is possible to choose between the LoRa physical layer with LoRaWAN protocol or the LoRa physical layer with a custom way of communication. Although both meet the system's requirements, the second option is chosen for economic reasons (there is no need to buy a LoRaWAN gateway, that costs $\approx$ 200 EUR).

This option presents some disadvantages, such as not having in-built data encryption and no message addressing (i.e., a LoRa device broadcast a message that could be intercepted by any other LoRa device).

3.2.3 The sensors

As seen in Sections 2.1.2.1 and 2.1.3.2 the air quality parameters used to monitor indoor and outdoor air quality are different. Thus, it is expected the sensors used to evaluate indoor and outdoor air are different. Besides using sensors to monitor air pollutants, evaluating environmental parameters – air temperature and relative humidity – is also important to have a complete air quality perception. Market research reveals sensors for the majority of the target parameters. Table 3.1 lists the sensors chosen to measure the parameters referred above, found in the leading electronic suppliers (DigiKey and Mouser).

3.2.4 The microcontroller

The market offers a wide range of solutions for microcontrollers and single-board computers. The microcontroller for this application has minimum requirements. It should support some features, such as having I^2C and SPI interface, supporting wireless communication, particularly LoRa and Wi-Fi, setting sleeping modes, low power consumption, and being inexpensive.

Various solutions were considered:

1. Waspmote[2], from the Spanish company Libelium, Figure 3.1a, presented as the lowest consumption IoT platform in the market, specially designed for smart city applications. This option was discarded because of its high cost ($\approx$ 150 EUR) and lack of support from the company;

[1] Adarsh Krishnan. LoRaWAN and Multi-RAN Architecture Connecting the Next Billion IoT Devices. Technical report, ABI Research, Oyster Bay, New York, United States of America, 2020. URL: https://info.semtech.com/abi-research-white-paper.

[2] Libelium Comunicaciones Distribuidas SL. Waspmote » The sensor platform to develop IoT projects | Libelium. URL: https://www.libelium.com/iot-products/waspmote/.

Table 3.1: List of sensors used in the project.

Sensor module	On-board sensor module	Measured parameters	Input voltage	Interface	Price	Consumption
MikroElektronika MIKROE-2767 (Ozone 2 click)	Hanwei Electronics MQ131	Ozone (10 - 1000 ppm)	3.3 and 5 V	Analog SPI	≈ 30 EUR	
MikroElektronika MIKROE-3098 (NO2 click)	SPEC Sensors 3SP-NO2-20	NO_2 (0 - 20 ppm)	3.3 or 5 V	Analog IC	≈ 45 EUR	
MikroElektronika MIKROE-3056 (Air quality 5 click)	SGX Sensortech MiCS-6814	CO (1 – 1000ppm), NO_2 (0 – 10ppm), NH_3 (1 – 500ppm)	3.3 and 5 V	I^2C	≈ 40 EUR	
Seeed Studio/ Shinyei PPD42NS	N/A	PM (0 - 28000 pcs/L) (just measure concentration, not the weight)	4.75 - 5.25 V	UART	≈ 30 EUR	90 mA (standby mode)
Sensirion SPS30	N/A	$PM_{1.0}$, $PM_{2.5}$, $PM_{4.0}$, PM_{10} (0 - 1000 µg/m^3)	4.5 - 5.5 V	UART I^2C	≈ 35 EUR	55 mA (avarage current)
DFRobot Gravity BME680	Bosch Sensortec BME680	T (-40 - 85 ºC), RH (0 - 100%), P (30 kPa - 110 kPa)	3.3 - 5 V	SPI I^2C	≈ 15 EUR	5mA
Sparkfun SEN-14348 Environmental Combo	Bosch Sensortec BME280 + ams CCS811	T (-40 - 85 ºC), RH (0 - 100%) P (30 kPa - 110 kPa), eCO2 (400 - 8192 ppm), TVOC (0 - 1187 ppm)	3.3 V	I^2C	≈ 30 EUR	14 mA (max. current)

2. Raspberry Pi 3[3], produced by Raspberry Pi Foundation (United Kingdom), Figure 3.1b, a single-board computer with Wi-Fi and Bluetooth connectivity that delivers high processing power. This option was discarded because it does not comply with the low power requirements imposed;

3. Arduino UNO[4], created in Interaction Design Institute Ivrea (Italy), Figure 3.1c, is an open-source MCU based on ATmega328P (from the American company Microchip Technology) equipped with wireless communication modules and the sensors to monitor air quality. This option is acceptable and meets all requirements but discarded because a better option was found;

4. An ESP32-based MCU (ESP32 is an MCU series integrating Wi-Fi and Bluetooth, developed by Chinese company Espressif Systems), including an on-board LoRa module. Heltec LoRa 32 (from Chinese company Heltec Automation)[5], Figure 3.1d, and TTGO T-Beam ESP32, Figure 3.1e, and TTGO LORA32, Figure. 3.1f (from Chinese company LILYGO)[6] are two examples of these modules .

For this project, two ESP32-based MCU models were used, TTGO LORA32 V2.1.6 and TTGO T-Beam V1.1, both from LILYGO, including LoRa transceiver chip SX1276 (from Semtech Corporation). The most important specifications of these PCBs are expressed in Table 3.2.

3.2.5 Data management and presentation

The data collected by the sensors installed on the system should be storage in a server to be easily accessed. Also, the data should be presented in a simple dashboard. A solution for storage and presentation could be an online service operating MQTT protocol, i.e., an MQTT broker.

[3]Raspberry Pi Foundation. Buy a Raspberry Pi 3 Model B – Raspberry Pi. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[4]Arduino. Arduino Uno Rev3 | Arduino Official Store. URL: https://store.arduino.cc/arduino-uno-rev3.

[5]Heltec Automation. WIFI LoRa 32 (V2) – Heltec Automation. URL: https://heltec.org/project/wifi-lora-32/.

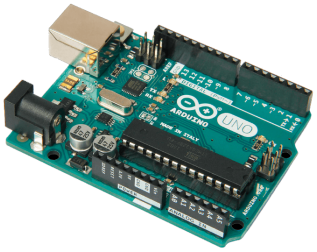
[6]LILYGO. TTGO Lora Series - Shenzhen Xin Yuan Electronic Technology Co., Ltd. URL: http://www.lilygo.cn/products.aspx?TypeId=50003&fid=t3:50003:3.



(a) Libelium Waspmote[2].



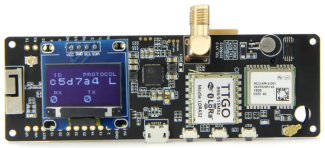
(b) Raspberry Pi 3[3].



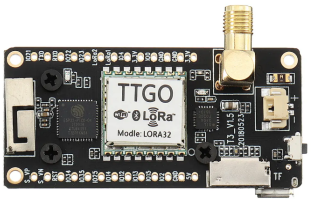
(c) Arduino Uno R3[4].



(d) Heltec LoRa 32[5].



(e) LILYGO TTGO T-Beam[6].



(f) LILYGO TTGO LORA32[6].

Figure 3.1: Examples of MCUs with LoRa capacities.

Table 3.2: Specifications for TTGO LORA32 V2.1.6 and TTGO T-Beam V1.1, both produced by LILYGO[6].

	TTGO LORA32 V2.1.6	**TTGO T-Beam V1.1**
LoRa transceivers	Chip: SX1276 (from Semtech Corp.) Operating frequency: 868 MHz Fast wake-up and frequency hopping	Chip: SX1276 (from Semtech Corp.) Operating frequency: 868 MHz Fast wake-up and frequency hopping
ESP32	ESP32 PICO-D4 Bluetooth 4.2 + Low Energy 4 MB Flash	ESP32 D0WDQ6 Bluetooth 4.2 + Low Energy 4MB Flash
Interfaces	SPI, I^2C, ADC, DAC, GPIO, UART	SPI, I^2C, ADC, DAC, GPIO, UART
Sleep modes	Includes sleep and idle modes	Includes sleep and idle modes
Size	100 mm × 33 mm	64.5 mm × 27 mm
Extra features	Built-in OLED Incorporate a SD card slot	GPS modules NEO-6M Includes a battery slot
Price	≈ 20 EUR	≈ 25 EUR
Supplier	Banggood (based in China, warehouse in UK)	Banggood (based in China, warehouse in UK)

3.3 Proposed solution

The proposed solution includes three nodes, one prepared to measure indoor air quality and two for outdoor air quality, communicating among them and sending pollution data to a network for analysis, as schematised in Figure 3.2.

- **Node 1** (outdoor) – a TTGO LORA32 unit coupled with MIKROE-2767 (ozone), MIKROE-3098 (NO_2) and Sparkfun SEN-14348 (temperature, humidity, pressure, CO_2 and Total VOCs) and powered using 5V DC. This unit sends, via LoRa, sensor data to Node 3 (that routes this data to database via MQTT).
- **Node 2** (outdoor) – a TTGO T-Beam unit coupled with DFRobot Gravity BME680 (temperature, humidity and pressure) and Sensirion SPS30 (particulate matter). This node is powered using a rechargeable battery charged with solar energy. Like node 1, node 2 sends, via LoRa, sensor data to

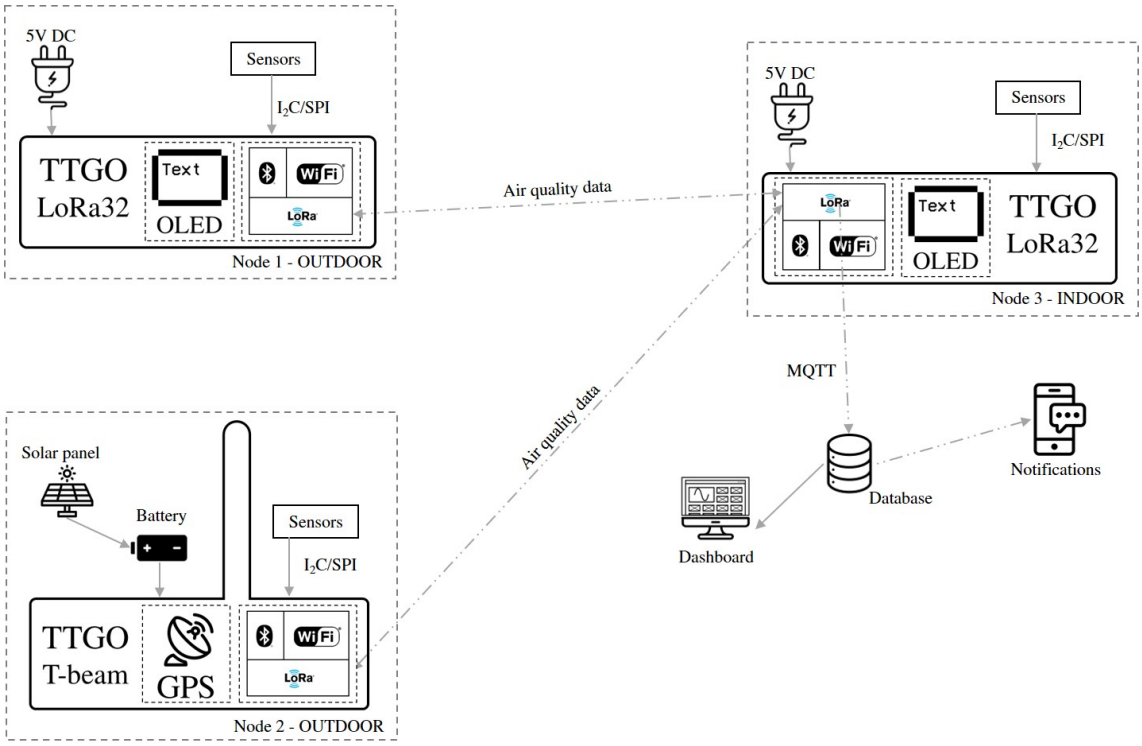


Figure 3.2: Architecture of the proposed solution.

Node 3 (that routes this data to the database via MQTT).

- **Node 3** (indoor and router) – this TTGO LORA32 unit is coupled with MIKROE-3058 (NO_2, NH_3 and CO), Sparkfun SEN-14348 and Shinyei PPD42NS (particulate matter) and powered with 5V DC. This node reads sensor data and sends them to the Internet, as well that route received data from other nodes to the database via MQTT. It connects to the Internet via Wi-Fi.

Moreover, it is proposed to construct a dashboard web page to visualise collected sensor data and create an application to send notifications to the system admin when some pollutant exceed their thresholds.

3.4 Summary

This chapter presents the answer to the project's requirements and priorities and a proposed solution based on that.

First of all, LoRa was selected as the wireless technology for communications between nodes mainly because of low-power consumption, communication capacity for long distances and increasingly usage projections. Some low-cost sensors are chosen based on measuring parameters, availability on the market and price. The ESP32-based microcontroller with LoRa was chosen mainly for its excellent price/quality ratio for the application in question.

The proposed solution was to use three different nodes (two for outdoor, one for indoor) reading environmental factors and pollution levels. The outdoor nodes route their data to the indoor node. This indoor node is connected to the Internet and reroutes all data for the server, using MQTT protocol.

Chapter 4

Hardware design

In this Chapter, it is presented the hardware part of the project. It specifies the mounting process and the analysis of sensor data. Moreover, PCB and PV power supply were designed to create resistance to environmental conditions and autonomy of the system.

4.1 Sensor modules analysis

This section presents an analysis of the components of the sensor modules and interprets the sensor output data. This study is crucial for obtaining correct readings of the sensors and translating current values into concentration units.

It is appropriate to point out that some sensor modules' manufacturers and the online community already constructed the necessary libraries, ready to be used in projects.

4.1.1 MIKROE-2767 – Ozone 2 click

The sensor module MIKROE-2767[1], Figure 4.1, is a PCB produced by the Serbian company MikroElektronika, marketed as "Ozone 2 click", based on the MQ131 sensor. The target gas of the sensor is ozone, but it has some cross-sensitivity for other oxidation gases, such as NO_x and Cl_2, i.e., these gases could affect the calculation of ozone concentration. Moreover, this sensor requires a pre-heat time of 48 hours to present stable results.

This MOX sensor has a sensitive layer made of SnO_2, which changes its resistance when exposed to ozone (Figures 4.4a and 4.4b on page 36). The analogue output of the sensor is sampled by a 22-bit A/D converter (ADC) offering the conversion result on the SPI interface.

4.1.1.1 SPI communication

A/D converter MCP3551 from Microchip Technology Inc. has an SPI interface[2]. This communication is possible with 3 wire:

[1] MikroElektronika. Ozone 2 click - board with an MQ131 sensor for O3 | MikroElektronika. URL: https://www.mikroe.com/ozone-2-click.

[2] Microchip Technology Inc. MCP3550/1/3 Data-sheet. Low-Power, Single-Channel 22-Bit DeltaSigma ADCs, 2014.

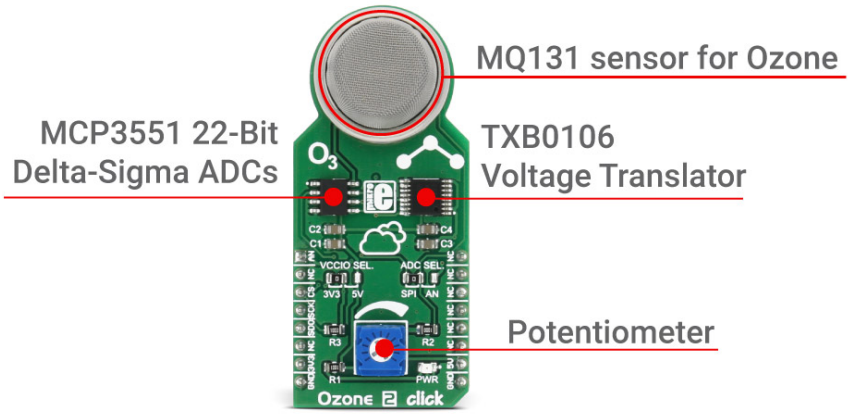


Figure 4.1: MIKROE-2767 marked with key components[1].

- SCK (Serial clock) synchronises data communication with the device;
- SDO (Data Output) or MISO (Master in, Slave out) is the output data pin of the device. Once a conversion is complete, this pin will go active-low, acting as a ready flag. Subsequent falling clock edges will place the 24-bit data word (two overflow bits and 22 bits of data (Bit 21 for signal, Bit 20 is the Most Significant Bit and bit 0 is the Least Significant bit) on the SPI bus through the SDO pin. Data is clocked out on the falling edge of SCK;
- $\overline{CS}$ (Chip select) gates all communication to the device and can be used to select various devices sharing the same SCK and SDO pins. This pin is normally high and when it falls, the conversion starts.

The typical serial device communication is presented in Figure 4.2.

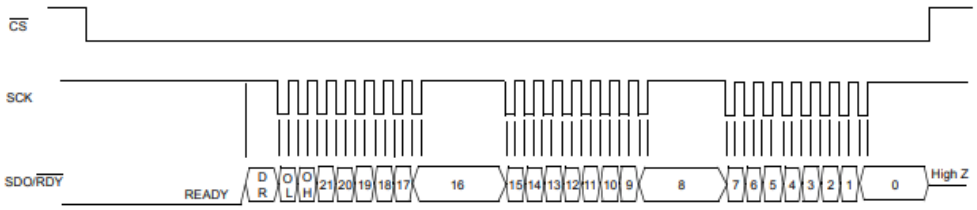


Figure 4.2: SPI communication of ADC, part of MIKROE-2767[2].

4.1.1.2 Concentration calculation

The value MCU receives via SPI is a binary value converted by A/D converter, *ADCvalue*. This ADC has 22 bits but according to its data-sheet[2], one of this bits is reserved for the signal (positive or negative). For this reason, the effective resolution for MIKROE-2767 is 21 bits. Also, according to the schematic $V_{CC} = 5.0$ V. The value of R_L is a value manually imposed through a physical potentiometer of 10 kΩ + 100 Ω[3]. The

[3]MikroElektronika. Ozone 2 click schematic v100, 2017. URL: https://download.mikroe.com/documents/add-on-boards/click/ozone-2/ozone-2-click-schematic.pdf.

value chosen for R_L was 100 Ω. Therefore, the voltage output of the transducer, V_{RL}, is:

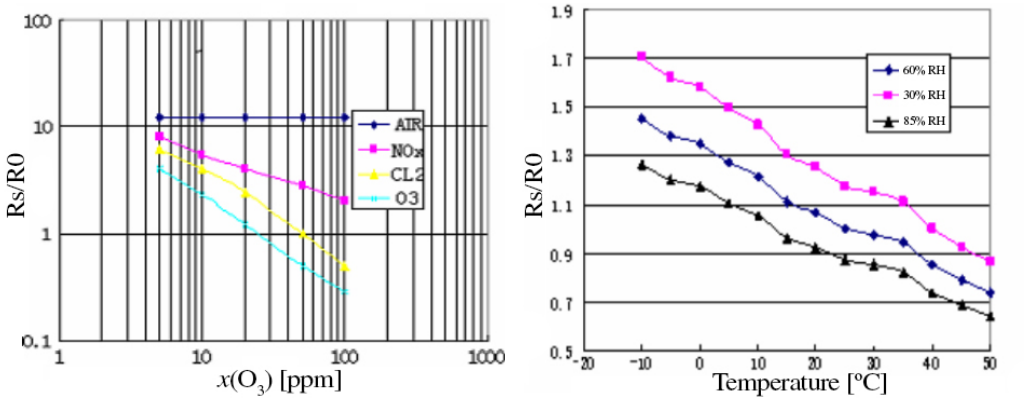
$$V_{RL} = \frac{ADCvalue}{2^{21}-1} \times V_{CC}, \text{ where } V_{CC} = 5.0 \text{ V}. \tag{4.1}$$

Figure 4.4c conduces to the value R_S, through Equation 4.2.

$$R_S = \left(\frac{5.0}{V_{RL}} - 1\right) R_L, \text{ where } R_L = 100\ \Omega \tag{4.2}$$

The sensitivity curve of MQ131 provided by Hanwei Electronics[4], Figure 4.3a, gives the relation between R_S/R_0 and the concentration of O_3 in ppm. Microsoft Excel was used to get the equation of the curve, where some points of the manufacturer's curve were traced (by overlaying the original graph and the new one, produced with the points). After that, an exponential regression was calculated. It is important to note that R_0, introduced in software, represents a value dependent on the calibration and means the sensor resistance for 50 ppm of Cl_2. Next, a correction factor CF is applied due to the influence of temperature and humidity on the transducer. The Equation 4.3 represents the result of the regression.

$$x(O_3) = 23.534 \times \left(\frac{R_S}{R_0}\right)^{-1.091} \times CF \text{ [ppm]} \tag{4.3}$$



(a) Sensitivity curve of MQ131 for ozone, Cl_2 and NO_x[4].

(b) Influence of temperature and humidity in output value of transducer[4].

Figure 4.3: MQ131 characteristic curves.

The correction factor CF due to the temperature and relative humidity is found through linear regression of curves in Figure 4.3b.

For humidity $> 75\%$, it will be used the 85% curve: $CF = -0.0141 \times T$ [ºC] $+ 1.5623$.

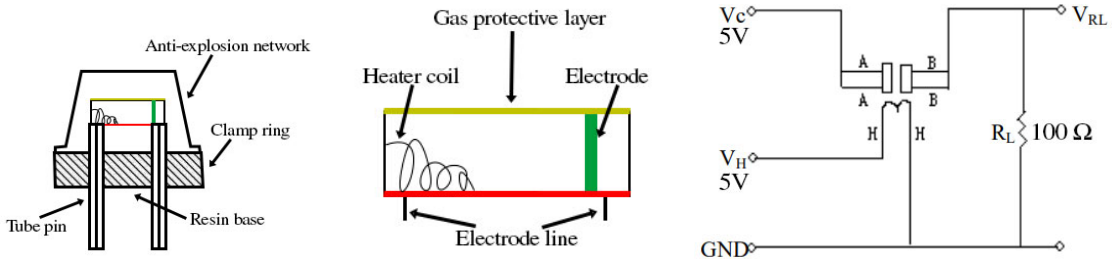
For humidity $> 50\%$, it will be used the 60% curve: $CF = -0.0119 \times T$ [ºC] $+ 1.3261$.

For humidity $< 50\%$, it will be used the 30% curve: $CF = -0.0103 \times T$ [ºC] $+ 1.1507$.

4.1.1.3 Code for this sensor

Bellow, an excerpt of the C++ code, uploaded in the MCU, created to read the ADC value and translate these equations is presented.

[4]Henan Hanwei Electronics Co Ltd. MQ131 Data-sheet. MQ131 Semiconductor Sensor for Ozone. URL: www.hwsensor.com.



(a) MQ131 sectional view[4]. (b) MQ131 sensor[4]. (c) MQ131 schematic[4].

Figure 4.4: MQ131 sensor details.

```
// set CS pin low to start conversion
digitalWrite (pinCS, LOW);

// wait for conversion complete from SDO pin (it goes low)
while (digitalRead(pinSDO));

uint8_t ADC_byte[3];
ADC_byte[0] = hspi.transfer(0x00);
ADC_byte[1] = hspi.transfer(0x00);
ADC_byte[2] = hspi.transfer(0x00);

// set CS pin high to turn off SPI data communication
digitalWrite (pinCS, HIGH);

uint32_t ADC_data = 0;
ADC_data = ADC_byte[0];
ADC_data <<= 8;
ADC_data |= ADC_byte[1];
ADC_data <<= 8;
ADC_data |= ADC_byte[2];

float RL = 100;
float R0 = 235; // arbitrary value
float vRL = ((float)ADC_data) / (21^2-1) * 5.0;
float rS = (5.0/vRL - 1.0) * RL;
float ratio = rS / R0 * getEnvCorrectRatio(hum, temp);
O3_ppm = 23.534 * pow(ratio,-1.091);
```

4.1.1.4 Calibration process

The sensors used in this project require calibration to provide reliable values. Some of them are calibrated by the manufacturer (such as MIKROE-3098, for NO_2). Other sensors need to be calibrated by the user

(such as MIKROE-2767, for O_3). Plus, other sensors do not mention calibration, but the results need to be validated.

It is necessary to compare the measurements given by the sensors to accurate values of the parameters. These reference values are got by tests in controlled atmosphere or by precise and accurate instrumentation, generally much more expensive than the type of sensors used in this project. As this equipment is not available at the time of this work, only the O_3 sensor (MIKROE-2767) will be calibrated using data collected by a governmental air quality monitoring station. The sensitivity curve of O_3 uses a coefficient to match the value of ozone concentration. This coefficient compares the value of the internal resistance of the sensor, R_S, with the value of that resistance in a controlled environment, R_0, as seen in Section 4.1.1.2. For that, an arbitrary constant was chosen for R_0 ($= 235\ \Omega$) and the hourly results of MIKROE-2767 were compared with the public hourly results of 4 hours, from the station of Sobreiras-Lordelo do Ouro (in Porto) *Agência Portuguesa do Ambiente*[5]. Table 4.1 presents the values for both measurements and the correspondent value of R_S/R_0, given by the Equation 4.2.

Table 4.1: Comparison of O_3 concentrations measured by the sensor and by public station for air quality monitoring (n=4).

Time	**Hourly results for sensor (ppb)**	$\frac{R_S}{R_0}$ **for sensor** (A)	**Hourly results for station (ppb)**	$\frac{R_S}{R_0}$ **for station** (B)	B/A
15:00	1140.68	16.03	22.92	575.79	35.92
16:00	969.66	18.60	22.41	587.77	31.60
17:00	1182.11	15.51	26.49	504.32	32.51
18:00	1084.04	16.79	24.96	532.55	31.71
Mean					32.94
Standard deviation					2.03 $= 6.17\%$

The Equation 4.4 calculates the value that R_0 should assume to present similar ozone concentrations to the station. That value were introduced in software to more accurate measurements.

$$\frac{R_{S,sensor}}{R_{0,sensor}} \times 32.94 = \frac{R_{S,station}}{R_{0,station}} \iff R_{0,station} = \frac{235}{32.94} = 7.134\ \Omega \ \because\ R_{S,sensor} = R_{S,station} \tag{4.4}$$

Alternatively, the value of R_S/R_0 could be manually adjusted by using the physical potentitometer.

This calibration methodology allows the system to present more accurate values. However, this process is not precise since it was carried out with a reduced set of samples (n=4). The test should be made for more samples. In this case, if the standard deviation were high, it meant that the precision of the sensor would be low. If the standard deviation were low, it meant that this low-cost sensor would be a good replacement to the more expensive sensors in mass projects (such as a Smart City).

4.1.2 MIKROE-3098 – NO2 click

NO2 click[6], Figure 4.5, is a sensor module equipped with "3SP-NO2 110-508" sensor produced by SPEC Sensors for detecting NO_2. Its sensor includes an onboard filter for ozone and chlorine that allows this chemical sensor to exhibit a low cross-sensitivity to other gases. Also, this sensor has a fast response time (< 15 seconds) and an expected lifespan of 10 years.

[5] Agência Portuguesa do Ambiente. QualAR - Qualidade do AR. URL: https://qualar.apambiente.pt/indices.

[6] MikroElektronika. NO2 click | Mikroelektronika. URL: https://www.mikroe.com/no2-click.

The sensor included in MIKROE-3098 is an amperometric gas sensor. In other words, it is an electrochemical sensor whose sensing electrode reacts with NO_2 generating a current linearly proportional to the gas concentration, as seen in Figure 4.6a. This current is converted and transformed into the voltage by the analogue front-end (AFE) potentiostat IC, LMP91000SD from Texas Instruments[7], and, after that, the resultant voltage is converted with the 12-bit A/D converter with I^{2}C interface, MCP1501T-20E/CHY from Microchip[8].

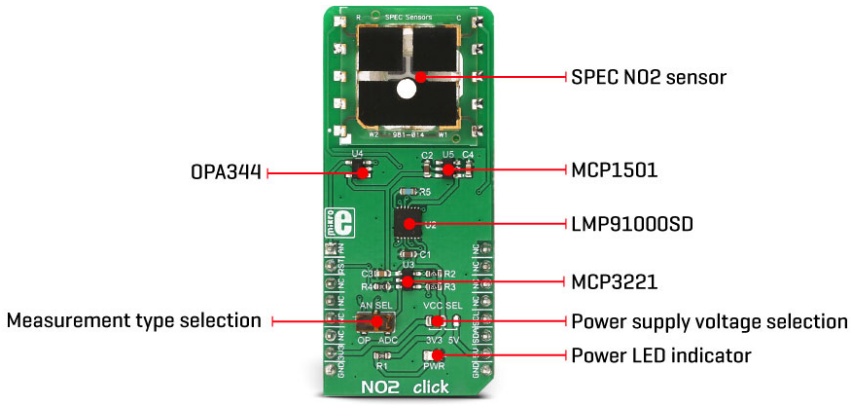


Figure 4.5: MIKROE-3098 marked with key components[6].

4.1.2.1 I^{2}C communication and initial configuration

The communication between MIKROE-3098 and MCU is achieved via I^{2}C, using two wires, SCL (Serial Clock) for clock synchronisation and SDA (Serial data) to receive and transmit data. The address of this sensor is `0x4D` (77 in decimal).

Before collecting data from NO_2 sensor, it is necessary to set work parameters of the analog front-end potentiostat IC. This configuration is set via I^{2}C registers. These parameters are set.

- Mode Control Register – Mode of Operation selected: Deep Sleep;
- Lock/Protection Register – Write protection: Write mode;
- TIA Control Register – TIA (transimpedance amplifier) gain = 2.75 kΩ and R_{LOAD} = 100 Ω. These two values are arbitrary and reflected in calculations below;
- Reference Control Register – Reference voltage source = internal, internal zero selection (% of source reference) = 50%, bias polarity = negative (VRE>VWE) and bias selection (% of source reference) = 0%.

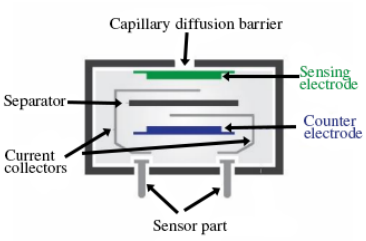
Bellow, part of the C++ code that runs in *setup()* function reflecting these configurations is presented:

```
1 const uint8_t _NO2_TIACN_REG      = 0x10;
2
3 const uint8_t _NO2_2750_OHM_TIA_RES = 0x04;
4 const uint8_t _NO2_100_OHM_LOAD_RES = 0x03;
```

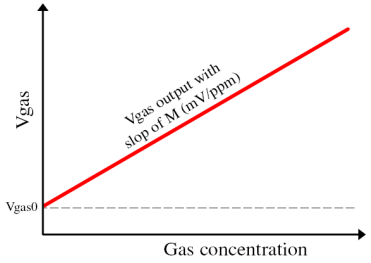
[7]Texas Instruments. LMP91000 Data-sheet. LMP91000 Sensor AFE System: Configurable AFE Potentiostat for Low-Power Chemical | Sensing Applications, jan 2011. URL: www.ti.com.

[8]Microchip Technology Inc. MCP3221 Data-sheet. Low-Power 12-Bit A/D Converter with I2C Interface, jan 2017. URL: https://download.mikroe.com/documents/datasheets/mcp3221.pdf.

```
5
6  Wire.begin(); // begin I2C communication
7  Wire.beginTransmission(77); // transmit to device at I2C address 0x4D = 77
8
9  uint8_t writeReg3[2];
10 writeReg3[0] = _NO2_TIACN_REG;
11 writeReg3[1] = _NO2_2750_OHM_TIA_RES | _NO2_100_OHM_LOAD_RES;
12
13 Wire.write(writeReg3, 2); // write the array writeReg3 with lenght of 2 bytes
14
15 Wire.endTransmission();
```



(a) SPEC sensor electrode[a].



(b) Relation between voltage and gas concentration.



(c) Sensor label.

[a]SPEC Sensors. SPEC Sensor Operation Overview. Technical report, may 2016.

Figure 4.6: Sensor SPEC 3SP-NO2 110-508 details.

4.1.2.2 Concentration calculation

After requesting 2 bytes from the device at the address 0x4D, the A/D converter sends a 12-bit conversion of the value, *ADCvalue*. The voltage reference used for this ADC is 2.0480 V given by a high-precision voltage reference, MCP1501T-20E/CHY[9]. The Equation 4.5 gives the input voltage of ADC (output voltage of the AFE IC), V_{gas}.

$$V_{gas} = \frac{ADCvalue}{2^{12}-1} \times V_{ref}, \text{ where } V_{ref} = 2.0480 \text{ V}^{9} \tag{4.5}$$

The relation between NO_2 concentration and the output current of the sensor is linear. Also, the function between current and voltage in the AFE IC is linear. With these, it is possible to trace the relation between V_{gas} and gas concentration, represented in Figure 4.6b.

The target gas, nitrogen dioxide, concentration is given by the Equation 4.6, where $V_{gas,0}$ is the voltage given by the transducer for clean-air environment (free of NO_2) and M is the the slope of the curve in the

[9]Microchip Technology Inc. MCP1501 Data-sheet. High Precision Buffered Voltage Reference, 2017. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/20005474E.pdf.

Figure 4.6b, representing the total sensitivity, given by Equantion 4.7.

$$x(NO_2) = \frac{1}{M} \times (V_{gas} - V_{gas,0}) \tag{4.6}$$

The sensitivity is given by the multiplication of sensitivities of different components. The sensor sensitivity is 26.90 nA/ppm given by a *Sensitivity Code* written in a print label present in the sensor, Figure 4.6c. *TIA Gain* is the gain of the AFE IC and was set as 2.75 kΩ in the Section 4.1.2.1.

$$M\left[\frac{\text{V}}{\text{ppm}}\right] = \textit{Sensitivity Code}\left[\frac{\text{nA}}{\text{ppm}}\right] \times \textit{TIA Gain}\left[\frac{\text{kV}}{\text{A}}\right] \times 10^{-9}\left[\frac{\text{A}}{\text{nA}}\right] \times 10^{3}\left[\frac{\text{V}}{\text{kV}}\right] \tag{4.7}$$

4.1.2.3 Code for this sensor

Bellow, an excerpt of the C++ code, uploaded in the MCU, created to read the ADC value and translate these equations is presented.

```
1  Wire.requestFrom(77, 2, 1); // request two bytes from device 0x4D = 77
2
3  if (Wire.available()==2) { // slave may send less than requested
4      uint8_t readData = Wire.read(); // receive the first byte (8 bits)
5      uint16_t ADC_data = readData;
6      ADC_data <<= 8;// bitwise left shift assignment
7      readData = Wire.read(); // receive the 2nd byte (8 bits)
8      ADC_data |= readData; // concatenate the 2 bytes (= 16 bits)
9
10     float NO2_Vref = 2.0480;
11     float Vgas0 = 0; // measured value
12     float NO2_voltage = (ADC_data/4095.0) * NO2_Vref;
13     float NO2_M = 26.90 * 2.75 * 10e-9 * 10e3;
14     float NO2_ppm = ( 1 / NO2_M ) * (NO2_voltage-Vgas0) ;
15 }
16
17 Wire.endTransmission();     // stop transmission
```

4.1.3 MIKROE-3056 – Air Quality 5 click

MIKROE-3056, Figure 4.7, is a sensor module with a triple MOX sensor, MiCS-6814 from the Swiss company SGX Sensortech MiCS-6814 internal circuits, detecting different gases (CO, NO_2 and NH_3)[10,11]. This sensor requires a pre-time of 30 seconds to obtain valid data.

The MiCS-6814 has three different sensor chips in one package with independent heaters and sensitive layers. Each sensor chip detects different gases types: oxidising gases (OX sensor, detecting NO_2), reducing gases (RED sensor, detecting CO) and ammonia (NH3 sensor). The internal connections of the sensor are shown in Figure 4.8. The voltage values are converted to a digital value using the 12-bits A/D Converter

[10] MikroElektronika. Air quality 5 click | MikroElektronika. URL: https://www.mikroe.com/air-quality-5-click.

[11] SGX Sensortech. MiCS-6814 Data-Sheet.

ADS1015 (with internal reference and programmable comparator), produced by Texas Instruments [12], and available via I^{2}C interface. ADC requires to receive information from MCU about whose channel should read (OX, RED or NH3).

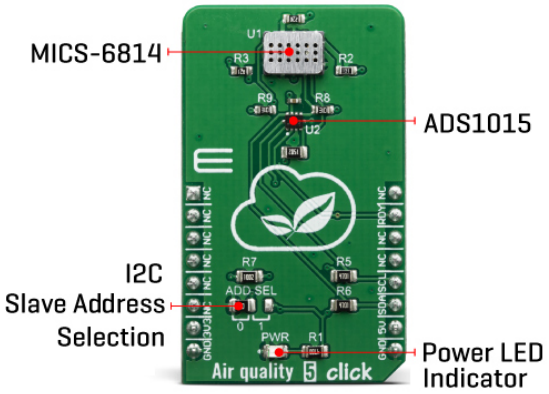


Figure 4.7: MIKROE-3056 marked with key components[10].

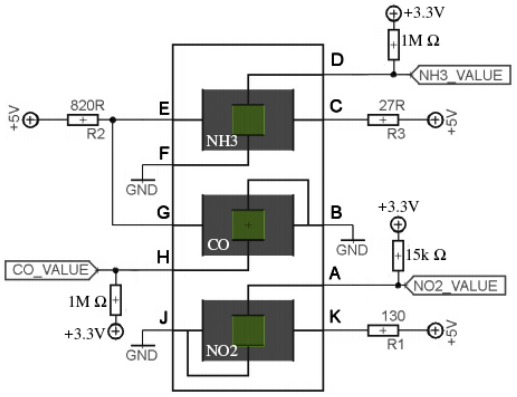


Figure 4.8: MiCS-6814 internal circuits[12].

4.1.3.1 I^{2}C communication, configuration and channel selection

The communication between MCU and the ADC is achieved via I^{2}C. The address of ADC is `0x48` (72 in decimal) or `0x4D` (77 in decimal), according to selection on-board (see "I2C Slave Address Selection" in Figure 4.7).

To obtain a sensor value, first it is needed to assign the pointer register of the ADC at the address `0x01` as the overlapping of:

- Setting configurations of the sensor. The most relevant parameters: device operating mode: single-shot mode, data rate: 1600 samples per second and, programmable gain amplifier configuration: ±2.048 V. Overlapping these instructions, the result is `0x8583`;
- Selecting the address of the channel to be read. `0x4000` for OX channel (NO_2), `0x5000` for NH3 channel and `0x6000` for RED channel (CO).

Below, it is presented the C++ code used to set this configurations before read data of NO_2 in the OX channel.

```
1 const uint16_t _AIRQ5_DATA_CHANNEL_NO2 = 0x4000;
2 const uint16_t setConfig = 0x8583;
3 const uint16_t _AIRQ5_REG_POINTER_CONFIG = 0x01:
4
5 Wire.begin(); // begin I2C communication
6 Wire.beginTransmission(72); // transmit to device 0x48=72;
7 uint8_t writeReg1[3];
8 uint16_t _data = setConfig | _AIRQ5_DATA_CHANNEL_NO2;
9 writeReg1[0] = _AIRQ5_REG_POINTER_CONFIG;
```

[12]Texas Instruments. ADS1015 Data-sheet. ADS101x Ultra-Small, Low-Power, I 2C-Compatible, 3.3-kSPS, 12-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator, 2018. URL: https://download.mikroe.com/documents/datasheets/ads1013.pdf.

```
writeReg1[1] = _data >> 8;
writeReg1[2] = _data & 0x00FF;
Wire.write(writeReg1,3);
Wire.endTransmission(); // stop transmission
```

Then, it is needed to request the ADC to perform one conversion of the input signal (once the "device operating mode" is set as "single-shot mode") and store that value in the internal conversion register, at address `0x00`. Below, it is shown the C++ code used in the MCU to request and read data of the OX channel.

```
uint8_t writeReg2[1];
writeReg2[0] = 0x00; // _AIRQ5_REG_POINTER_CONVERT
Wire.write(writeReg2,1);
Wire.endTransmission();

uint8_t readReg[2];
Wire.requestFrom(72, 2, 1);
if (Wire.available()){
    readReg[0] = Wire.read();
    readReg[1] = Wire.read();
}
Wire.endTransmission();

uint16_t dataValue = readReg[0];
dataValue <<= 8;
dataValue |= readReg[1];
dataValue >>= 4;
```

4.1.3.2 Concentration calculation

After the data request, the MCU receives a digital 12-bit value. The correspondent voltage value for the corresponding gas, V_{gas}, is given by:

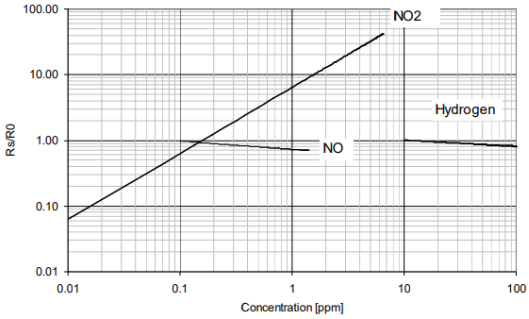
$$V_{gas} = \frac{ADCvalue}{2^{12}-1} \times V_{CC}, \text{ where } V_{CC} = 3.3 \text{ V}^{13}. \tag{4.8}$$

Figure 4.8 allows us to identify two independent circuits for each sensor chip, the heater circuit and the measurement circuit. The measurement circuit is composed of two resistances in series – R_{ext} (dependent of the gas) and R_S, the resistance induced by gas – composing a voltage divider referenced to ground[13]. Equation 4.9 gives the value of R_S.

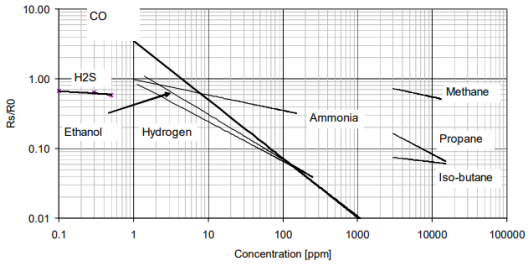
$$R_S = \left(\frac{V_{gas}}{V_{CC}-V_{gas}}\right) \times R_{ext}, \text{ where } \begin{cases} R_{ext} = 15\text{k}\Omega & \text{for OX sensor} \\ R_{ext} = 1\text{M}\Omega & \text{for RED sensor} \\ R_{ext} = 1\text{M}\Omega & \text{for OX sensor}^{13} \end{cases} \tag{4.9}$$

[13] MikroElectronika. Air quality 5 click schematic. URL: https://download.mikroe.com/documents/add-on-boards/click/air-quality-5/air-quality-5-clickschematic-v101.pdf.

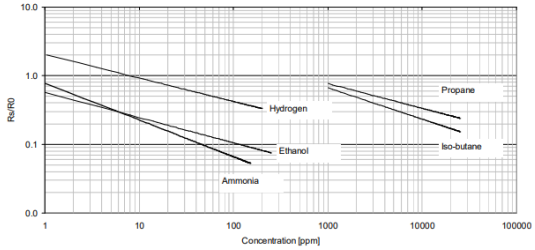
The relation between R_S and R_0 (the resistance of the the sensor in a clean-air environment) is given by curves present in Figure 4.9. Tracing the curves with Microsoft Excel, the same method used in Section 4.1.1.2, resulted in sensitivity equations by exponential regression, Equations 4.10a, 4.10b and 4.10c.



(a) MiCS-6814 OX sensor[11].



(b) MiCS-6814 RED sensor[11].



(c) MiCS-6814 NH3 sensor[11].

Figure 4.9: Curves of sensitivity of the different sensor chips of MiCS-6814.

$$x(NO_2) = \left(\frac{R_S}{R_0}\right)^{1.007} \times \frac{1}{6.8585} \tag{4.10a}$$

$$x(CO) = \left(\frac{R_S}{R_0}\right)^{-1.179} \times 4.385 \tag{4.10b}$$

$$x(NH_3) = \left(\frac{R_S}{R_0}\right)^{-1.67} \times \frac{1}{1.47} \tag{4.10c}$$

4.1.3.3 Code for this sensor

For NO_2, the resulting C++ code for these operations is:

```
1 R0 = 1000; // example value
2 float Vgas = (float(dataValue) / 2047) * 3.3;
3 float Rs = ((Vgas) / (3.3 - Vgas)) * 15000;
4 float ratio = Rs/R0;
5 float NO2_ppm = pow(ratio, 1.007) / 6.855;
```

4.1.4 Shinyei PPD42NS

PPD42NS, Figures 4.10a, is a sensor for particulate matter produced by the Japanese company Shinyei Technology and distributed by Chinese company Seeed Technology re-marketed as "Grove - Dust Sensor"[14]. This is a low-cost sensor designed to measure the concentration of particulate matter with at least 1 µm of diameter. This sensor uses a counting method to test dust concentration, but not the weight method, presenting the measurement results in particles/L (from now on, indicated as pcs/L). For the first time running, it needs 3 minutes to present reliable data. A blog author about Indian air quality, made a comparison by adding some features to improve PPD42NS data quality and found that the best results were by attaching a fan to the sensor and adding a 10 kΩ resistor between the Pin 5 (PPD42NS's data pin) and Ground[15]. This conclusion was subscribed by the sensor distributor Seeed Company[16].

4.1.4.1 Working principle

Based on the light scattering method, it detects airborne particles continuously. Pulse output that corresponds to concentration per unit volume of particles is obtained, using an approach based on the light scattered principle similar to the particle counter, as represented in Figure 4.10b. The heater (resistor) generates an updraft. The infrared light beam from LED has been focused by lens to sense point at the centre. Airborne particles have been taken into the sensor box with the updraft. Particles pass through the sensing point scattering the light, the receptor receives scattered light through the lens and transforms it into a pulse signal. Pulse per unit of time is proportional to the particle concentration. The pulse signal is converted into a voltage output. An output signal example is presented in Figure 4.10c and should be interpreted by an MCU capable of "reading" Power Width Modulation (PWM) (both MCU used, TTGO LORA32 and TTGO T-Beam, have this feature).

The sensitivity curve, Figure 4.10d, leads to the equation that relate pulse time with concentration of particulate matter, Equation 4.11.

$$percent = \frac{LowPulseTime\ [\mu s]}{SampleTime\ [ms] \times 10}\ \%$$

$$x(PM) = \left(1.1 \times percent^3 - 3.8 \times percent^2 + 520 \times percent + 0.620\right) \times \frac{1}{0.283}\ \text{pcs/L} \tag{4.11}$$

4.1.4.2 Code for this sensor

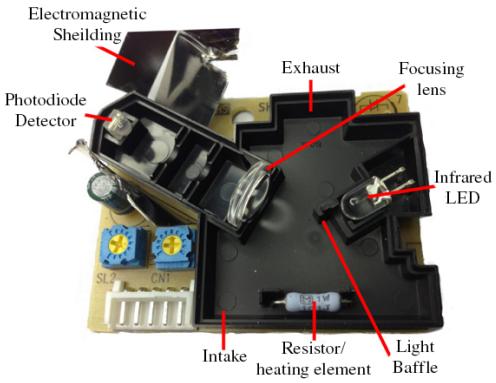
The C++ code running in `loop()` function for this reading and calculation is shown below. `sampletime_ms` represents the sample time. And, in `setup()` function, `lowpulseoccupancy`, `ratio` and `concentration` are defined as `0`.

```
1 while(isSampling){
2     duration = pulseIn(pin, LOW);
3     lowpulseoccupancy = lowpulseoccupancy + duration;
```

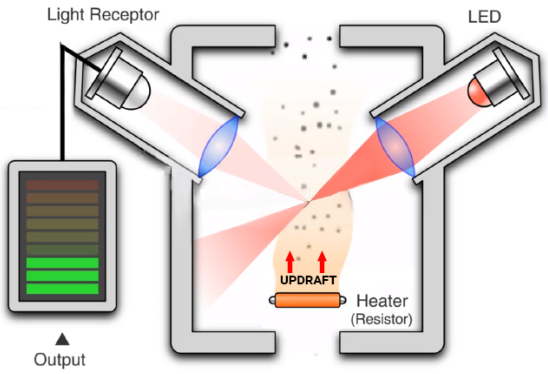
[14] Seeed Studio. Particle Sensor Model PPD42NS Data-sheet, 2010. URL: https://github.com/SeeedDocument/Grove_Dust_Sensor/raw/master/resource/Grove_-_Dust_sensor.pdf.

[15] A. J. Measuring the Pickle Jr. – a modified PPD42 with an attached fan., dec 2014. URL: https://indiaairquality.com/2014/12/14/measuring-the-pickle-jr-a-modified-ppd42-with-an-attached-fan/.

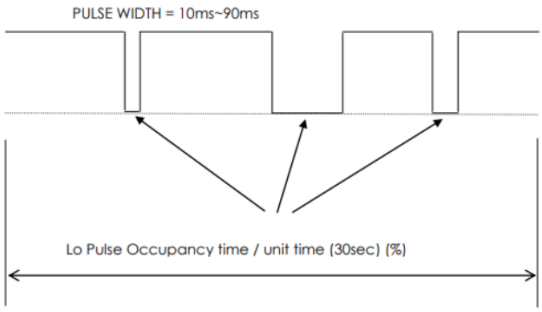
[16] Seeed Studio. Grove-Dust Sensor User Manual. Technical report, sep 2015.



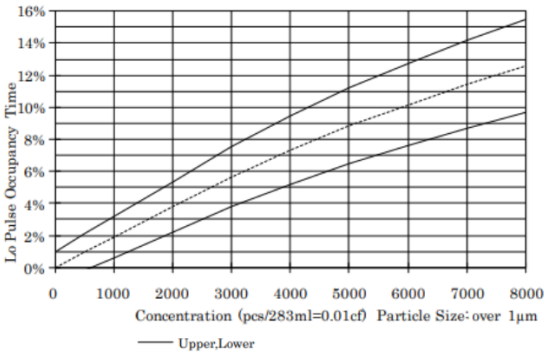
(a) Sensor without shield labelled[16].



(b) Principle of operation



(c) Example of an output signal[14].



(d) Sensitivity curve[14].

Figure 4.10: Shinyei PPD42NS sensor.

```
4
5       if ((millis() - starttime) > sampletime_ms)
6       {
7           ratio = lowpulseoccupancy / (sampletime_ms * 10.0);
8           concentration = 1.1 * pow(ratio,3) - 3.8 * pow(ratio,2) + 520 *
             ↪  ratio + 0.62;
9           concentration = concentration / 0.283;
10          lowpulseoccupancy = 0;
11          starttime = millis();
12          isSampling = false;
13      }
14  }
```

4.1.5 Sensirion SPS30

SPS30 is a real-time optical particle counter (OPC) made by the Swiss company Sensirion[17] and advertised by them as having “high-quality and long-lasting components, enabling precise measurements from

[17] Sensirion. SPS30 Data-sheet. Technical report. URL: www.sensirion.com.

the device's first operation and throughout its lifetime of more than ten years". This particulate matter sensor received an MCERTS certification recognising high performance and validating integration into applications that comply with the European Air Quality Standard DIN EN 15267-3. This sensor can count particles with more than 0.3 µm, and using "advanced" algorithms can exhibit results for $PM_{0.5}$, $PM_{1.0}$, $PM_{2.5}$, $PM_{4.0}$ and PM_{10} values with high precision. SPS30 support three modes operation modes: idle with consumption of $\approx$ 330 µA, measurement (1 second per measurement) with consumption between 45 and 65 mA and sleep mode with ultra-reduced functionalities consuming $<$ 50 µA. Also, an automatic fan-cleaning procedure is triggered periodically following a defined cleaning interval. This sensor communicates with MCU by sing I^2C interface (at address `0x69`).

4.1.5.1 Working principle

The working principle is similar to the OPC Shinyei PPD42NS, Section 4.1.4, being based on laser scattering, Figure 4.11c. A fan incites a controlled airflow (guaranteed by the internal microprocessor, Figure 4.11d), carrying the particles (black dots in Figure 4.11c) from inlet to outlet. Particles in the airflow pass through a laser beam (indicated in red in Figure 4.11c), causing a scattering of light. That scattered light is detected by the photodiode and converted into mass or concentration values by using algorithms executed on the internal microprocessor[18,19].

4.1.5.2 Library

Multiple libraries were developed for this sensor[20,21]. In this project, Paul van Haastrecht's library will be used in preference to the library developed by Sensirion, as it is more organised. In the Table 4.2, some relevant functions of this library are presented.

Table 4.2: Most relevant functions of SPS30 library.

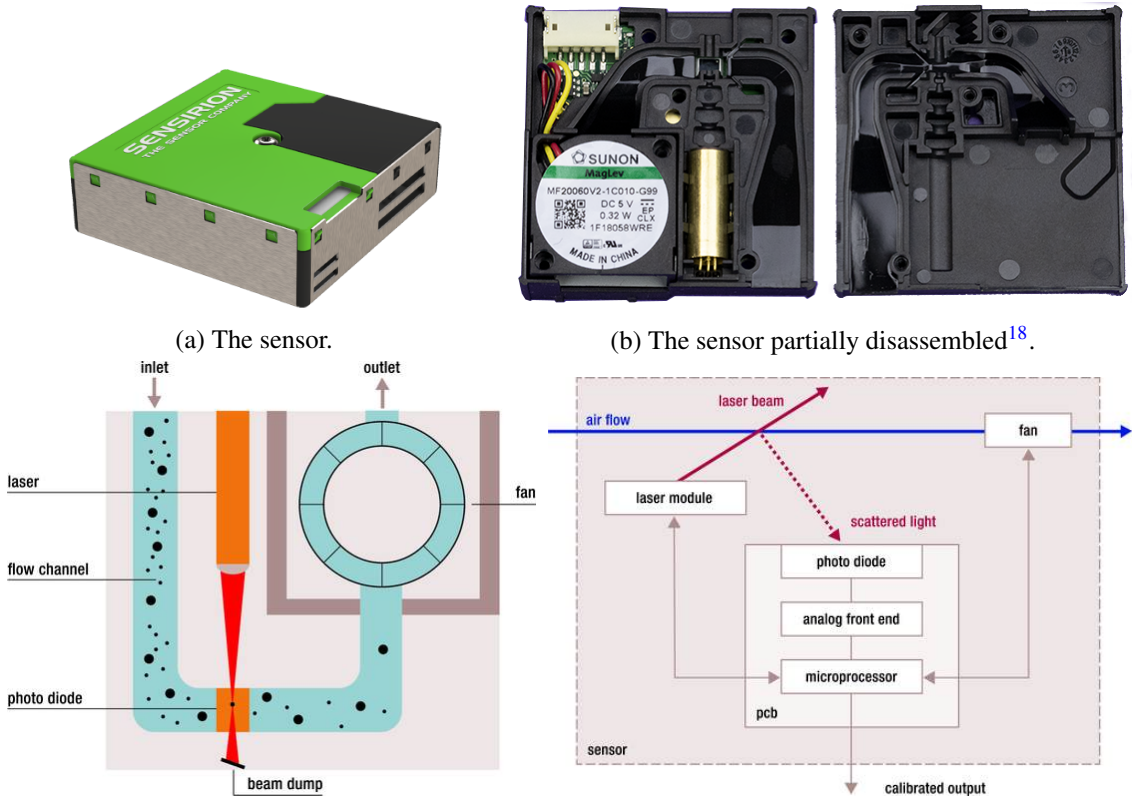
Function	Description
`begin(I2C_add)`	Initialises the I^2C communication, by giving slave address
`reset(), start(), stop()`	Performs SPS30 instructions
`clean()`	Starts the fan-cleaning manually
`sleep(), wakeup()`	Sets SPS30 to sleep or wakeup
`SetAutoCleanInt(uint32_t val)`	Set Auto Clean interval
`GetValues(struct values)`	Retrieves all measurement values from SPS-30
`GetMass2(), GetMass10()`	Returns the concentration of $PM_{2.5}$ or PM_{10}, in $\mu g/m^3$
`GetNumPM0(), GetNumPM1()`	Returns the number of $PM_{0.5}$ or $PM_{1.0}$, in particles/cm^3
`GetPartSize()`	Typical Particle Size, the average particle diameter, in µm

[18] Steve Cooper and Angela May. Teardown: Sensirion SPS30 Particle Matter Sensor, apr 2019. URL: https://www.mistywest.com/posts/teardown-sensirion-particle-matter-sensor/.

[19] Livio Lattanzio. Particulate Matter Sensing for Air Quality Measurements. Technical report, Sensirion AG, Zürich, Switzerland. URL: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Pictures/3_About_Us/Newsroom/Specialist_Articles/sps30/Sensirion_Specialist_Article_PM_final.pdf.

[20] Sensirion. GitHub - Sensirion/arduino-sps: Arduino library for Sensirion SPS30, dec 2020. URL: https://github.com/Sensirion/arduino-sps.

[21] Paulvha. GitHub - paulvha/sps30: Sensirion SPS30 driver for ESP32, SODAQ, MEGA2560, UNO, ESP8266, Particle-photon on UART OR I2C coummunication, nov 2020. URL: https://github.com/paulvha/sps30.



(a) The sensor.

(b) The sensor partially disassembled[18].

(c) Working principle[19].

(d) Block diagram. Extracted from[19].

Figure 4.11: Sensirion SPS30.

4.1.6 DFRobot Gravity BME680

BME680 is a sensor manufactured by the German company Bosch Sensortec[22]. This metal oxide sensor has a short response time (1 second) and I^2C interface (at address `0x77`), providing data about environmental factors (temperature, humidity and pressure) and air quality data (VOC level). To evaluate VOC, the sensor uses a software solution created through controlled lab tests, making correspondence between the resistance created by VOC and IAQ index. This IAQ index corresponds to an air quality level, from Excellent to Extremely Polluted, as shown in the Figure 4.12b. This sensor is integrated into a PCB sensor module produced by the Chinese company DFRobot[23], in Figure 4.12a.

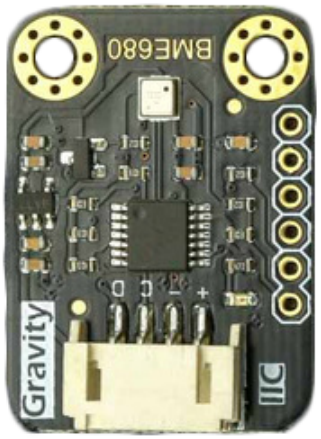
4.1.6.1 Library

Different libraries could be found for this wide-used sensor. This project uses the library from the sensor module's manufacturer[24]. Some relevant functions built-in this library are listed in Table 4.3

[22]Bosch Sensortec. BME680 Data-Sheet. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme680-ds001.pdf.

[23]DFRobot. Gravity: I2C BME680 Environmental Sensor – DFRobot. URL: https://www.dfrobot.com/product-1697.html.

[24]DFRobot. GitHub - DFRobotDFRobot_BME680, sep 2019. URL: https://github.com/DFRobot/DFRobot_BME680.



(a) Sensor module from DFRobot.

IAQ Index	Air Quality	Impact (long-term exposure)
0 – 50	Excellent	Pure air; best for well-being
51 – 100	Good	No irritation or impact on well-being
101 – 150	Lightly polluted	Reduction of well-being possible
151 – 200	Moderately polluted	More significant irritation possible
201 – 250[9]	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible

(b) Table for IAQ index from BME680 data-sheet[22].

Figure 4.12: DFRobot Gravity BME680 sensor details.

Table 4.3: Most relevant functions of BME680 library.

Function	Description
`begin()`	Begins the sensor and causes the settings to be loaded
`startConvert()`	Starts the conversion
`update()`	Updates all data to BME680's RAM
`iaqUpdate()`	Updates all data to BME680's RAM including IAQ
`ReadIAQ()`	Returns IAQ index, Figure 4.12b
`readTemperature()`	Returns temperature value, in degree Celsius, with a resolution of 0.01 ℃
`readHumidity()`	Returns relative humidity, in %, with a resolution of 0.01 %
`setGasHeater(T, t)`	Sets BME680 gas heater working t milliseconds at T ℃

4.1.7 Sparkfun SEN-14348 – Environmental Combo

The SparkFun SEN-14348[25] combines two popular low-cost environmental sensors, CCS811, produced by the Austrian company ams[26] and, BME280 produced by Bosch Sensortec[27].

4.1.7.1 ams CCS811

This sensor is an ultra-low-power digital gas sensor solution that integrates a MOX gas sensor to detect a wide range of Total Volatile Organic Compounds (TVOC) with a microcontroller unit (MCU), which includes an ADC (10-bit) and an I^2C interface. CCS811 provides an output of equivalent CO_2 (eCO_2) levels, using intelligent algorithms to process raw sensor measurements. This sensor also can compensate the gas readings by having access to current data and relative humidity (data provided by BME280). This sensor also calibrates itself after running 24 hours. *ams* recommends running the sensor 48 hours in the first

[25] Sparkfun. SparkFun Environmental Combo Breakout - CCS811/BME280 (Qwiic) - SEN-14348 - SparkFun Electronics. URL: https://www.sparkfun.com/products/14348.

[26] ams AG. CCS811 Data-sheet, feb 2019. URL: https://ams.com/documents/20143/36005/CCS811_PN000162_1-00.pdf.

[27] Bosch Sensortec. BME280 Data-sheet, nov 2020. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf.

use to "burn it in" and running 20 minutes in the desired mode every time the sensor is in use. The expected lifetime for this sensor is more than five years.

Sparkfun provides a library for this sensor containing functions to read data and make possible the integration between CCS811 and BME280[28]. The communication between CCS811 and MCU (TTGO LORA32) is possible using I^2C by the address `0x5B` or by the address `0x5A`.

Table 4.4: Most relevant functions of CCS811 library.

Function	Description
`beginWithStatus()`	Begins the sensor and prints error status
`dataAvailable()`	Boolean function to check if data is available
`readAlgorithmResults()`	Calling this function updates the global TVOC and eCO2 variables
`getCO2()`	Returns eCO_2, in ppm
`getTVOC()`	Returns TVOC, in ppb
`setEnvironmentalData(RH,T)`	Sends the temperature and humidity data to the CCS811
`checkForStatusError()`	Boolean function to check if there are an internal error
`getErrorRegister()`	Retuns internal error type

4.1.7.2 Bosch Sensortec BME280

BME280 provides relative humidity, temperature and barometric pressure. Has a short response time (1 second) and a I^2C interface (at the address `0x77` or at the address `0x76`) sending the converted values in ºC, Pa or %. These digital output values have a resolution of 0.01 ºC for temperature, 1 Pa for pressure and 0.001% for humidity (the ADC has 20 bits for temperature and pressure and 16 bit for humidity).

Table 4.5: Most relevant functions of BME280 library.

Function	Description
`begin()`	Begins the sensor and causes the settings to be loaded

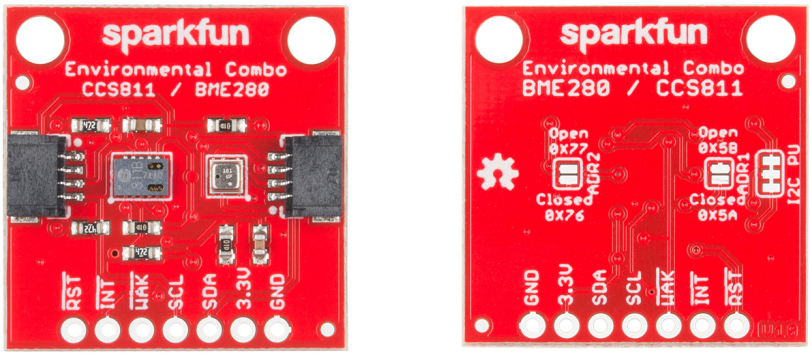
`settings`	Function to set configurations. Examples:
	`.commInterface = I2C_MODE`
	`.I2CAddress = 0x77`
	`.runMode = 3` (Normal mode)
	`.tStandby = 0`

`readTempC()`	Returns temperature, in degree Celsius, with a resolution of 0.01 ºC
`readFloatPressure()`	Returns barometric pressure, in Pascals, with a resolution of 1 Pa
`readFloatHumidity()`	Returns relative humidity, in %, with a resolution of 0.001%

4.2 Node assembly

After the analysis of the sensors and their operation, it is possible to make the assembly of MCU with them, according to the proposed solution in Section 3.3. It is important to note that connection wires should be as short as possible for I^2C due to electromagnetic interference.

[28] Sparkfun. GitHub - sparkfun/SparkFunCCS811_Arduino_Library: A library to drive the AMS CCS811 air quality sensor, apr 2020. URL: https://github.com/sparkfun/SparkFun_CCS811_Arduino_Library.



(a) Sparkfun SEN-14348 (front). CCS811 sensor (left) and BME280 (right).

(b) Sparkfun SEN-14348 (back). I^2C address selection.

Figure 4.13: Sparkfun SEN-14348 – Environmental Combo.

4.2.1 Node 1 – Outdoor

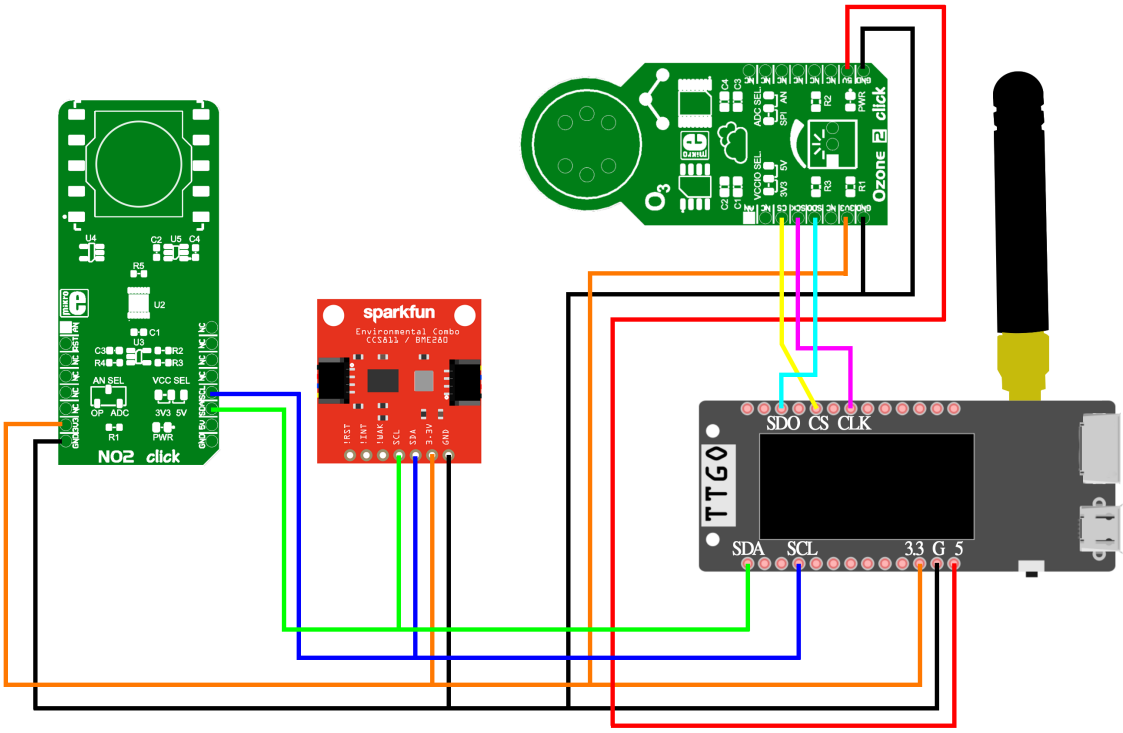
This node is composed by:

- ESP32-based LILYGO TTGO LORA32 unit;
- MIKROE-3098 (NO2 click);
- MIKROE-2767 (Ozone 2 click);
- Sparkfun SEN-14348.

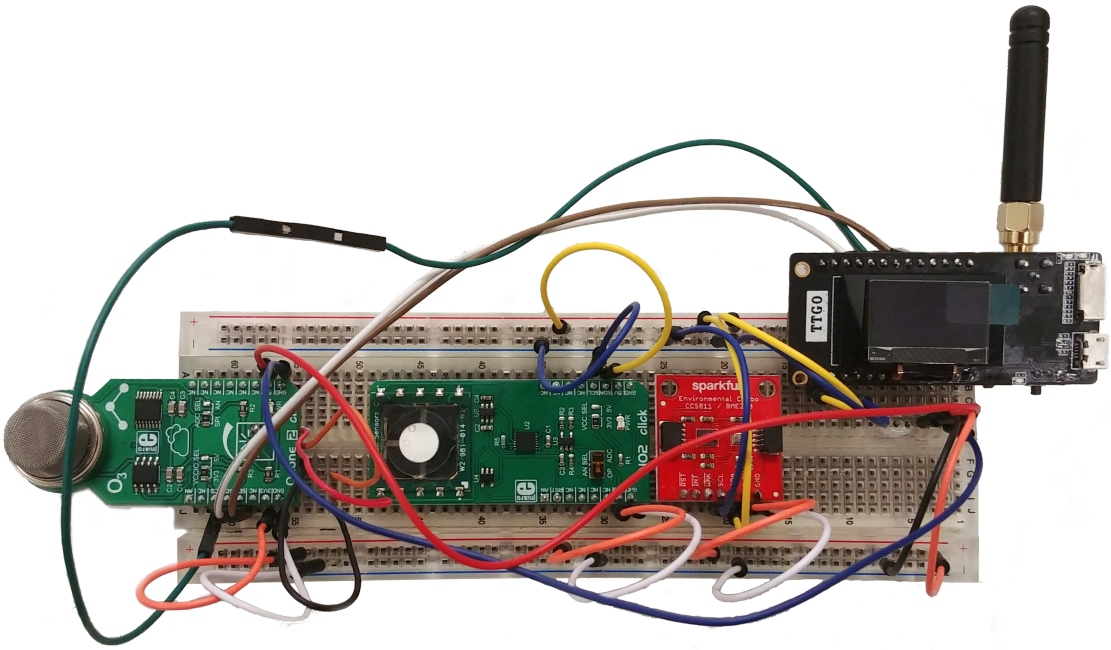
Figure 4.14 shows the diagram and the corresponding photograph of the assembly made on a breadboard. The Table 4.6 presents the pin connections between the MCU and the sensors.

Table 4.6: Pin connection for Node 1.

	TTGO LORA32 (MCU)	**NO2 click**	**Ozone 2 click**	**Sparkfun SEN-14348**
5 V	5V		5V	
3.3 V	3V3	3V3	3V3	3.3V
GND	GND	GND	GND	GND
I^2C - SCL	22/SCL	SCL		SCL
I^2C - SDA	21/SDA	SDA		SDA
SPI - SCK	14/CLK/SCK		SCK	
SPI - SDO	2/SDO/MISO		SDO	
SPI - CS	13/CS		CS	



(a) Assembly diagram of Node 1.



(b) Assembly of Node 1 in a breadboard (photo).

Figure 4.14: Assembly of Node 1.

4.2.2 Node 2 – Outdoor

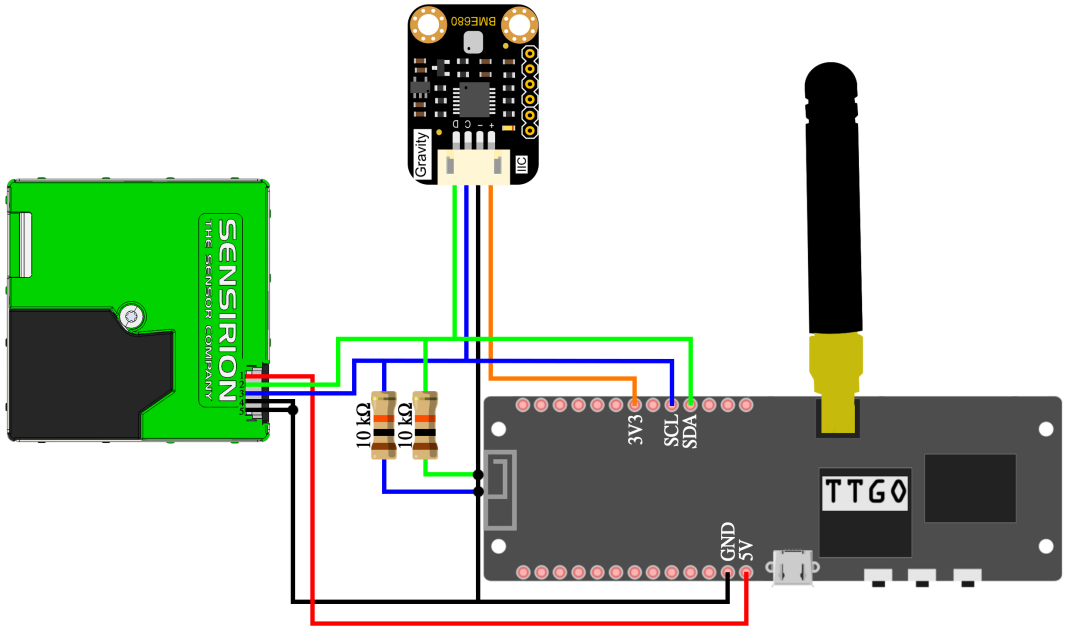
This node is composed by:

- ESP32-based LILYGO TTGO T-Beam unit;
- Sensirion SPS30;
- DFRobot Gravity BME680.

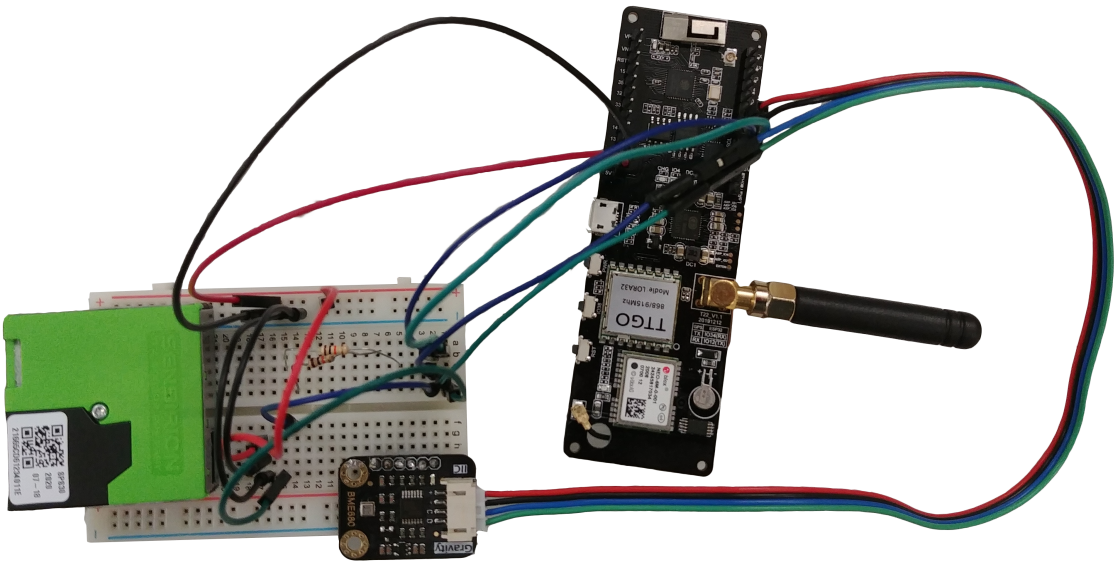
Figure 4.15 shows the diagram and the corresponding photograph of the assembly made on a bread-board. The Table 4.7 presents the pin connections between the MCU and the sensors.

Table 4.7: Pin connection for Node 2.

	TTGO LORA32 (MCU)	**Sensirion SPS30**	**DFRobot Gravity BME680**
5 V	5V	Pin 1/VDD	
3.3 V	3V3		+
GND	GND	Pin 4/SEL Pin 5/GND	-
I^2C - SCL	22/SCL	Pin 3/SCL	L
I^2C - SDA	21/SDA	Pin 2/SDA	D



(a) Assembly diagram of Node 2.



(b) Assembly of Node 2 in a breadboard (photo).

Figure 4.15: Assembly of Node 2.

4.2.3 Node 3 – Indoor

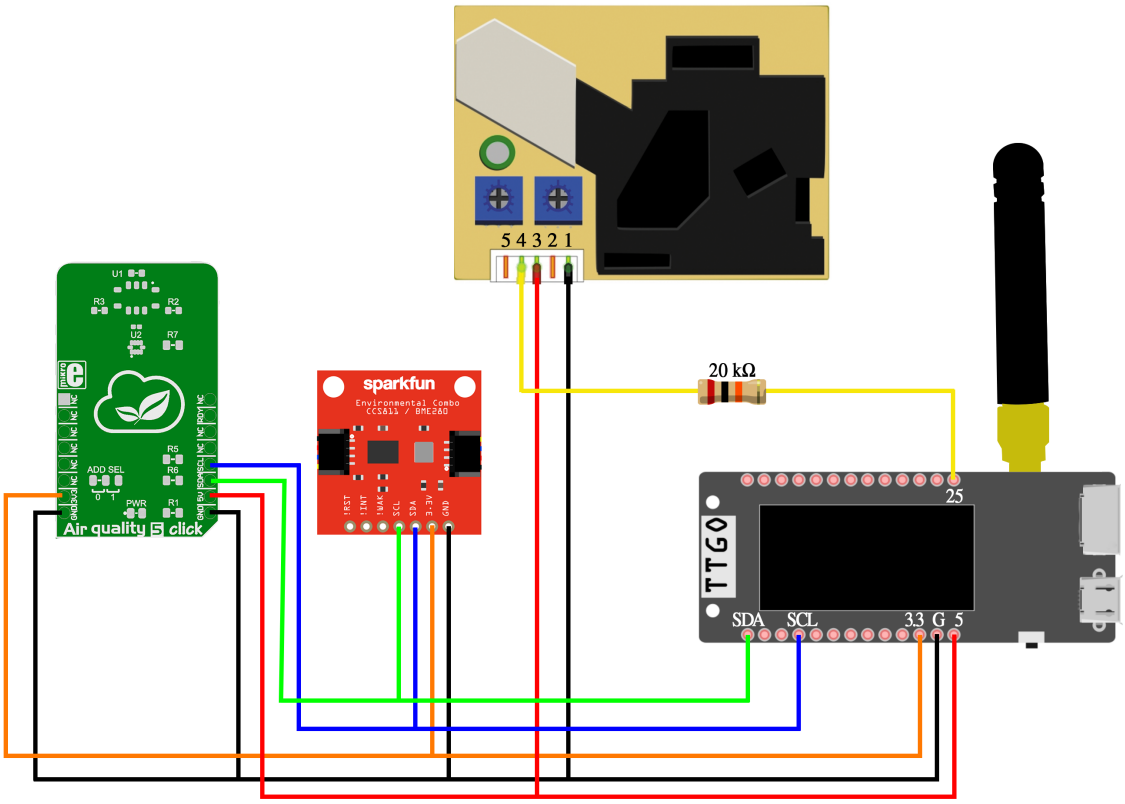
This node is composed by:

- ESP32-based LILYGO TTGO LORA32 unit;
- MIKROE-3058 (Air Quality 5 click);
- Sparkfun SEN-14348;
- Shinyei PPD42NS.

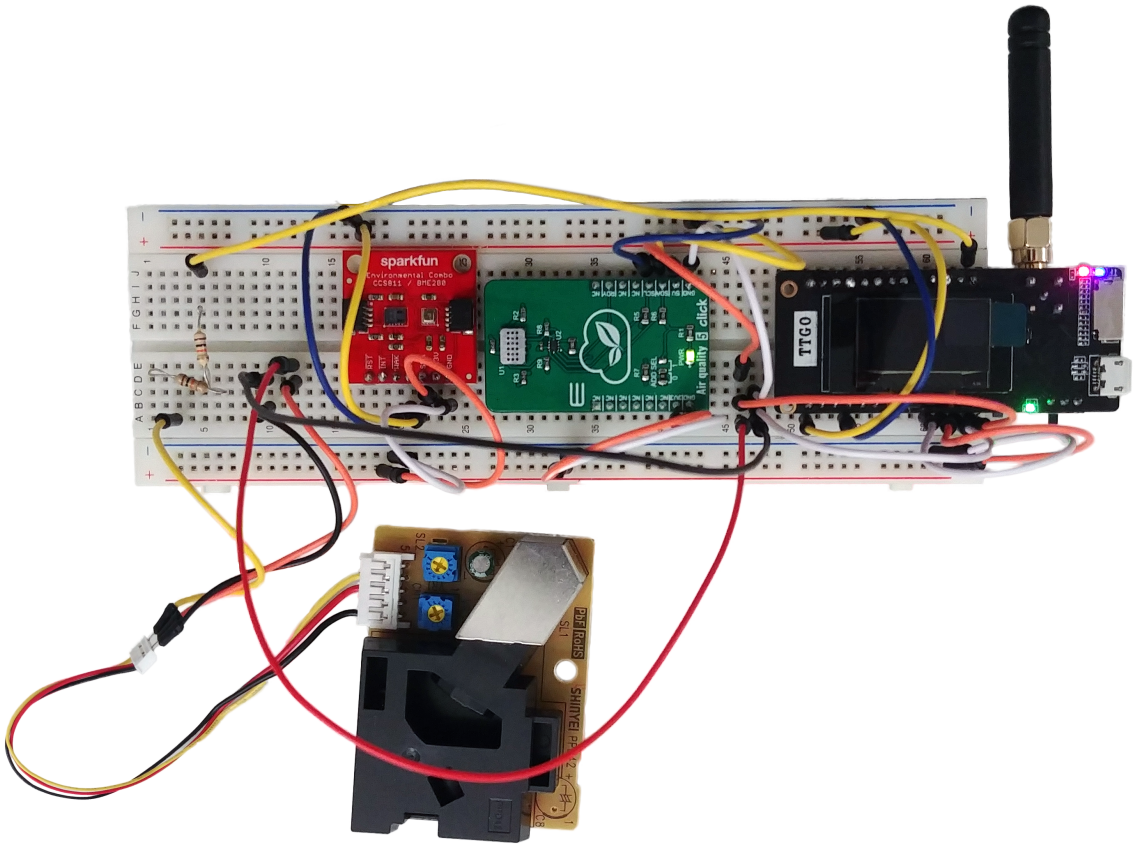
Figure 4.16 shows the diagram and the corresponding photograph of the assembly made on a bread-board. The Table 4.8 presents the pin connections between the MCU and the sensors.

Table 4.8: Pin connection for Node 3.

	TTGO LORA32 (MCU)	**Air quality 5 click**	**Sparkfun SEN-14348**	**Shinyei PPD42NS**
5 V	5V	5V		Pin 3/5V
3.3 V	3V3	3V3	3V3	
GND	GND	GND	GND	Pin 1/GND
I^2C - SCL	22/SCL	SCL	SCL	
I^2C - SDA	21/SDA	SDA	SDA	
	GPIO25			Pin 4



(a) Assembly diagram of Node 3.

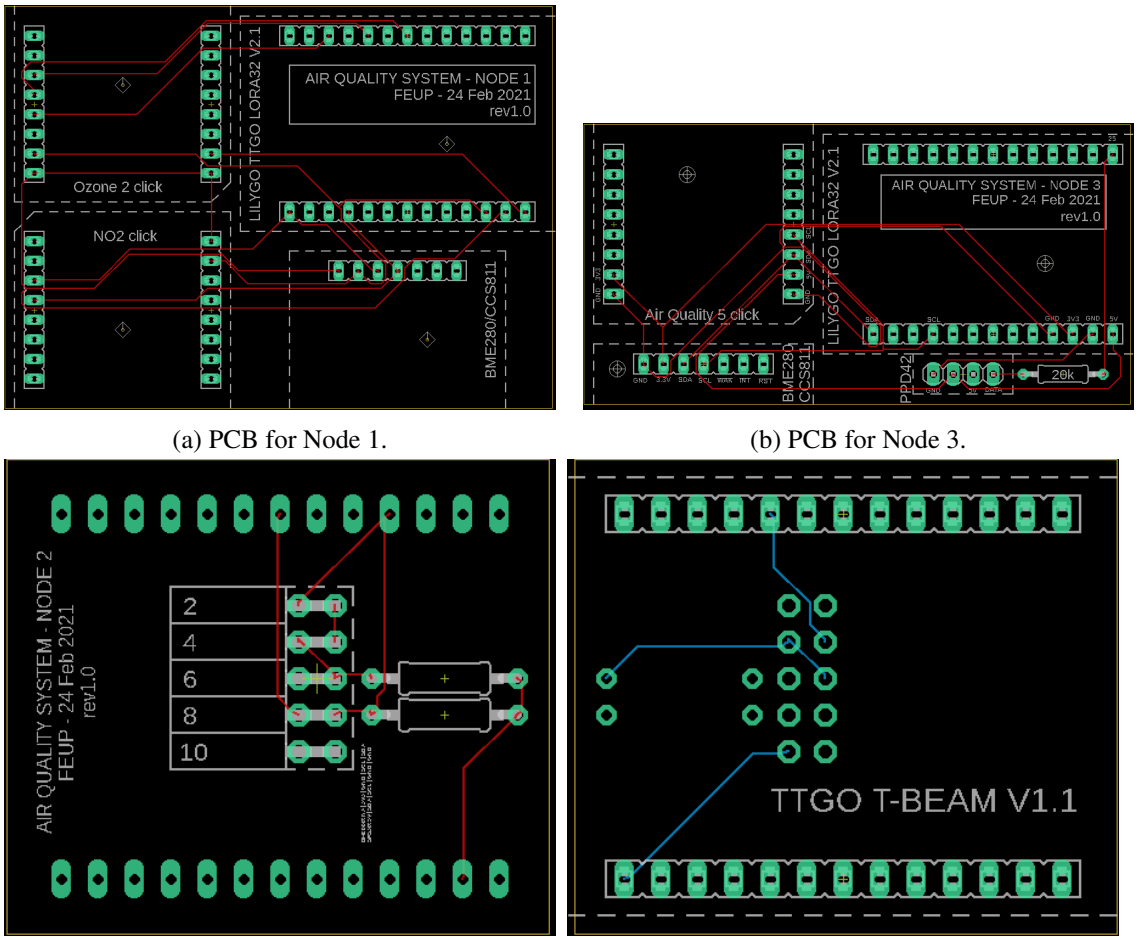


(b) Assembly of Node 3 in a breadboard (photo).

Figure 4.16: Assembly of Node 3.

4.2.4 Development of PCB for the Nodes

PCB (Printed Circuit Boards) were designed to make the prototypes compact in size and to provide immunity of the electronic system to movements using EAGLE, a software from the American company Autodesk. For Nodes 1 and 3, a single-sided PCB was designed, Figure 4.17a and 4.17b, respectively. On the other hand, the PCB for Node 2 is double-sided since it is necessary to couple components on both sides, Figure 4.17c and 4.17d. Elementary connection elements such as male and female pin headers would be attached to these PCB to connect the sensors and the MCU.



(a) PCB for Node 1.

(b) PCB for Node 3.

(c) PCB for Node 2 (top).

(d) PCB for Node 2 (bottom).

Figure 4.17: PCB for Nodes.

4.3 Power supply of nodes

As seen above, Node 1 and Node 3 are powered with a 5V DC power supplier (a smartphone charger or a PC USB port).

Node 2 ideally would be powered via a photovoltaic solar panel (PV). To design the PV it is crucial to understand the system's energy consumption and, consequently, how much energy the PV needs to produce.

4.3.1 Power consumption calculations of Node 2

Each duty cycle, the Node will acquire the data from both sensors (BME680 and SPS30). The BME680 sensor only consumes energy when data is requested. The PM sensor consumes a low quantity of energy even in sleeping mode, having a peak of energy consumption when data is requested due to fan acceleration and data processing. After the sensing process, the LoRa module routs the acquired data to Node 3 (connected to the server), requiring a high energy consumption (depending on the strength of the signal used to transmit). After that, the microcontroller and every module will enter sleep mode, saving energy. Every process and the associated consumption is shown in Table 4.9.

The total daily consumption is 1.154 Wh, value given by calculation, for the longest possible times to execute the most consumption processes of the duty cycle, i.e. for the worst-case scenario. For sizing the solar panel, the daily consumption will be multiplied by a safety factor of 3 for another not considered spends (e.g., daily fan auto-cleaning) and because the sleeping times could be changed via software. The final daily power consumption of Node 2, rounded up, is 1.15 Wh.

Table 4.9: Worst-case scenario for a Node 2 duty cycle.

Process	**Time**	**TTGO T-Beam consumption**	**Sensor consumption**	**Total energy consumption**
Sensing with BME680	2 s	44 mA (experimental) @ 3.7 V = 162.8 mW (idle)	BME680: 5 mA @ 3.3 V (sensing) SPS30: 50 µA @ 5V (sleep) = 16.75mW	179.55 mW 0.0998 mWh
Sensing with SPS30: Turning on the fan	0.75 s	44 mA @ 3.7 V = 162.8 mW	SPS30: 80 mA @ 5V = 400 mW (sensing)	562.8 mW 0.117 mWh
Sensing with SPS30: Reading data	1.25 s	44 mA @ 3.7 V = 162.8 mW	SPS30: 65 mA @ 5V = 325 mW (sensing)	487.8 mW 0.169 mWh
Sensing with SPS30: Sleeping and waking up	2 s	44 mA @ 3.7 V = 162.8 mW	SPS30: 330 µA @ 5V = 1.5 mW (idle)	164.3 mW 0.0712 mWh
LoRa transmittion: Sending data	12 s	(44 + 120[6]) mA @ 3.7 V = 606.8 mW	SPS30: 50 µA @ 5V = 0.25mW	607.05 mW 2.023 mWh
LoRa transmittion: Waiting for feedback	13 s	44 mA @ 3.7 V = 162.8 mW	SPS30: 50 µA @ 5V = 0.25mW	163.05 mW 0.589 mWh
Other waiting times	9 s	44 mA @ 3.7 V = 162.8 mW	SPS30: 50 µA @ 5V = 0.25mW	163.05 mW 0.408 mWh
MCU and sensors sleeping	10 min	10 mA @ 3.7V = 37 mW (sleep)	SPS30: 50 µA @ 5V = 0.25mW	101.59 mW 5.07 mWh
TOTAL	10 min 40 s			**8.55 mWh**
	1 day			**1.154 Wh (135 cycles)**
	1 year			**768.34 Wh**

4.3.2 Design the photovoltaic solar panel for Node 2

For safety reasons, a 03962A charger module will be used[29], guaranteeing:

- Constant current charging the lithium battery;
- Over-discharge protection, preventing the battery from discharging below 2.4V;
- Over-charge protection, safely charging the battery to 4.2V;
- Soft-start protection, limiting rush current.

Using the charger module enables the PV cell to charge the battery and supply the node circuit (Figure 4.15a). When no solar power is available, the battery is used as the power source of the node circuit, as represented in Figure 4.18.

[29] Amazon UK. TP4056 5V 1A Micro USB 18650 Lithium Battery Charging. URL: https://www.amazon.co.uk/TP4056-Lithium-Charging-Protection-Function/dp/B07BSVS842.

Instituto Português do Mar e da Atmosfera (Portuguese Sea and Atmosphere Agency) indicates a mean value of 2393.6 hours of sunlight per year for the Portuguese city of Porto (where the Node 2 will be installed)[30]. The smallest monthly sunlight is in December, 112.2 hours of sunlight per month, giving a mean of 3.62 hours of sunlight per day.

Using this information, it is possible to calculate the minimum necessary power of the solar panel to power the node in December:

$$P_{PV} = \frac{Energy\ [\text{Wh}]}{time\ [\text{h}]} = \frac{3.463\ [\text{Wh}]}{3.619\ [\text{h}]} = 0.957\ [\text{W}]. \tag{4.12}$$

A photovoltaic cell of 1 W and 6 V is represented in the mounting schematic of Figure 4.18.

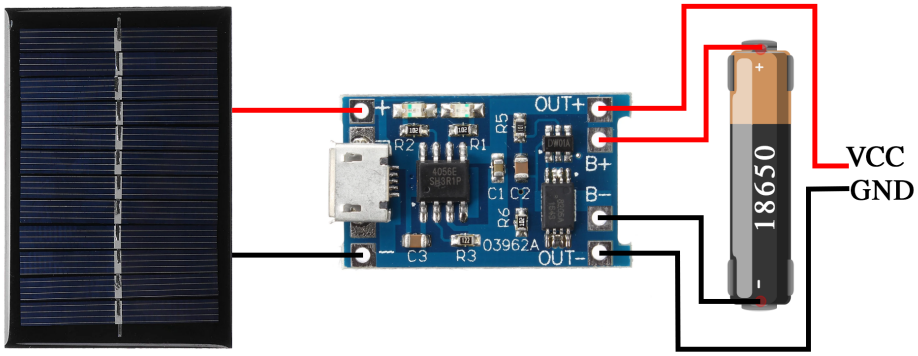


Figure 4.18: Photovoltaic solar panel circuit schematic.

4.3.3 Battery requirements of Node 2

Considering the battery used by the TTGO T-Beam is a 3.7 V lithium battery, reference 18650, and the market offers units of these batteries with electric charge of 2.6 Ah[31], it is interesting to calculate how long the battery lasts without photovoltaic cell. Equation 4.13 gives the duration of the battery.

$$P_{consumption}\ [\text{W}] = \frac{Energy\ [\text{Wh}]}{time[h]} = \frac{1.154\ [\text{Wh}]}{24\ [\text{h}]} = 0.048\ [\text{W}]$$

$$Electric_charge\ [\text{Ah}] = \frac{P_{consumption}\ [\text{W}]}{Voltage\ [\text{V}]} \times time\ [\text{s}] \Longleftrightarrow time\ [\text{h}] = 200\ \text{hours} \approx 8.33\ \text{days} \tag{4.13}$$

4.4 Development of a case for mechanical protection

Nodes 1 and 3 were designed to be powered in a controlled ambient, it is not expected that theses nodes require a special physical protection.

Node 2 needs special protection against adverse weather events such as rain and strong wind. For this reason, a protective case was modelled, Figure 4.19. The simple construction allows that it could be printed using a 3D printer. This design attended to the following requirements.

[30] Instituto de Meteorologia, I.P. Ficha Climatológica 1971-2000, Porto/Serra do Pilar. Technical report, Instituto de Meteorologia, I.P., Lisbon, Portugal, 2001. URL: http://www.ipma.pt/bin/file.data/climate-normal/cn_71-00_PORTO_SERRA_PILAR.pdf.

[31] Samsung SDI Co Ltd - Energy Business Division. ICR18650-26F Data-sheet. Technical report, nov 2009.

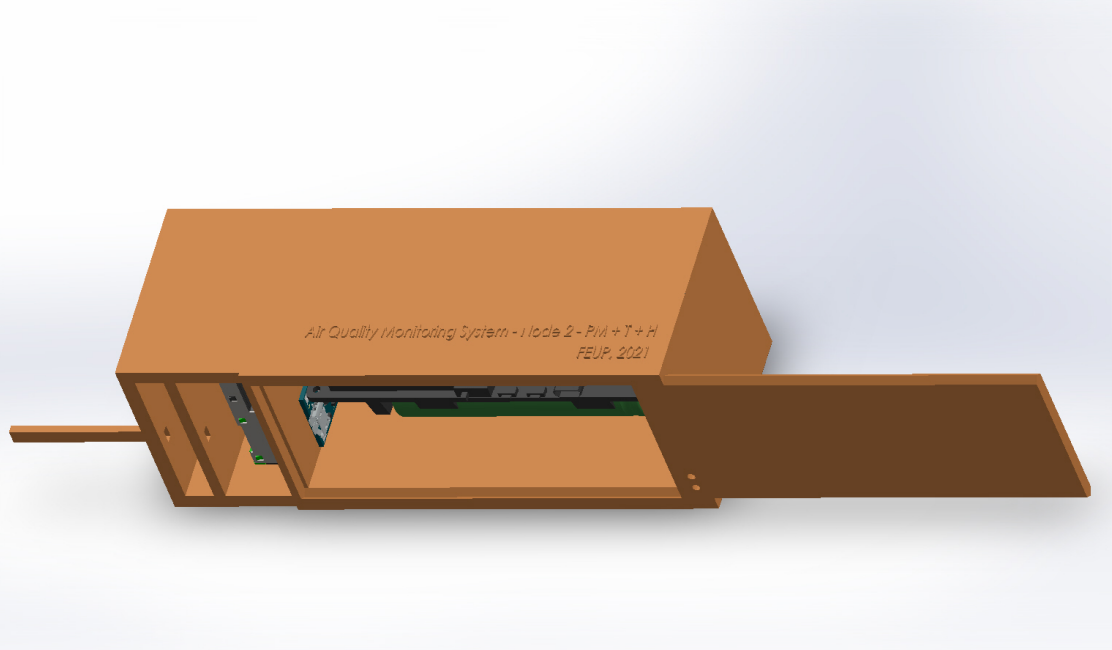
- Capacity for the MCU (TTGO T-Beam), the sensors and the PCB;
- Separated spaces for the sensor, once SPS30 vibration could affect BME680's results;
- Open contact of the sensors to air;
- Spaces to wire connection between MCU and the sensor, and between photovoltaic cell (not represented) to the battery;
- Elements to facilitate the installation of the equipment;
- Removable cover to place the MCU and the sensors.

4.5 Summary

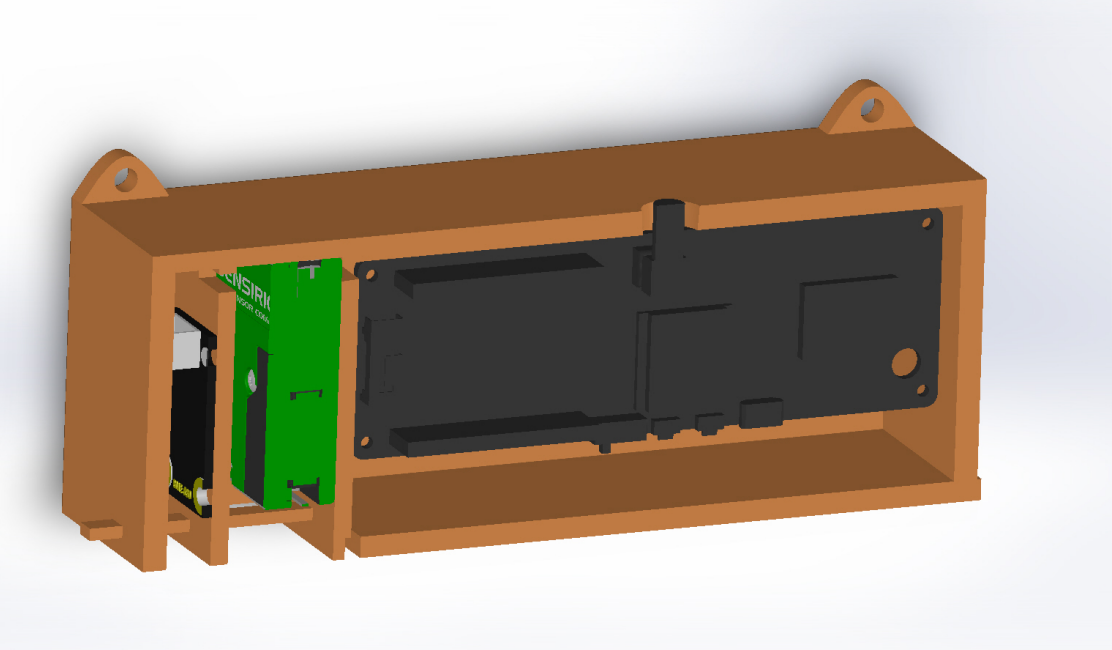
The sensors used have distinct differences, in the parameters that they can measure and in the communication interface used. Libraries for some sensors are found on the Internet, some of them created by the manufacturers easing data-sheet interpretation. Other sensor modules do not provide a complete data-sheet of the module, and the output values are obtained through the analysis of the distinct components of the sensor module.

A prototype per each node was mounted. The sensors are connected to the microcontrollers using wires and breadboards. Printed Circuit Boards were designed to replace them, reducing the space used by the nodes.

Some nodes have special requirements, having energy limitations and resistance to weather conditions. For that reason, a case and power supply using PV cell were projected.



(a) Front/bottom 3D view of the case.



(b) Cutted 3D view of the case.

Figure 4.19: Snapshots of designed case for Node 2.

Chapter 5

Software design

This chapter approaches the working dynamic of the system, illustrating the algorithm employed through flowcharts.

5.1 Node 1 (outdoor)

This node was planned to be powered by a 5V DC power supply, so the software design of Node 1 will not implement sleeping or deep-sleeping modes of the MCU LILYGO LORA32.

As represented in Figure 5.6, after the initialisation process, the sensors will return pollution and environmental parameters to the MCU. The MCU will redirect these data to the indoor node through LoRa modulation after collecting sensor data twice (to guarantee the quality of data and internal calibration and correction of the sensor CCS811). MCU will attempt to transmit data to the indoor node (every 2 seconds) until either it receives a feedback message from that node or 25 seconds have elapsed. It ensures that data is received by the indoor node, even if this node is in its sensing routine.

5.2 Node 2 (outdoor)

This node was projected to have power consumption limitations being powered by a battery and a photovoltaic solar panel, as referred in Section 4.3. So, this node should take advantage of the sleep modes of the microcontroller and the sensors.

As illustrated in Figure 5.7, the initialisation process is similar to Node 1. The main difference is that the sensors enter in sleep mode after readings and wake up before readings (the BME680 is not turned on when it is not reading). Also microcontroller enter in a low-power sleep mode when waiting for the next measurement.

5.3 Node 3 (indoor)

Node 3 was designed to be the indoor, and it is connected to the Internet publishing the data of all the nodes, through MQTT. So, this node must always be awake collecting data from its sensors or receiving from other nodes. As reflected on the flowchart in the Figure 5.8, after setting up the ESP32 and the

sensors, the sensor will make two measurements before sending them to the MQTT broker. Before each data collecting routine, the MCU enters in a "Receive mode", receiving data from other sensors and sending it to the MQTT broker. In the "Receive mode", the MCU will try to parse packets received by the LoRa module present on the LILYGO LORA32 board. If the module captures something valid (data specifically addressed for this node), data will be analysed and the measurements are isolated and sent to the MQTT broker. When some data is required to be sent to the server via MQTT, the MCU checks if it is connected to the Internet and if the MQTT connection is valid. After these conditions are verified, the strings containing data will be published to the respective topics.

5.4 Code implementation

Flowcharts presenting nodes behaviour were converted to C++ code using Arduino IDE, Figure 5.1, and, afterwards, compiled and upload to the MCUs using the same software. The complete programming code is presented in Appendix A.

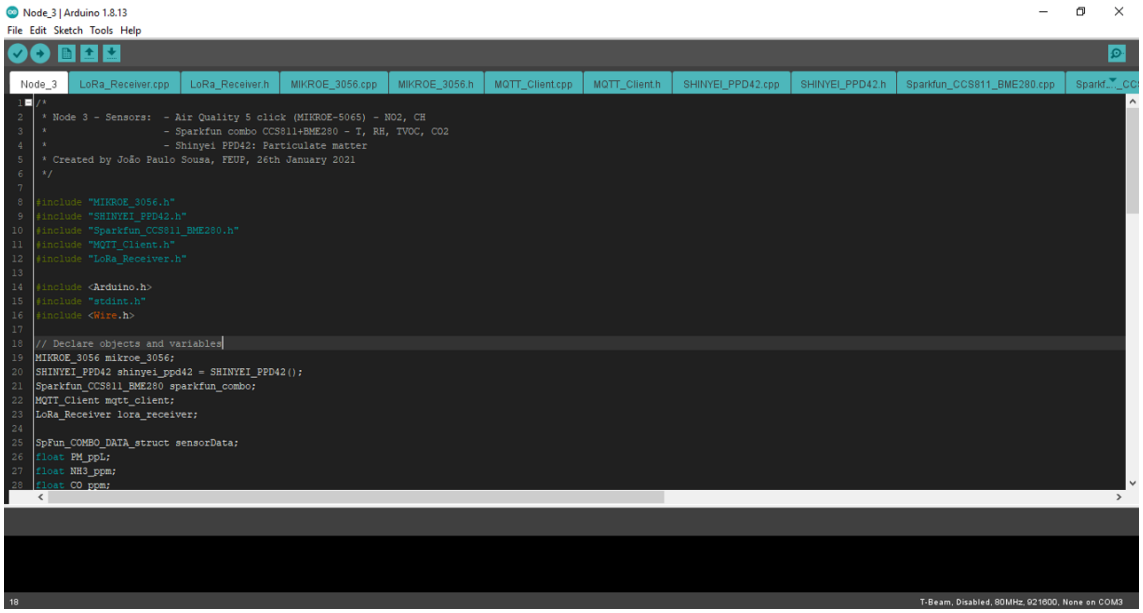


Figure 5.1: Screenshot of Arduino IDE

5.5 Configuration and connection to the MQTT Broker

To use the MQTT protocol and to present the data, external services are required. The market offers numerous services [1] for MQTT broker/server and display data. The outstanding services aggregating the MQTT broker and dashboards for displaying data are Cayenne myDevices and Ubidots. These two services are similar, but Ubidots present a cleaner interface, the reason for the choice[2,3].

[1] MQTT. Software. URL: https://mqtt.org/software/.
[2] Ubidots STEM. Ubidots STEM. URL: https://stem.ubidots.com/.
[3] Cayenne myDevices. Cayenne myDevices. URL: https://cayenne.mydevices.com/.

The chosen MQTT broker, Ubidots, defines "dot" as "a data point containing a value and a timestamp". Ubidots has a throttling limit of 30 dots per minute across all devices and up to 4,000 dots per day. Node 1 sends 7 values every duty cycle (temperature, relative humidity, pressure, CO_2, TVOC, NO_2 and O_3). Node 2 sends 6 values every duty cycle (temperature, relative humidity, pressure, $PM_{1.0}$, $PM_{2.5}$ and PM_{10}). Node 3 sends 9 values (temperature, relative humidity, pressure, CO_2, TVOC, NO_2, CO, NH_3 and PM). Thus, the sensors send a total of 22 values. For this reason, it is necessary to adjust the sampling time (time between measurements) to at least 7.92 minutes to comply the Ubidots requirements. This limitation is presented in flowcharts presented in this Chapter.

After creating an account in Ubidots, it is possible to configure the devices that will send data to the server, Figure 5.2. To simplify this process and, because, in our system only one device communicate with the server, only one device will be configured. To distinguish the source of the data, the name of variables is preceded by the number of the node. For example, variable "3t" is labled as "3_Temperature" and represents the variable Temperature of the Node 3, as can be seen in Figure 5.3.

The programming code used to connect Node 3 and MQTT broker is presented in Sections A.3.10 and A.3.11.

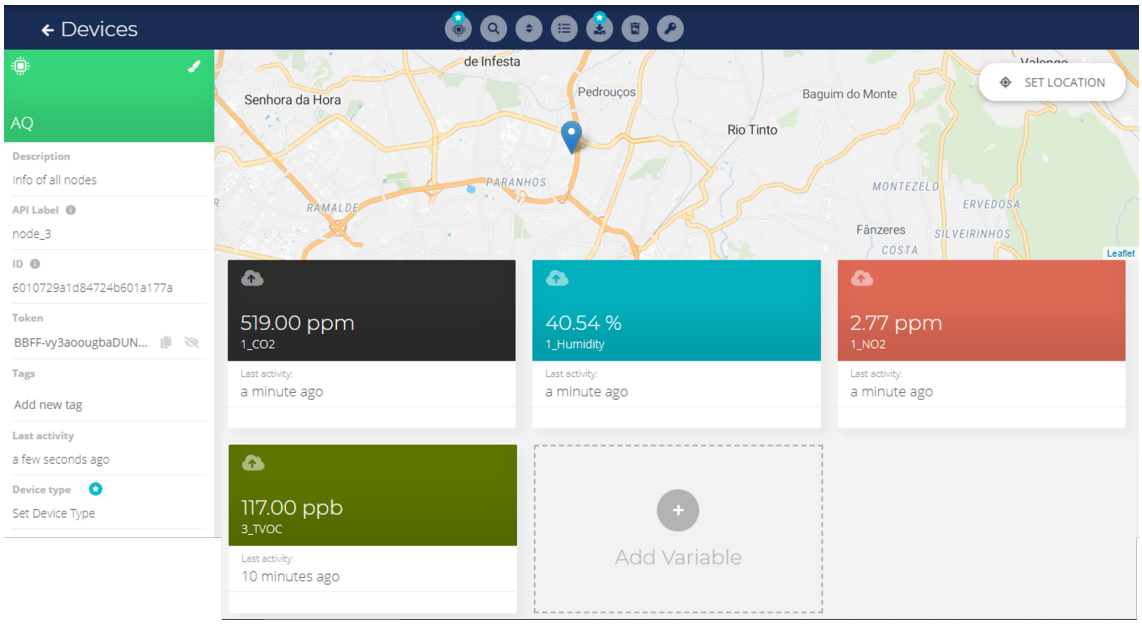


Figure 5.2: Screenshot of device configuration panel in Ubidots platform, presenting environmental variables.

5.6 Creation of the dashboard

After the configuration of the device and variables, the next step is to create a dashboard to display the information. Ubidots allows to use different types of widgets to present data, such as gauges, thermometers, different type of charts and personalised HTML widgets, as seen in Figure 5.4.

The dashboard should present organised information by node/collecting local, prioritise simplicity, and present some real-time data and evolution of gas concentrations for the last 24 hours. The dashboard with results is presented in the next Chapter (Figure 6.2).



Figure 5.3: Screenshot of variable configuration in Ubidots platform, displaying the existing data.

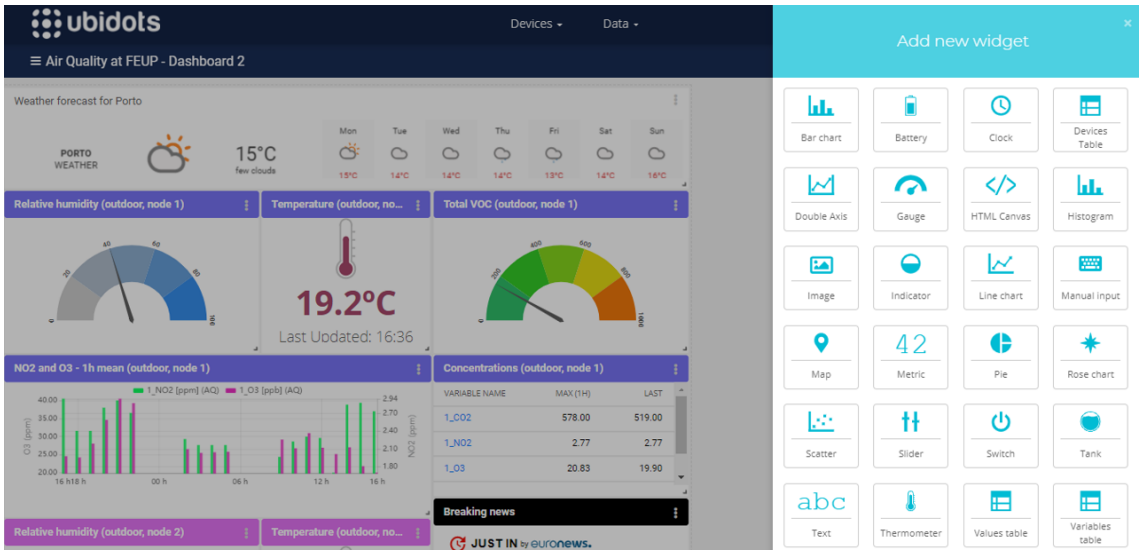


Figure 5.4: Screenshot of dashboard, displaying the existing widgets provided by Ubidots.

5.7 Alerts for poor air quality

It is possible to configure alerts. Ubidots allow actions such as SMS, voice call, e-mails, Slack messages and set of a variable when a programmed event happens. This event could be triggered when some variable is greater or less than determined value. Figure 5.5 presents screenshots of the SMS alert created, the action is triggered when value of CO concentrations is greater than 2000 ppb for 30 minutes.

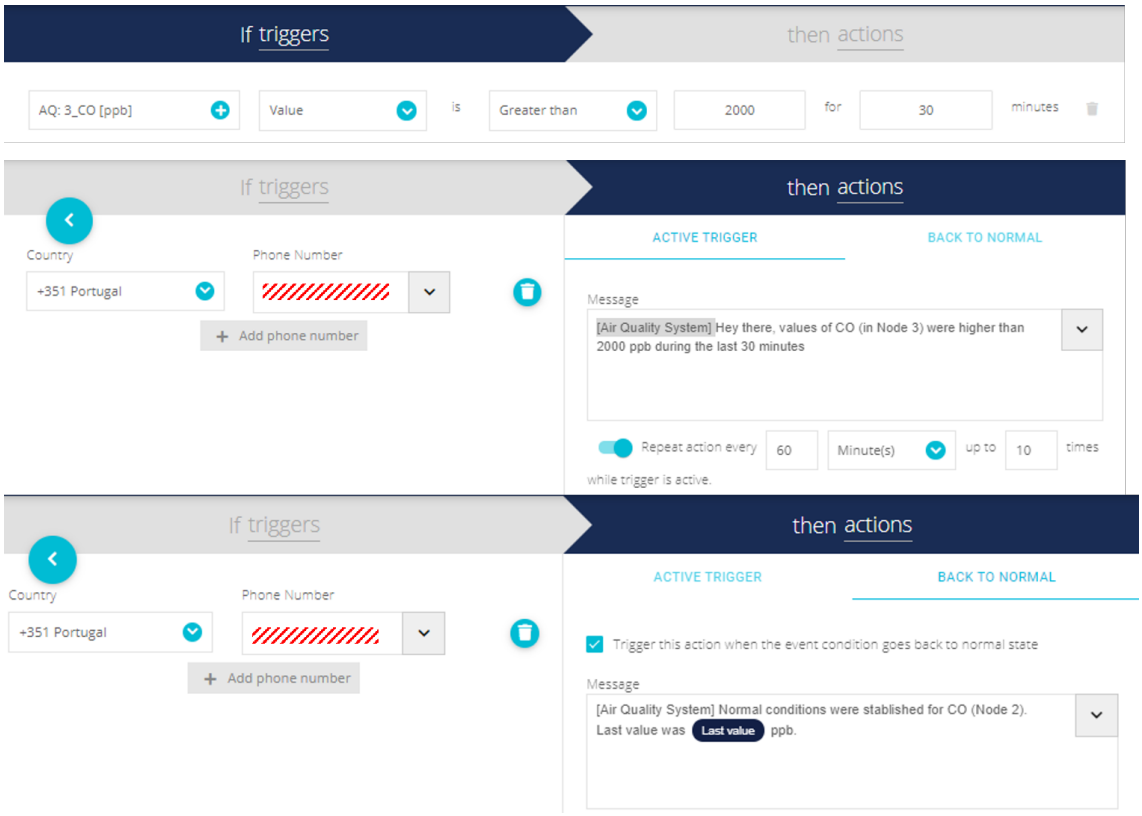


Figure 5.5: Screenshots of configuration of an SMS alert when CO is greater than 2000 ppb for 30 minutes.

5.8 Summary

To control the behaviour of the system, it is necessary to programme the actions that should happen and when they should happen. This system was designed to send data to a server each 10 minutes, verifying, every cycle, if all the communication system is OK. This Chapter presents the flowcharts of the three nodes and how they were implemented,

This Chapter also presents the configuration of the MQTT broker and the dashboard displaying data provided by the sensors, from the three nodes.

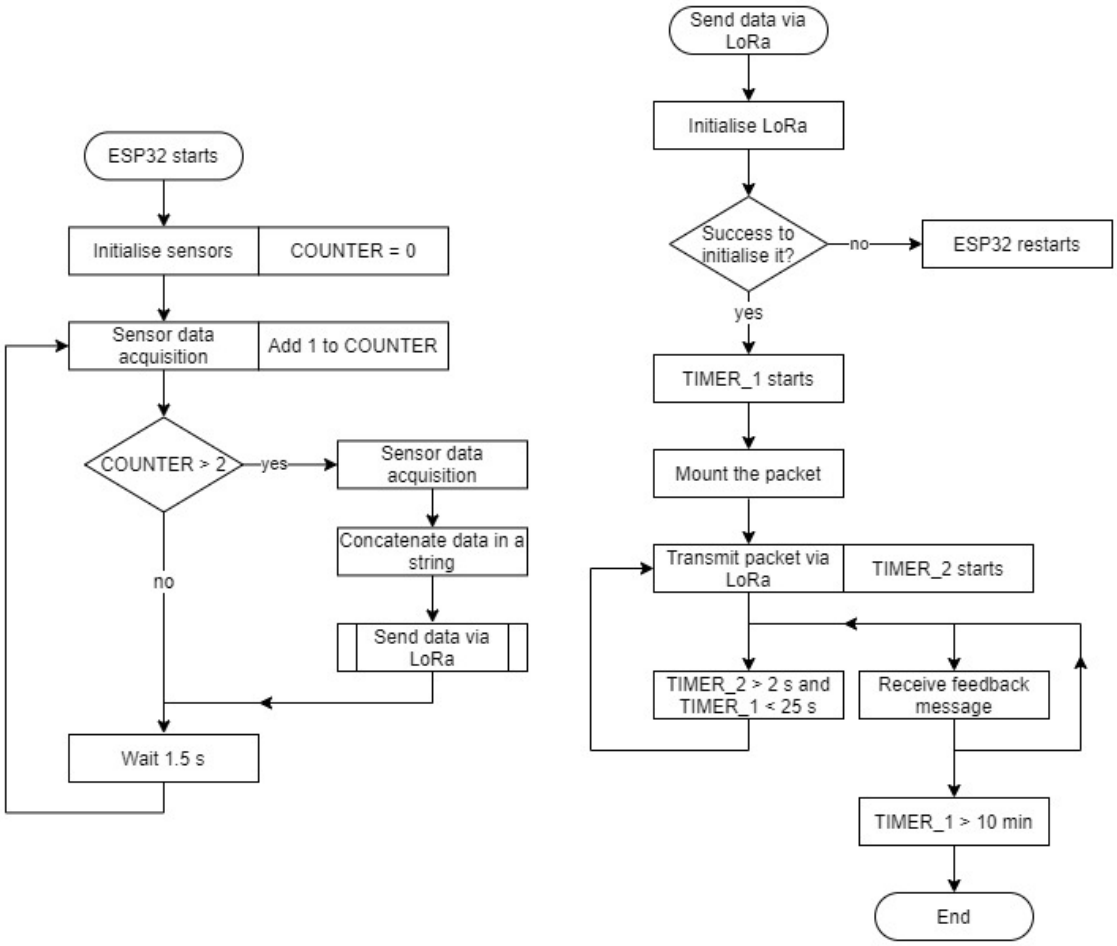


Figure 5.6: Flowcharts for Node 1.

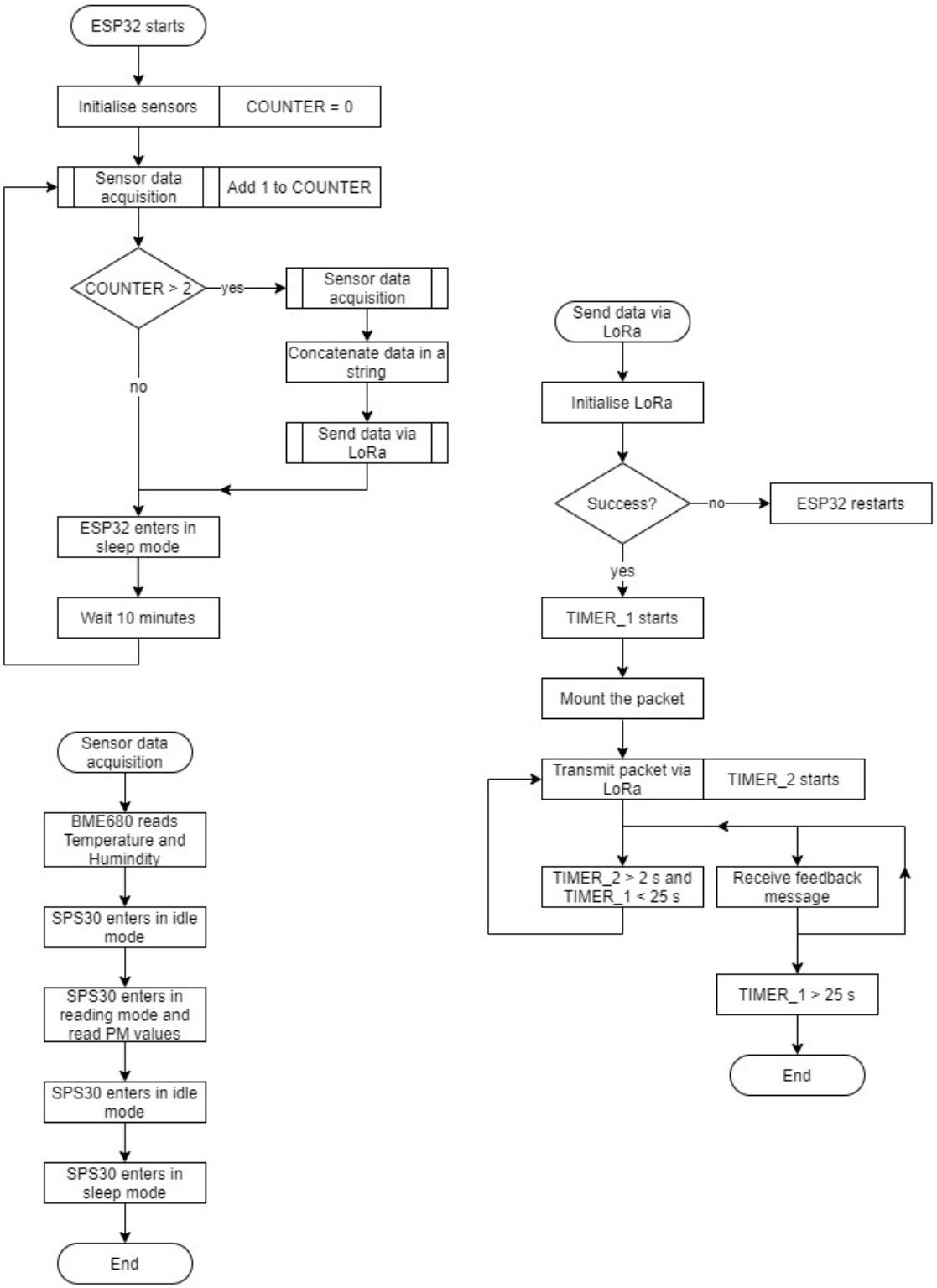


Figure 5.7: Flowcharts for Node 2.

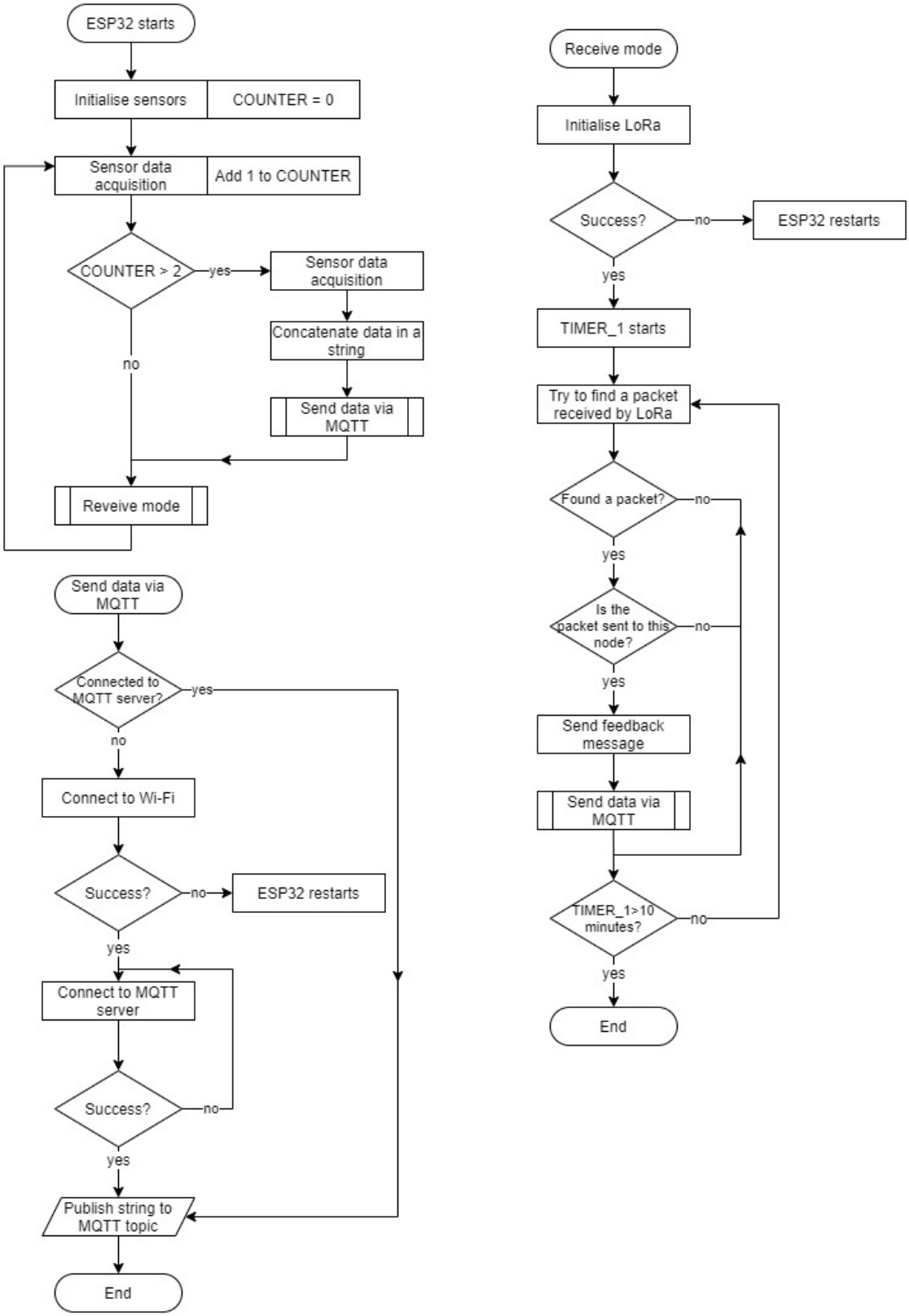


Figure 5.8: Flowcharts for Node 3.

Chapter 6

Experimental results and analysis

This Chapter presents the results of the tests made to test hardware and software. Firstly, range of LoRa modulation technology was tested. And, after that, the results of the system working during one complete day are taken to test the dashboard and the resistance and stability of the system. Other tests for evaluate the data given by sensors were performed to evaluate if the sensors are suitable for an air quality monitoring system.

6.1 Range tests of LoRa

In Section 2.2.1.5, LoRa was addressed as being a technology for long distance communications. This modulation technique is sensitive to buildings, achieving better results in line-of-sight.

A range test was carried out between the LoRa devices on the FEUP campus. Node 3 (which reroutes all the data to the server) was placed near a window of the far north-east building. Node 1 was placed in a laboratory of the nearest building (a route with many obstacles). And node 2 had a mobile position. The map in the Figure 6.1 present the locations where tests were performed and its results are indicated. Node 3 is marked with purple. The tests resulted in successful communication are marked with green. The tests where the communications were only established with the window open are marked with yellow. And, tests with no communication established are marked with red. Table 6.1 present the distance between Node 3 (stationary) and the other nodes (Node 1 is stationary and Node 2 mobile), the building density (qualitative value), and the result of each test.

The points marked with “2d”, “2e” and “2f” in the map are closer to Node 3 than the point “2g” but present worse results. This fact provides the conclusion that building density is a determining factor in the success of LoRa communication. Unfortunately, it was not possible to conclude about the communication range of the nodes. It would be necessary to test longer distances for that.

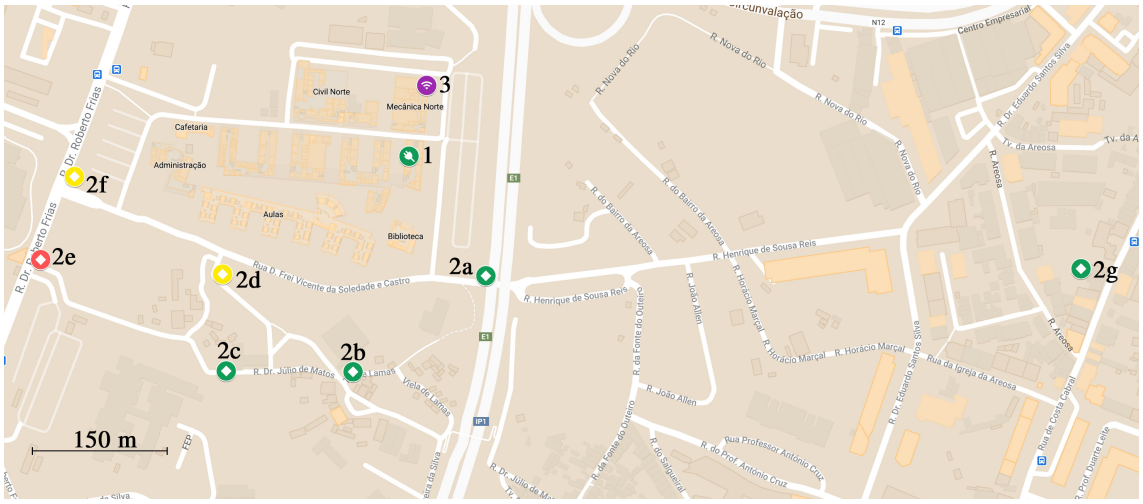


Figure 6.1: Map of the locations used to test the LoRa range

Table 6.1: Locations used to test the LoRa range.

Node and reference in the map (Fig. 6.1) and local	**Distance to node 3**	**Building density**	**Result**
1, Laboratory L003	81 m	Very high	Successful connection with the window closed
2a, *Parque dos Alunos*	222 m	Zero	Successful connection with the window closed
2b, Canteen	330 m	Low	Successful connection with the window closed
2c, *Associação de Estudantes*	389 m	Medium	Successful connection with the window closed
2d, *Lamas*	310 m	High	Successful connection with the window open
2e, *Parque de Lamas*	473 m	Very high	Connection failed
2f, *Parque Pessoal Permanente*	406 m	High	Successful connection with the window open
2g, *Rua de Costa Cabral*	764 m	Low	Successful connection with the window closed

6.2 Installation of the system

To get 24 hours of results, the monitoring system was mounted in a residential area. The Node 1 was placed in a extreme of the area, outdoor. The Node 2 was placed in the another extreme, outdoor. And, the Node 3

was placed in a kitchen (spatially between Node 1 and Node 2). The distance between nodes was no longer than 50 meters.

6.3 Visualisation of the results in the dashboard

As in last Chapter, the dashboard was built using Ubidots platform and it is presented in Figure 6.2 displaying data for the last 24 hours. This dashboard includes data from all the nodes, grouped and distinguished by colours of the labels. Relative humidity, a measurement taken by the three nodes, is indicated by a gauge. For Total VOC, the gauge indicates the last value of it concentration, and the corresponding colour is based on the legal values for the gas, Section 2.1.3, but it is relevant to note that the thresholds are created for a 8-hour mean and not for an isolated value. Temperature is presented by a specific widget designed exclusively for that. Gas concentrations are exhibited by charts (1 hour-mean values) and by tables (maximum values for a certain interval and the last values). Also, header and footer was placed in the dashboard to present title of the project, weather forecast (using HTML code) and the entities involved in this project.

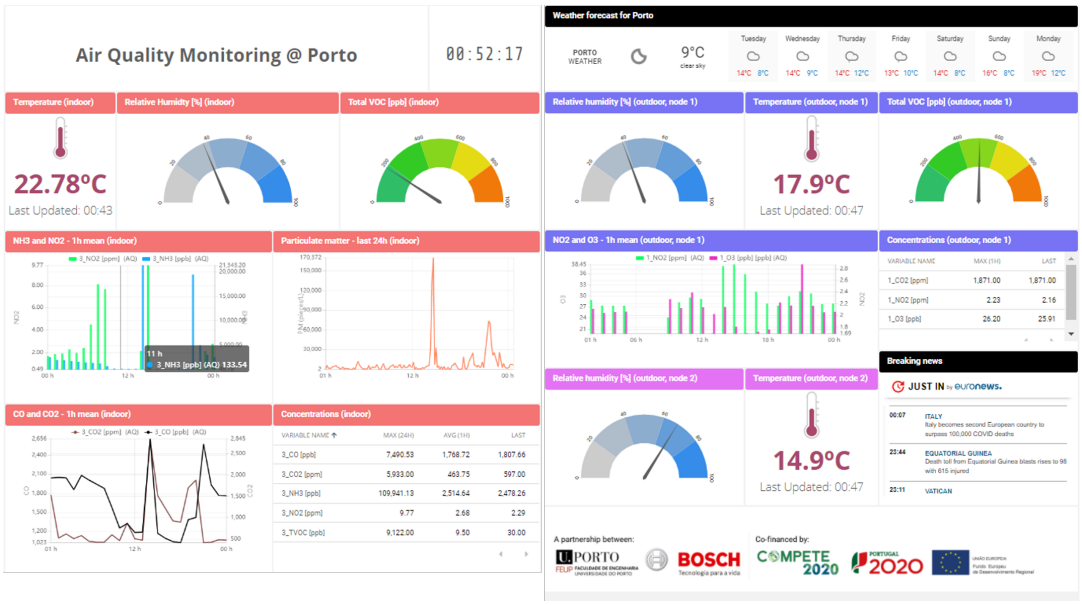


Figure 6.2: Screenshot of the dashboard displaying results for the last 24 hours.

6.4 Analysis of the results

Some notes could be taken from the results presented above. The Figure 6.3 present some areas analysed here.

- Area 1 – There are any log of NO_2 concentration for 10 hours for the Node 1. In software part, the range of values was controlled and the values above the maximum expected are denied. This indicates an error or damage in I_2C communication of the sensor, only in the NO2 channel of MIKROE-3056 (Air Quality 5 click).
- Area 2 and 3 – There are two peaks in particulate matter in meal times (lunch and dinner), corresponding to the kitchen usage.

- Area 4 – Nodes 1 and 2 did not communicate with Node 3 for three hours, although it continued to send its data to the server. After an forced manual restart of the Node 3, this node again received data from the other nodes. Here, LoRa module failed but the software did not detect that. A solution in software of the Node 3 can be implemented to detect if the LoRa module receives any data packet in the last 20 minutes and, if anything is received, the Node 3's MCU restarts.
- Area 5 – The O_3 sensor was the only that had possibility of calibration. A comparison of the O_3 data of the sensor with data from the *Agência Portuguesa do Ambiente* station[1], Figure 6.4, allow to compare that the same upward and downward trend in O_3 levels was visible at the two collecting points. And, the higher point of the day is at the same hour of the day and the value similar, 38.45 ppb in our station and 34.13 ppb (= 67 µg/m^3) in the governmental station.

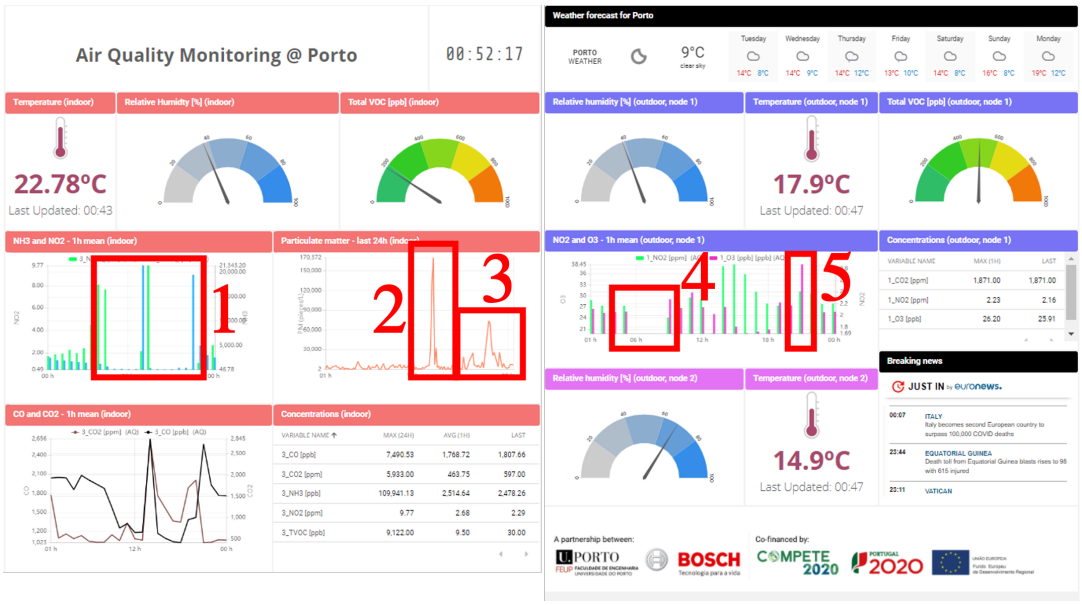


Figure 6.3: Screenshot of the dashboard displaying results for the last 24 hours annotated.

6.5 Alert triggering

The event programmed in Section 5.7 was triggered, sending an SMS according to the imposed configurations. The result is presented in Figure 6.5

[1] Agência Portuguesa do Ambiente. QualAR - Qualidade do AR. URL: https://qualar.apambiente.pt/indices.

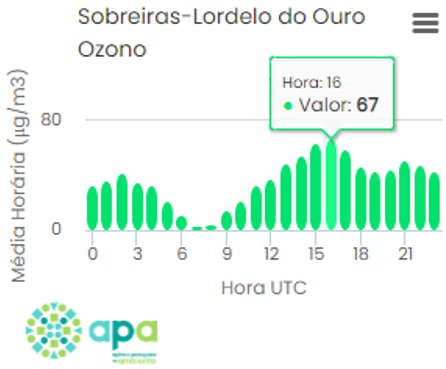


Figure 6.4: Data of O_3 hourly concentrations in the Sobreiras-Lordelo do Ouro (in Porto) air quality monitoring station *Agência Portuguesa do Ambiente*[1].

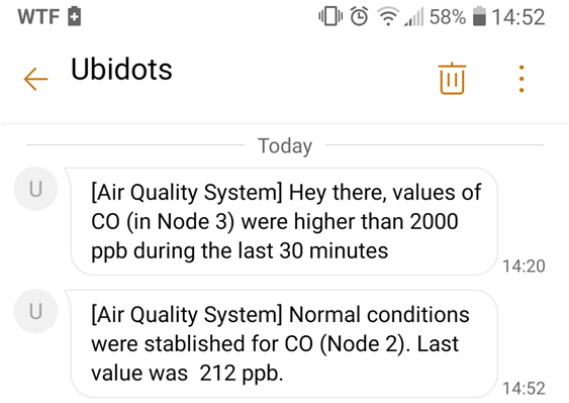


Figure 6.5: Screenshot of the SMS sent by Ubidots, when the programmed event triggered.

6.6 Other tests

6.6.1 Effect of direct sunlight exposure

Figure 6.6 shows the effect of direct sunlight exposure to the measured temperature. When the BME680 sensor, installed in Node 2, was exposed directly to sunlight, the values of temperature are much higher (plus 15 ºC) than when protected by a shadow. This happens because of the heating of PCB that supports BME680, caused by solar radiation. This test allow to take the conclusion that in an ideal system, the sensors should be physically protected from the solar radiation. Moreover, this indicates that this sensor is not be usable in outdoor conditions without a mechanical protection.

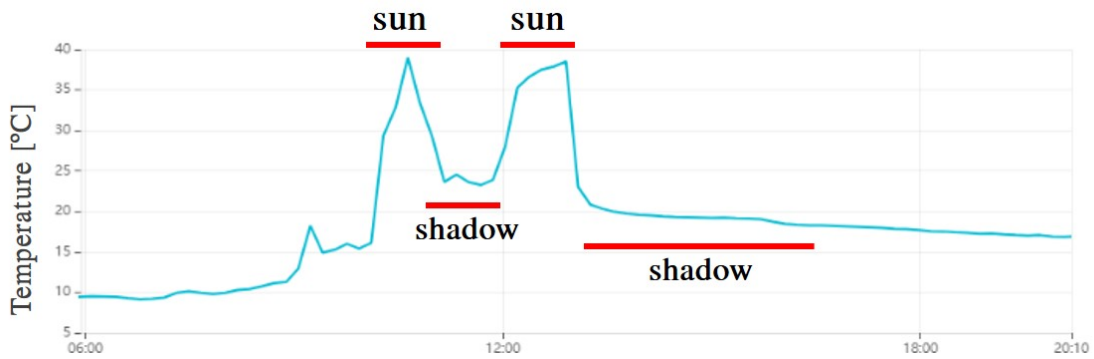


Figure 6.6: Effect of sunlight exposure in temperature measurements.

6.6.2 Comparison of temperature sensors

The three nodes contain a sensor to read the temperature, BME280 for Node 1 and 3, and BME680 for Node 2. To compare the temperature results between them, a test was performed. In this test, all the sensors are placed side by side with a distance between them no longer than 20 cm, i.e. all of them in very similar test conditions. The test lasted for 24 hours straight. The configuration of the sensors was reorganised several times during the test. The results are presented in Figure 6.7.

Area marked with "1" represents a configuration where the sensors are placed as in the Figures 4.14b, 4.15b and 4.16b. This configuration led to a significant difference in the value of temperature measured by the three sensors.

A suspicion that the temperature read by the BME280 sensor of node 3 was influenced by the heating of the MICS-6814 sensor (MIKROE-3056 module, Air Quality 5 click), the BME280 sensor was repositioned to place it outside of the area of heat influence of the MICS-6814 sensor. The area marked with "2" represents this configuration. For this part of the test, the difference between the values read by both BME280 sensors is very small. But there is a significant difference between these values (2 ºC) and the values from the sensor BME680. This alignment of the values of the sensors indicates that the placement of the sensors must take into consideration nearby heat sources.

The area marked with "3" represents the last part of the test, where the three temperature sensors were placed in the same breadboard with a distance between them no longer than 5 cm, all of them far from any heat source. In this test, the same difference of temperature values (2 ºC) between BME280 sensors and BME680 sensor is present. At the instant marked with "Pt100", the temperature was measured by a Pt100 resistance temperature detector, a high accuracy sensor for temperature. The Pt100 indicates a temperature of 21.64 ºC (resistance = 108.43 Ω), the BME280 of Node 1 indicates 24.10 ºC, the BME680 of Node 2 indicates 21.93 ºC, and the BME280 sensor of Node 3 indicates 24.16 ºC. Assuming the Pt100 value as the reference value for temperature, the BME680 sensor presents a much higher accuracy than the BME280 sensors. For this reason, when the choice is possible, the BME680 sensor should be preferred over the BME280 sensor. The Figure 6.8 presents this configuration with the Pt100 equipment.

6.7 Summary

LoRa modulation technology presents a excellent range but high sensitivity to built-up areas. During the tests, problems in the LoRa module were found, but corrected. A complete 24 hour sample was possible to

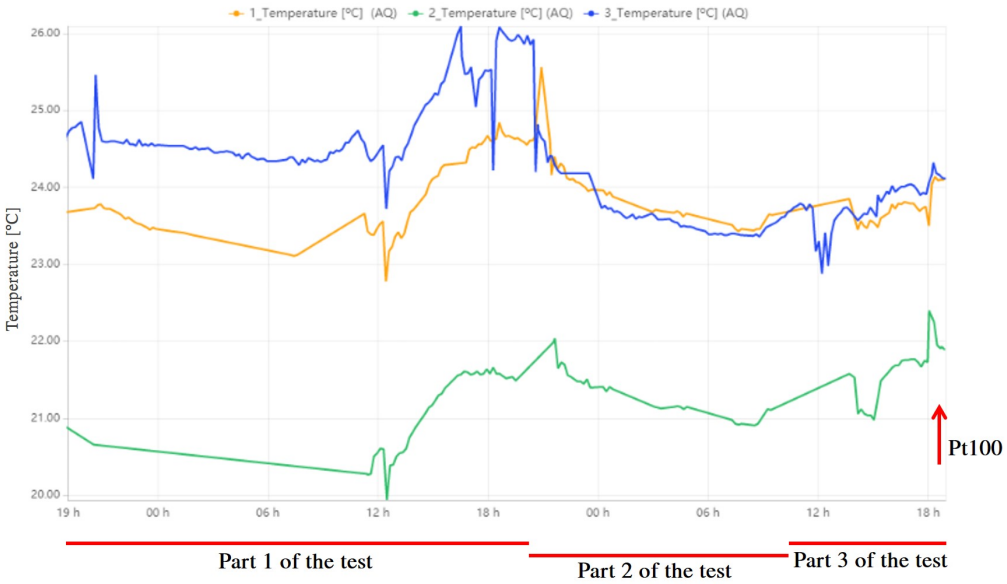


Figure 6.7: Comparison of temperature readings of the sensors.

test the dashboard. The tests verified the stability and usability of the projected system and dashboard. A relationship was found between the data obtained by the sensors and the surrounding events. Furthermore, it was possible to compare the O_3 sensor data with calibrated values, finding a good relation between them. Other tests proved that the sensors should be protected from direct exposure to the sunlight because the measurements of temperature are influenced by the solar radiation. The temperature sensors must be placed away from heat sources to avoid influence of other sensors in the temperature measurements. Moreover, the temperature sensors used in this work presented different accuracies in the tests performed.

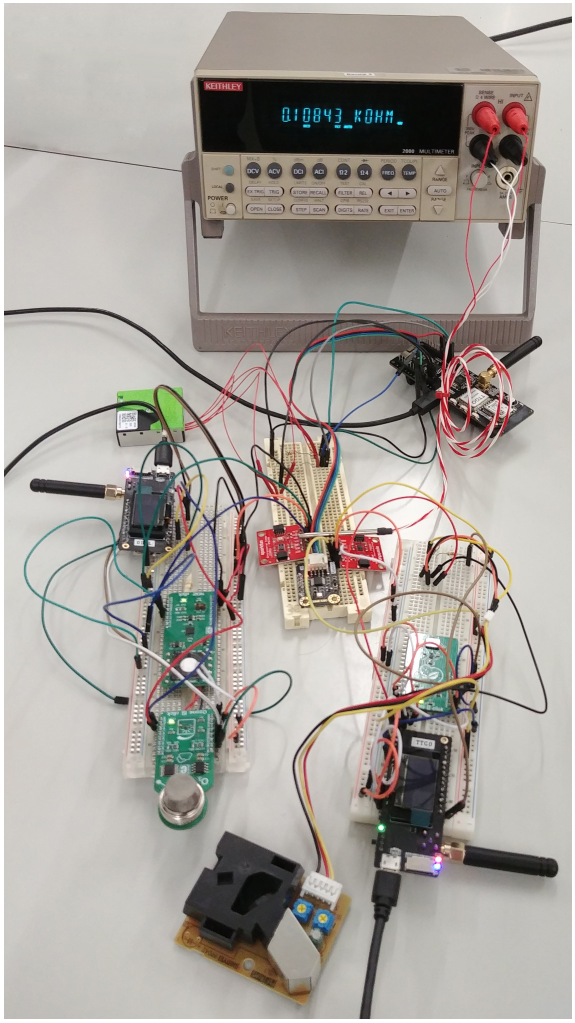


Figure 6.8: Configuration of the test for the Pt100 temperature sensor.

Chapter 7

Conclusions and Future Work

This Chapter presents the conclusions of this work and suggestions for futures works, based on what could be done based in this work.

7.1 Conclusions

The aim of this project was to develop an air quality system to assess indoor and outdoor air quality in urban environments. Firstly, the air quality was studied to understand what parameters reflect air quality. For ambient (outdoor), the system should measure at least the concentration of PM_{10}, $PM_{2.5}$, NO_2 and O_3, and environmental factors, such as temperature and relative humidity. Indoor air quality is assessed by measuring concentrations of PM_{10}, $PM_{2.5}$, CO, CO_2, NO_2 and Total VOC and measuring temperature and relative humidity. After that, wireless communication protocols for Smart Cities were studied, where cellular and LoRa/LoRaWAN technology proved to be the most promising for long distance communications.

The built prototype included three nodes, two for outdoor and one for indoor air quality assessment, collecting different parameters. To store and manage data, the system was designed to communicate among them, sending the sensor data to a gateway node that sends data to a server. The city context implies long distances between collection points, and low power consumption of devices for low maintenance. LoRa modulation meets these requirements and was adopted as communication standard used among the system. For data storage and management, the MQTT protocol (over TCP/IP) was used by the gateway node to send all the data to the MQTT broker, that stores and display the data. A photovoltaic cell was projected to ensure that sunlight allows energy self-sufficiency. And, a plastic protection case was designed to make the system resistant to weather events, such as rain and wind gusts.

The results demonstrated the stability of the projected system. The measurements of temperature, using BME280 and BME680 sensors, revealed that they present different temperature values for the same test conditions, being the BME680 more accurate. Furthermore, they are sensitive to the exposure to solar radiation, requiring a mechanical protection to allow them to be used outdoor. The O_3 sensor present similarity between collected values and the values given by governmental stations for the tests performed. For other sensors, calibration and data validation were not possible due to the lack of precise and accurate instrumentation. A good relationship between the sensor data used and the data from this measurement instrumentation would indicate that the sensors are suitable for urban environments.

7.2 Further Work

As further work, it would be interesting to perform the following tasks:

- Perform tests in a controlled environment to calibrate correctly the sensors;
- Compare the results of all the low-cost sensors used in this project to calibrated and professional instrumentation measurements. This allow to assess the precision of the sensors, giving a good indicator of the quality of the sensor measurements. A good relation between the values measured by these sensors and the values given by professional instrumentation will help to decide what sensors could be used in a comprehensive air quality monitoring system;
- Perform tests in city context, which is the aim of the product of this work. This task would allow the system to level up from "Science education" to "community based monitoring" category (Figure 2.7), and to test the scalability of the system;
- Long-term test of the PV cell + battery solution projected in Section 4.3;
- Test another LoRa topologies, such as a LoRaWAN architecture. This option allow much more security and stability in LoRa communications. However, a custom mesh LoRa network could be interesting for wide areas where could be not possible to communicate to a central gateway due to building density and long distances;
- Create variables in the dashboard that communicate air quality in a simpler way, indicating air quality with terms as "Good", "Medium" or "Bad";
- Apply an algorithm to process data and eliminate the non-sense values;
- Apply an algorithm for air quality forecast, making use of machine learning algorithms.

References

[1] World Health Organization. Ambient air pollution: A global assessment of exposure and burden of disease. Technical report, World Health Organization, Geneva, Switzerland, 2016. URL: https://www.who.int/phe/publications/air-pollution-global-assessment/en/.

[2] "Right to Clean Air" (LIFE15 GIE/DE/000795). Legal Actions on Clean Air - Summary 2016 – 2019. Technical report, Deutsche Umwelthilfe e.V., 2020. URL: https://www.right-to-clean-air.eu/fileadmin/Redaktion/Downloads/Laymans_report_ENG_Right_to_clean_Air.pdf.

[3] World Health Organization. Regional Office for Europe. The Right to Healthy Indoor Air. Technical report, Bilthoven, Netherlands, 2000. URL: https://www.euro.who.int/__data/assets/pdf_file/0019/117316/E69828.pdf.

[4] World Health Organization. Burden of disease from the joint effects of household and ambient Air pollution for 2016. Technical report, World Health Organization, Geneva, Switzerland, 2018. URL: https://www.who.int/airpollution/data/AP_joint_effect_BoD_results_May2018.pdf.

[5] Health Effects Institute. State of Global Air 2020. Technical report, Health Effects Institute, Boston, Massachusetts, United States of America, 2020. URL: https://www.stateofglobalair.org/sites/default/files/documents/2020-10/soga-2020-report.pdf.

[6] International Agency for Research on Cancer. IARC: Outdoor air pollution a leading environmental cause of cancer deaths. Technical report, WHO's International Agency for Research on Cancer, Lyon, France; Geneva, Switzerland, 2013.

[7] World Health Organization. Regional Office for Europe. Air quality guidelines for Europe, 1987. URL: https://apps.who.int/iris/handle/10665/107364.

[8] World Health Organization. Regional Office for Europe. Updating and revision of the air quality guidelines for Europe : report on a WHO Working Group on Volatile Organic Compounds. Technical report, WHO, Brussels, Belgium, oct 1995. URL: http://whqlibdoc.who.int/euro/1994-97/EUR_ICP_EHAZ_94_05_MT_12.pdf.

[9] Regional Office for Europe of the World Health Organization. Air Quality Guidelines for Europe: Second Edition. Technical report, World Health Organization, Copenhagen, Denmark, 2000. URL: euro.who.int/__data/assets/pdf_file/0005/74732/E71922.pdf.

[10] World Health Organization. Regional Office for Europe. Air Quality Guidelines: Global Update 2005. Technical report, 2006. URL: www.euro.who.int.

[11] Wayne R. Ott. Concepts of human exposure to air pollution. *Environ. Int.*, 7(3):179–196, jan 1982. doi:10.1016/0160-4120(82)90104-0.

[12] Philip K. Hopke, Qili Dai, Linxuan Li, and Yinchang Feng. Global review of recent source apportionments for airborne particulate matter, oct 2020. doi:10.1016/j.scitotenv.2020.140091.

[13] Rajen N. Naidoo. NO 2 increases the risk for childhood asthma: a global concern, apr 2019. doi:10.1016/S2542-5196(19)30059-2.

[14] Dongni Hou, Yihui Ge, Cuicui Chen, Qiang Tan, Renjie Chen, Yanjie Yang, Li Li, Jian Wang, Maosong Ye, Chun Li, Xia Meng, Haidong Kan, Jing Cai, and Yuanlin Song. Associations of long-term exposure to ambient fine particulate matter and nitrogen dioxide with lung function: A cross-sectional study in China. *Environ. Int.*, 144:105977, nov 2020. doi:10.1016/j.envint.2020.105977.

[15] Yaron Ogen. Assessing nitrogen dioxide (NO2) levels as a contributing factor to coronavirus (COVID-19) fatality. *Sci. Total Environ.*, 726:138605, jul 2020. doi:10.1016/j.scitotenv.2020.138605.

[16] Endwoke Amsalu, Yuming Guo, Haibin Li, Tianqi Wang, Yue Liu, Anxin Wang, Xiangtong Liu, Lixin Tao, Yanxia Luo, Feng Zhang, Xinghua Yang, Xia Li, Wei Wang, and Xiuhua Guo. Short-term effect of ambient sulfur dioxide (SO2) on cause-specific cardiovascular hospital admission in Beijing, China: A time series study. *Atmos. Environ.*, 208:74–81, jul 2019. doi:10.1016/j.atmosenv.2019.03.015.

[17] Neil E. Klepeis, William C. Nelson, Wayne R. Ott, John P. Robinson, Andy M. Tsang, Paul Switzer, Joseph V. Behar, Stephen C. Hern, and William H. Engelmann. The National Human Activity Pattern Survey (NHAPS): A resource for assessing exposure to environmental pollutants. *J. Expo. Anal. Environ. Epidemiol.*, 11(3):231–252, jul 2001. URL: www.nature.com/jea, doi:10.1038/sj.jea.7500165.

[18] WHO Regional Office for Europe. WHO guidelines for indoor air quality: dampness and mould. Technical report, 2009. URL: http://www.who.int/airpollution/guidelines/dampness-mould/en/.

[19] World Health Organization. Regional Office for Europe. WHO guidelines for indoor air quality: selected pollutants. Technical report, World Health Organization, Copenhagen, Denmark, 2010. URL: www.euro.who.int.

[20] World Health Organization. Indoor Air Quality Guidelines: Household Fuel Combustion. Technical report, World Health Organization, Geneva, Switzerland, 2014. URL: http://apps.who.int/iris/handle/10665/144309/.

[21] Scientific Committee on Health and Environmental Risks. Opinion on risk assessment on indoor air quality. Technical report, European Commission, 2007. URL: http://ec.europa.eu/health/ph_risk/risk_en.htm.

[22] Igor Mujan, Aleksandar S. Anđelković, Vladimir Munćan, Miroslav Kljajić, and Dragan Ružić. Influence of indoor environmental quality on human health and productivity - A review. *J. Clean. Prod.*, 217:646–657, apr 2019. doi:10.1016/j.jclepro.2019.01.307.

[23] Xin Shan, Anastasia Nissa Melina, and En-Hua Yang. Impact of indoor environmental quality on students' wellbeing and performance in educational building through life cycle costing perspective. *J. Clean. Prod.*, 204:298–309, dec 2018. doi:10.1016/j.jclepro.2018.09.002.

[24] Monika Ścibor, Bartosz Balcerzak, Andrzej Galbarczyk, Nikodem Targosz, and Grazyna Jasienska. Are we safe inside? Indoor air quality in relation to outdoor concentration of PM10 and PM2.5 and to characteristics of homes. *Sustain. Cities Soc.*, 48(April):101537, jul 2019. doi:10.1016/j.scs.2019.101537.

[25] S. Tham, R. Thompson, O. Landeg, K.A. Murray, and T. Waite. Indoor temperature and health: a global systematic review. *Public Health*, 179:9–17, feb 2020. doi:10.1016/j.puhe.2019.09.005.

[26] Gaetano Settimo, Maurizio Manigrasso, and Pasquale Avino. Indoor Air Quality: A Focus on the European Legislation and State-of-the-Art Research in Italy. *Atmosphere (Basel).*, 11(4):370, apr 2020. doi:10.3390/atmos11040370.

[27] Agência Portuguesa do Ambiente and Direção-Geral da Saúde. Metodologia de avaliação da qualidade do ar interior em edifícios de comércio e serviços no âmbito da Portaria 353-A/2013. Technical report, 2015.

[28] World Health Organization. WHO handbook on indoor radon: a public health perspective. Technical report, Geneva, Switzerland, 2009. URL: https://www.who.int/ionizing_radiation/env/9789241547673/en/.

[29] Amit Sharma, Pradeep Kumar Singh, and Yugal Kumar. An integrated fire detection system using IoT and image processing technique for smart cities. *Sustain. Cities Soc.*, 61:102332, oct 2020. doi:10.1016/j.scs.2020.102332.

[30] Liesbet van Zoonen. Privacy concerns in smart cities. *Gov. Inf. Q.*, 33(3):472–480, jul 2016. doi:10.1016/j.giq.2016.06.004.

[31] Brian Ray. Examining Cellular IoT: Cost, Battery, & Data, aug 2015. URL: https://www.link-labs.com/blog/cellular-iot.

[32] Market Intelligence & Consulting Institute (MIC). Global Wi-Fi Enabled Devices Shipment Forecast, 2020-2024. Technical report, Market Intelligence & Consulting Institute (MIC), jul 2020. URL: https://www.researchandmarkets.com/reports/5135535/global-wi-fi-enabled-devices-shipment-forecast#rela0-4826074.

[33] Nur Aziemah Azmi Ali and Nurul Adilah Abdul Latiff. Environmental Monitoring System Based on LoRa Technology in Island. *Proc. - 2019 IEEE Int. Conf. Signals Syst. ICSigSys 2019*, pages 160–166, 2019. doi:10.1109/ICSIGSYS.2019.8811066.

[34] Yansong Wang, Yi Huang, and Chaoyun Song. A New Smart Sensing System Using LoRaWAN for Environmental Monitoring. In *2019 Comput. Commun. IoT Appl.*, pages 347–351. IEEE, oct 2019. doi:10.1109/ComComAp46287.2019.9018829.

[35] Bernat Carbones Fargas and Martin Nordal Petersen. GPS-free geolocation using LoRa in low-power WANs. In *GIoTS 2017 - Glob. Internet Things Summit, Proc.* Institute of Electrical and Electronics Engineers Inc., aug 2017. doi:10.1109/GIOTS.2017.8016251.

[36] Yasir Mehmood, Farhan Ahmad, Ibrar Yaqoob, Asma Adnane, Muhammad Imran, and Sghaier Guizani. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Commun. Mag.*, 55(9):16–24, 2017. doi:10.1109/MCOM.2017.1600514.

[37] Michele Penza. Low-cost sensors for outdoor air quality monitoring. In *Adv. Nanomater. Inexpensive Gas Microsens.*, pages 235–288. Elsevier, jan 2020. doi:10.1016/b978-0-12-814827-3.00012-8.

[38] Daniel Schürholz, Sylvain Kubler, and Arkady Zaslavsky. Artificial intelligence-enabled context-aware air quality prediction for smart cities. *J. Clean. Prod.*, 271:121941, oct 2020. doi:10.1016/j.jclepro.2020.121941.

[39] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. AirCloud : A Cloud-based Air-Quality Monitoring System for Everyone. In *SenSys '14 Proc. 12th ACM Conf. Embed. Netw. Sens. Syst.*, pages 251–265, Memphis, Tennessee, USA, nov 2014. Association for Computing Machinery. doi:10.1145/2668332.2668346.

[40] Zeba Idrees and Lirong Zheng. Low cost air pollution monitoring systems: A review of protocols and enabling technologies. *J. Ind. Inf. Integr.*, 17(April 2019):100123, mar 2020. doi:10.1016/j.jii.2019.100123.

[41] Nuno Canha, Ana Carolina Alves, Catarina Simão Marta, Joana Lage, Joana Belo, Tiago Faria, Sandra Cabo Verde, Carla Viegas, Célia Alves, and Susana Marta Almeida. Compliance of indoor air quality during sleep with legislation and guidelines – A case study of Lisbon dwellings. *Environ. Pollut.*, 264:114619, sep 2020. doi:10.1016/j.envpol.2020.114619.

[42] Priyanka DeSouza, Amin Anjomshoaa, Fabio Duarte, Ralph Kahn, Prashant Kumar, and Carlo Ratti. Air quality monitoring using mobile low-cost sensors mounted on trash-trucks: Methods development and lessons learned. *Sustain. Cities Soc.*, 60(September 2019):102239, sep 2020. doi:10.1016/j.scs.2020.102239.

[43] Sachit Mahajan and Prashant Kumar. Evaluation of low-cost sensors for quantitative personal exposure monitoring. *Sustain. Cities Soc.*, 57(November 2019):102076, jun 2020. doi:10.1016/j.scs.2020.102076.

[44] Thang Viet Tran, Nam Trung Dang, and Wan-young Chung. Battery-free smart-sensor system for real-time indoor air quality monitoring. *Sensors Actuators B Chem.*, 248:930–939, sep 2017. doi:10.1016/j.snb.2017.01.198.

[45] Mohieddine Benammar, Abderrazak Abdaoui, Sabbir H M Ahmad, Farid Touati, and Abdullah Kadri. A Modular IoT Platform for Real-Time Indoor Air Quality Monitoring. *Sensors (Basel).*, 18(2):581, feb 2018. doi:10.3390/s18020581.

[46] Huang-chen Lee and Kai-hsiang Ke. Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation. *IEEE Trans. Instrum. Meas.*, 67(9):2177–2187, sep 2018. doi:10.1109/TIM.2018.2814082.

[47] Jeferson Rodrigues Cotrim and João Henrique Kleinschmidt. LoRaWAN Mesh Networks: A Review and Classification of Multihop Communication. *Sensors*, 20(15):4273, jul 2020. doi:10.3390/s20154273.

[48] AllTheResearch. Global LoRaWAN Market- Segment Analysis, Opportunity Assessment, Competitive Intelligence, Industry Outlook 2016-2027 | Inforgrowth - Market Research. Technical report, AllTheResearch, 2019. URL: https://inforgrowth.com/report/4946697/lorawan-market.

Appendix A

Programming code

A.1 C++ code uploaded in the MCU of the Node 1

Table A.1 presents the list of files used in programming code of Node 1.

Table A.1: Organisation of programming code files for the Node 1

Function	Description
`Node_1.ino`	Main file
`MIKROE_2767.h`	Header file of MIKROE-2767 code. Includes function and class declarations
`MIKROE_2767.cpp`	Implementation file of MIKROE-2767's class. This file contains the reading routines of ozone
`MIKROE_3098.h`	Header file of MIKROE-3098 code. Includes function and class declarations
`MIKROE_3098.cpp`	Implementation file of MIKROE-3098's class. This file contains the reading routines of NO_2
`Sparkfun_CCS811_BME280.h`	Header file of Sparkfun SEN-14348 code. Includes function and class declarations
`Sparkfun_CCS811_BME280.cpp`	Implementation file of Sparkfun SEN-14348's class. This file contains the readings of the BME280 and CCS811
`LoRa_Sender.h`	Header file of LoRa code. Includes function and class declarations
`LoRa_Sender.cpp`	Implementation file of Lora's class. This file contains LoRa communication routines

A.1.1 `Node1.ino`

```
/*
 * Node 3 - Sensors:   - Ozone 2 click - O3
 *                     - Sparkfun combo CCS811+BME280 - T, RH, TVOC, CO2
 *                     - NO2 click - NO2
 * Created by João Paulo Sousa, FEUP, 26th January 2021
 */

#include "MIKROE_2767.h"
#include "MIKROE_3098.h"
#include "Sparkfun_CCS811_BME280.h"
#include "Lora_Sender.h"

```

```
#include <Arduino.h>
#include "stdint.h"
#include <Wire.h>

// Declare objects and variables
MIKROE_2767 mikroe_2767; //Ozone 2 click
MIKROE_3098 mikroe_3098; // NO2 click
Sparkfun_CCS811_BME280 sparkfun_combo; //Sparkfun CCS7811+BME280
LoRa_Sender lora_sender;
SpFun_COMBO_DATA_struct sensorData;
float NO2_ppm;
float O3_ppb;
int counter = 0;

void setup() {
  Serial.begin(115200);           // start serial for output
  Wire.begin();                 // join i2c bus
  Serial.println("--- System init ---");
  delay(10);
  sparkfun_combo.firstRun();
  mikroe_2767.firstRun();
  mikroe_3098.firstRun();
  lora_sender.firstRun();
  delay(3000);
  sensorData = sparkfun_combo.get_SparkfunComboData();
  ↪ //SpFun_COMBO_DATA_struct. Contains CO2, T, RH, TVOC, Pressure
  delay(1000);
}

// Get data from sensors
void getData(){
  NO2_ppm = mikroe_3098.getNO2_ppm();
  sensorData = sparkfun_combo.get_SparkfunComboData();
  ↪ //SpFun_COMBO_DATA_struct. Contains CO2, T, RH, TVOC, Pressure
  O3_ppb = mikroe_2767.getO3_ppb();
}

// Show data in Serial Monitor
void showDataSerial(){
  Serial.println("CO2: " + String(sensorData.CO2_ppm) + " ppm");
  Serial.println("TVOC: " + String(sensorData.TVOC_ppb) + " ppb");
  Serial.println("Temperature: " + String(sensorData.Temp_C, 2) + "
  ↪ °C");
  Serial.println("Pressure: " + String(sensorData.Pressure_Pa/1000, 2)
  ↪ + " kPa");
```

```
54    Serial.println("RH: " + String(sensorData.RH_perc, 2) + " %");
55    Serial.println("NO2: " + String(NO2_ppm*1000, 2) + " ppb");
56    Serial.println("O3: " + String(Serial.print(O3_ppb)) + " ppb");
57  }
58
59  void loop() {
60    counter++;
61    Serial.println();
62    Serial.println(" >>>>> COLLECTING DATA (NODE 1): SAMPLE #" +
      ↪  String(counter));
63    getData();
64    if (counter > 2) // when counter = 1, data are not fine. And, we want
      ↪  more time for some stability in data
65    {
66      char json[250];
67      sprintf(json, "{\"1t\":%02.02f, \"1h\":%02.02f, \"1p\":%02.02f,
        ↪  \"1co2\":%02.02f, \"1tvoc\":%02.02f, \"1no2\":%06.06f,
        ↪  \"1o3\":%02.02f}", sensorData.Temp_C, sensorData.RH_perc,
        ↪  sensorData.Pressure_Pa, sensorData.CO2_ppm,
        ↪  sensorData.TVOC_ppb, NO2_ppm, O3_ppb);
68      showDataSerial();
69      lora_sender.sendMessage(json,25000,600e3);
70    }
71    delay(1500);
72  }
```

A.1.2 MIKROE_2767.h

```
1  /*
2   * MIKROE_2767.h - Ozone 2 click
3   * Created by João Paulo Sousa, FEUP, 28th January 2021
4  */
5  #include <Arduino.h>
6  #include "stdint.h"
7
8  #ifndef MIKROE_2767_h
9  #define MIKROE_2767_h
10
11 class MIKROE_2767
12 {
13   public:
14     void firstRun( void );
```

```
15      float getO3_ppm(float humidityPercent = 60, float
        ↪  temperatureCelsius = 20, float valueRL = 100, float valueR0 =
        ↪  7.1341);
16      float getO3_ppb(float humidityPercent = 60, float
        ↪  temperatureCelsius = 20, float valueRL = 100, float valueR0 =
        ↪  7.1341);
17    private:
18      uint32_t readADC_Data( );
19      float readRs (float valueRL);
20      float getEnvCorrectRatio(float humidityPercent, float
        ↪  temperatureCelsius);
21  };
22  #endif
```

A.1.3 MIKROE_2767.cpp

```
1   /*
2    * MIKROE_2767.cpp - Ozone 2 click
3    * Created by João Paulo Sousa, FEUP, 28th January 2021
4   */
5
6   #include "MIKROE_2767.h"
7   #include "HardwareSerial.h"
8   #include <Wire.h>
9   #include <SPI.h>
10
11  // SPI pins
12  const byte pinSCK = 14;
13  const byte pinMISO = 2; // =SDO
14  const byte pinMOSI = 15;
15  const byte pinCS = 13; // =SS
16
17  SPIClass hspi(HSPI); //we need to create a spi object because we will
    ↪  use two spi channels (for LoRa and for MIKROE-2767 (ozone 2 click)
18
19  // Function to run in setup(), sets SPI configurations for the sensor
    ↪  module
20  void MIKROE_2767::firstRun( void )
21  {
22      pinMode( pinMISO, INPUT );
23      pinMode( pinCS, OUTPUT );
24      digitalWrite( pinCS, HIGH );
25      hspi.begin(pinSCK, pinMISO, pinMOSI, pinCS); // start the SPI
26      hspi.setBitOrder( MSBFIRST );
```

```
27      hspi.setDataMode( SPI_MODE0 );
28      hspi.setClockDivider( SPI_CLOCK_DIV16 );
29  }
30
31  // Get sensor data
32  uint32_t MIKROE_2767::readADC_Data( ){
33    uint8_t ADC_byte[3];
34    uint32_t ADC_data;
35    digitalWrite( pinCS, LOW ); // set CS low = turn on SPI data
       ↪  communication
36    delay(50);
37    while(digitalRead(pinMISO));
38    ADC_byte[0] = hspi.transfer(0x00);
39    ADC_byte[1] = hspi.transfer(0x00);
40    ADC_byte[2] = hspi.transfer(0x00);
41    ADC_data = 0;
42    ADC_data = ADC_byte[0];
43    ADC_data <<= 8;
44    ADC_data |= ADC_byte[1];
45    ADC_data <<= 8;
46    ADC_data |= ADC_byte[2];
47    digitalWrite( pinCS, HIGH ); //set CS high = turn off SPI data
       ↪  communication
48    if ( ADC_data < 2097151 && ADC_data> 0) //2^21-1
49        return ADC_data;
50    else return 0;
51    delay(50);
52  }
53
54
55  // Convert ADC_data from SPI communicatin into resistance of the sensor
56  float MIKROE_2767::readRs (float valueRL){
57    float vRL = ((float)readADC_Data())/2097152.0*5.0; // Voltage on load
       ↪  resistance. 21^2-1=2097152
58    if(!vRL) return 0.0f; // division by zero prevention
59    float rS = (5.0/vRL - 1.0) * valueRL; // resistance of the sensor. RL
       ↪  is the variable resistance on the sensor+100!
60    return rS;
61  }
62
63  //Get correction to apply on Rs depending on environmental
64  float MIKROE_2767::getEnvCorrectRatio(float humidityPercent, float
     ↪  temperatureCelsius) {
65     // Select the right equation based on humidity. If default value,
        ↪  ignore correction ratio
```

```
66    if(humidityPercent == 60 && temperatureCelsius == 20) {
67      return 1.0;
68    }
69    else if(humidityPercent > 75) {  // For humidity > 75%, use the 85%
       ↪  curve
70      return -0.0141 * temperatureCelsius + 1.5623;// R^2 = 0.9986
71    }
72    else if(humidityPercent > 50) { // For humidity > 50%, use the 60%
       ↪  curve
73      return -0.0119 * temperatureCelsius + 1.3261; // R^2 = 0.9976
74    }
75    return -0.0103 * temperatureCelsius + 1.1507; // Humidity < 50%, use
       ↪  the 30% curve. R^2 = 0.996
76  }
77
78  // Returns O3 in ppm. R0 result from calibration. RL is 100 ohm +
     ↪  resistance in potentiometer
79  float MIKROE_2767::getO3_ppm(float humidityPercent, float
     ↪  temperatureCelsius, float valueRL, float valueR0){
80    float valueRs = readRs(valueRL);
81    if (valueRs < 0) {
82      return 0.0;
83    }
84    float ratio = 0.0;
85    ratio = valueRs / valueR0 * getEnvCorrectRatio(humidityPercent,
       ↪  temperatureCelsius);
86    return 23.534 * pow(ratio,-1.091); //R^2=0.9996
87  }
88
89  // Returns O3 in ppb
90  float MIKROE_2767::getO3_ppb(float humidityPercent, float
     ↪  temperatureCelsius, float valueRL, float valueR0){
91    return getO3_ppm(humidityPercent, temperatureCelsius, valueRL,
       ↪  valueR0)*1000;
92  }
```

A.1.4 MIKROE_3098.h

```
1  /*
2   * MIKROE_3098.h - NO2 click
3   * Created by João Paulo Sousa, FEUP, 28th January 2021
4  */
5
6  #include <Arduino.h>
```

```
#include "stdint.h"

/* MIKROE-3098 Variables */
const uint8_t _NO2_STATUS_REG     = 0x00;
const uint8_t _NO2_LOCK_REG       = 0x01;
const uint8_t _NO2_TIACN_REG      = 0x10;
const uint8_t _NO2_REFCN_REG      = 0x11;
const uint8_t _NO2_MODECN_REG     = 0x12;
const uint8_t _NO2_WRITE_MODE      = 0x00;
const uint8_t _NO2_READ_ONLY_MODE = 0x01;

const uint8_t _NO2_EXT_TIA_RES       = 0x00;
const uint8_t _NO2_2750_OHM_TIA_RES = 0x04;
const uint8_t _NO2_3500_OHM_TIA_RES = 0x08;
const uint8_t _NO2_7K_OHM_TIA_RES    = 0x0C;
const uint8_t _NO2_14K_OHM_TIA_RES   = 0x10;
const uint8_t _NO2_35K_OHM_TIA_RES   = 0x14;
const uint8_t _NO2_120K_OHM_TIA_RES = 0x18;
const uint8_t _NO2_350K_OHM_TIA_RES = 0x1C;
const uint8_t _NO2_10_OHM_LOAD_RES   = 0x00;
const uint8_t _NO2_33_OHM_LOAD_RES   = 0x01;
const uint8_t _NO2_50_OHM_LOAD_RES   = 0x02;
const uint8_t _NO2_100_OHM_LOAD_RES = 0x03;

const uint8_t _NO2_VREF_INT              = 0x00;
const uint8_t _NO2_VREF_EXT              = 0x80;
const uint8_t _NO2_20_PERCENTS_INT_ZERO = 0x00;
const uint8_t _NO2_50_PERCENTS_INT_ZERO = 0x20;
const uint8_t _NO2_67_PERCENTS_INT_ZERO = 0x40;
const uint8_t _NO2_INT_ZERO_BYPASSED     = 0x60;
const uint8_t _NO2_BIAS_POL_NEGATIVE     = 0x00;
const uint8_t _NO2_BIAS_POL_POSITIVE     = 0x10;
const uint8_t _NO2_0_PERCENTS_BIAS       = 0x00;
const uint8_t _NO2_1_PERCENT_BIAS        = 0x01;
const uint8_t _NO2_2_PERCENTS_BIAS       = 0x02;
const uint8_t _NO2_4_PERCENTS_BIAS       = 0x03;
const uint8_t _NO2_6_PERCENTS_BIAS       = 0x04;
const uint8_t _NO2_8_PERCENTS_BIAS       = 0x05;
const uint8_t _NO2_10_PERCENTS_BIAS      = 0x06;
const uint8_t _NO2_12_PERCENTS_BIAS      = 0x07;
const uint8_t _NO2_14_PERCENTS_BIAS      = 0x08;
const uint8_t _NO2_16_PERCENTS_BIAS      = 0x09;
const uint8_t _NO2_18_PERCENTS_BIAS      = 0x0A;
const uint8_t _NO2_20_PERCENTS_BIAS      = 0x0B;
const uint8_t _NO2_22_PERCENTS_BIAS      = 0x0C;
```

```
52 const uint8_t _NO2_24_PERCENTS_BIAS      = 0x0D;
53
54 const uint8_t _NO2_FET_DIS              = 0x00;
55 const uint8_t _NO2_FET_EN               = 0x80;
56 const uint8_t _NO2_DEEP_SLEEP_MODE      = 0x00;
57 const uint8_t _NO2_2_LEAD_MODE          = 0x01;
58 const uint8_t _NO2_STANDBY_MODE         = 0x02;
59 const uint8_t _NO2_3_LEAD_MODE          = 0x03;
60 const uint8_t _NO2_TEMP_MODE_TIA_OFF    = 0x06;
61 const uint8_t _NO2_TEMP_MODE_TIA_ON     = 0x07;
62
63 const uint8_t _NO2_DEVICE_EN            = 0x00;
64 const uint8_t _NO2_DEVICE_DIS           = 0x01;
65
66 const uint8_t ADC_DEVICE_ADDR           = 0x4D; //0x4D = 77
67
68 #ifndef MIKROE_3098_h
69 #define MIKROE_3098_h
70
71 class MIKROE_3098
72 {
73   public:
74     void firstRun( void );
75     float getNO2_ppm(float Vgas0 = 0); //Vgas0 is the offset,
       ↪  corresponding to voltage to 0 ppm
76     float getNO2_ppb(float Vgas0 = 0);
77 };
78
79 #endif
```

A.1.5 MIKROE_3098.cpp

```
1  /*
2   * MIKROE_3098.cpp - NO2 click
3   * Created by João Paulo Sousa, FEUP, 28th January 2021
4  */
5
6  #include <Wire.h>
7  #include "MIKROE_3098.h"
8  #include "HardwareSerial.h"
9
10 // Function to run setup(). Sets sensor
11 void MIKROE_3098::firstRun(){
12     Wire.begin();                // join i2c bus with address #4
```

```
13      delay(20);
14      Wire.beginTransmission(77); // transmit to device 0x48=72; 0x4D =
        ↪  77
15      delay(20);
16      uint8_t writeReg1[ 2 ];
17      writeReg1[ 0 ] = _NO2_MODECN_REG;  // Mode Control Register
18      writeReg1[ 1 ] = _NO2_DEEP_SLEEP_MODE;  // Mode of Operation
        ↪  selection: 000 Deep Sleep (default)
19      uint8_t writeReg2[ 2 ];
20      writeReg2[ 0 ] = _NO2_LOCK_REG;  // LOCK -- Protection Register
        ↪  (Address 0x01)
21      writeReg2[ 1 ] = _NO2_WRITE_MODE;  // Means: Write protection: 0
        ↪  Registers 0x10, 0x11 in write mode
22      uint8_t writeReg3[ 2 ];
23      writeReg3[ 0 ] = _NO2_TIACN_REG;  // TIACN -- TIA Control Register
        ↪  (Address 0x10)
24      writeReg3[ 1 ] = _NO2_2750_OHM_TIA_RES | _NO2_100_OHM_LOAD_RES;  //
        ↪  0x03 Means: TIA_GAIN= external resistance and RLOAD=100 Ohm
        ↪  AND 0x07 means: TIA_GAIN= 2.75kOhm and RLOAD=100 Ohm
25      uint8_t writeReg4[ 2 ];
26      writeReg4[ 0 ] = _NO2_REFCN_REG;  // REFCN -- Reference Control
        ↪  Register (Address 0x11)
27      writeReg4[ 1 ] = _NO2_VREF_INT | _NO2_50_PERCENTS_INT_ZERO |
        ↪  _NO2_BIAS_POL_NEGATIVE | _NO2_0_PERCENTS_BIAS;  // Means:
        ↪  Reference voltage source = internal, internal zero selection (%
        ↪  of source reference) = 50%, bias polarity = negative (VRE>VWE),
        ↪  bias selection (% of source reference = 0%)
28      Wire.write(writeReg1,2);
29      Wire.write(writeReg2,2);
30      Wire.write(writeReg3,2);
31      Wire.write(writeReg4,2);
32      delay(30);
33      Wire.endTransmission();    // stop transmitting
34  }
35
36  // Gets sensor data. Retuns NO2 in ppm
37  float MIKROE_3098::getNO2_ppm(float Vgas0){
38    uint8_t readData;
39    uint16_t ADC_data;
40    float NO2_ppm;
41    Wire.requestFrom(77, 2, 1);    // request info (2 bytes) from slave
      ↪  device 0x48=72. 3rd ardgument: true will send a stop message
      ↪  after the request; 0x4D = 77 (USE 0x42!!!)
42    if (Wire.available()==2) { // slave may send less than requested
43      readData = Wire.read(); //we will receive the first byte (8 bits)
```

```
44      ADC_data = readData;
45      ADC_data <<= 8;// Bitwise left shift assignment
46      readData = Wire.read(); // we will receive the 2nd byte (8 bits)
47      ADC_data |= readData; // we will concatenate the two bits. Now, we
         ↪ have a 16bit var
48      float NO2_voltage;
49      float NO2_M; // calibration factor
50      float NO2_Vref = 2.0480; // given by MCP1501T-20E/CHY
51      // SPECS 3SP-NO2-20 (NO2 sensor) is the sensor MIKROE says in
         ↪ documentation. Label of the sensor says 110-508 (3SP-NO2-5F)
52      //12-bit A/D converter Microchip MCP3221 gives us a value converted
         ↪ to 12-bit (2^12=4096), i.e, 4095 divisions
53      NO2_voltage = (ADC_data/4095.0)*NO2_Vref;
54      // According to the document ULPSM-NO2 968-047, concentration(in
         ↪ ppm) = (1/M)*(Vgas-Vgas0), Vgas0 = Vref + Voffset
55      // M[V/ppm] = Sensitivity[nA/ppm] * TIA_Gain[kV/A] * 10^-9[A/nA] *
         ↪ 10^3[V/kV]       and      TIA_Gain[kV/A] for NO2 = 2.75
         ↪ (According to what is wrote via I2C)
56      // sensitivity is 26.90nA/ppm according to sensor label
57      NO2_M = 26.90 * 2.75 * 10e-9 * 10e3; // 1/M = 137.793241
58      NO2_ppm = ( 1 / NO2_M ) * (NO2_voltage-Vgas0) ; //
59    }
60    Wire.endTransmission();    // stop transmitting
61    return NO2_ppm;
62  }
63
64  // Retuns NO2 in ppb
65  float MIKROE_3098::getNO2_ppb(float Vgas0){
66    return getNO2_ppm(Vgas0)*1000;
67  }
```

A.1.6 Sparkfun_CCS811_BME280.h

```
1   /*
2    * MIKROE_2767.h - Ozone 2 click
3    * Created by João Paulo Sousa, FEUP, 28th January 2021
4   */
5   #include <Arduino.h>
6   #include "stdint.h"
7
8   #ifndef MIKROE_2767_h
9   #define MIKROE_2767_h
10
11  class MIKROE_2767
```

```
12 {
13   public:
14     void firstRun( void );
15     float getO3_ppm(float humidityPercent = 60, float
       ↪  temperatureCelsius = 20, float valueRL = 100, float valueR0 =
       ↪  7.1341);
16     float getO3_ppb(float humidityPercent = 60, float
       ↪  temperatureCelsius = 20, float valueRL = 100, float valueR0 =
       ↪  7.1341);
17   private:
18     uint32_t readADC_Data( );
19     float readRs (float valueRL);
20     float getEnvCorrectRatio(float humidityPercent, float
       ↪  temperatureCelsius);
21 };
22 #endif
```

A.1.7 Sparkfun_CCS811_BME280.cpp

```
1  /*
2   * MIKROE_2767.cpp - Ozone 2 click
3   * Created by João Paulo Sousa, FEUP, 28th January 2021
4  */
5
6  #include "MIKROE_2767.h"
7  #include "HardwareSerial.h"
8  #include <Wire.h>
9  #include <SPI.h>
10
11 // SPI pins
12 const byte pinSCK = 14;
13 const byte pinMISO = 2; // =SDO
14 const byte pinMOSI = 15;
15 const byte pinCS = 13; // =SS
16
17 SPIClass hspi(HSPI); //we need to create a spi object because we will
    ↪  use two spi channels (for LoRa and for MIKROE-2767 (ozone 2 click)
18
19 // Function to run in setup(), sets SPI configurations for the sensor
    ↪  module
20 void MIKROE_2767::firstRun( void )
21 {
22     pinMode( pinMISO, INPUT );
23     pinMode( pinCS, OUTPUT );
```

```
24      digitalWrite( pinCS, HIGH );
25      hspi.begin(pinSCK, pinMISO, pinMOSI, pinCS); // start the SPI
26      hspi.setBitOrder( MSBFIRST );
27      hspi.setDataMode( SPI_MODE0 );
28      hspi.setClockDivider( SPI_CLOCK_DIV16 );
29  }
30
31  // Get sensor data
32  uint32_t MIKROE_2767::readADC_Data( ){
33    uint8_t ADC_byte[3];
34    uint32_t ADC_data;
35    digitalWrite( pinCS, LOW ); // set CS low = turn on SPI data
      ↪  communication
36    delay(50);
37    while(digitalRead(pinMISO));
38    ADC_byte[0] = hspi.transfer(0x00);
39    ADC_byte[1] = hspi.transfer(0x00);
40    ADC_byte[2] = hspi.transfer(0x00);
41    ADC_data = 0;
42    ADC_data = ADC_byte[0];
43    ADC_data <<= 8;
44    ADC_data |= ADC_byte[1];
45    ADC_data <<= 8;
46    ADC_data |= ADC_byte[2];
47    digitalWrite( pinCS, HIGH ); //set CS high = turn off SPI data
      ↪  communication
48    if ( ADC_data < 2097151 && ADC_data> 0) //2^21-1
49        return ADC_data;
50    else return 0;
51    delay(50);
52  }
53
54
55  // Convert ADC_data from SPI communicatin into resistance of the sensor
56  float MIKROE_2767::readRs (float valueRL){
57    float vRL = ((float)readADC_Data())/2097152.0*5.0; // Voltage on load
      ↪  resistance. 21^2-1=2097152
58    if(!vRL) return 0.0f; // division by zero prevention
59    float rS = (5.0/vRL - 1.0) * valueRL; // resistance of the sensor. RL
      ↪  is the variable resistance on the sensor+100!
60    return rS;
61  }
62
63  //Get correction to apply on Rs depending on environmental
```

```
64 float MIKROE_2767::getEnvCorrectRatio(float humidityPercent, float
    ↪ temperatureCelsius) {
65   // Select the right equation based on humidity. If default value,
      ↪ ignore correction ratio
66   if(humidityPercent == 60 && temperatureCelsius == 20) {
67     return 1.0;
68   }
69   else if(humidityPercent > 75) {  // For humidity > 75%, use the 85%
      ↪ curve
70     return -0.0141 * temperatureCelsius + 1.5623;// R^2 = 0.9986
71   }
72   else if(humidityPercent > 50) { // For humidity > 50%, use the 60%
      ↪ curve
73     return -0.0119 * temperatureCelsius + 1.3261; // R^2 = 0.9976
74   }
75   return -0.0103 * temperatureCelsius + 1.1507; // Humidity < 50%, use
      ↪ the 30% curve. R^2 = 0.996
76 }
77
78 // Returns O3 in ppm. R0 result from calibration. RL is 100 ohm +
    ↪ resistance in potentiometer
79 float MIKROE_2767::getO3_ppm(float humidityPercent, float
    ↪ temperatureCelsius, float valueRL, float valueR0){
80   float valueRs = readRs(valueRL);
81   if (valueRs < 0) {
82     return 0.0;
83   }
84   float ratio = 0.0;
85   ratio = valueRs / valueR0 * getEnvCorrectRatio(humidityPercent,
      ↪ temperatureCelsius);
86   return 23.534 * pow(ratio,-1.091); //R^2=0.9996
87 }
88
89 // Returns O3 in ppb
90 float MIKROE_2767::getO3_ppb(float humidityPercent, float
    ↪ temperatureCelsius, float valueRL, float valueR0){
91   return getO3_ppm(humidityPercent, temperatureCelsius, valueRL,
      ↪ valueR0)*1000;
92 }
```

A.1.8 LoRa_Sender.h

```
1 /*
2  * LoRa_Sender.h - send information to the "router"
```

```
 * Created by João Paulo Sousa, FEUP, 2nd February 2021
 * Based on 'LoRa Duplex' created by Tom Igoe
*/

#include <Arduino.h>
#include "stdint.h"

#ifndef LoRa_Sender_h
#define LoRa_Sender_h

class LoRa_Sender
{
  public:
    void firstRun( void );
    void sendMessage(String message, long timeToTyToSend = 25000, long
     ↪  timeUntilNextRead = 90000, byte nodeToConnect = 0xF3);
  private:
    void onReceive(int packetSize, byte nodeToConnect);
};
#endif
```

A.1.9 LoRa_Sender.cpp

```
/*
 * LoRa_Sender.cpp - send information to the "router"
 * Created by João Paulo Sousa, FEUP, 2nd February 2021
 * Based on 'LoRa Duplex' created by Tom Igoe
*/

#include "LoRa_Sender.h"
#include "HardwareSerial.h"
#include <SPI.h>
#include <LoRa.h>

SPIClass vspi(VSPI); //we need to create a spi object because we will
 ↪  use two spi channels (for LoRa and for MIKROE-2767 (ozone 2 click)

#define SCK     5    // GPIO5  -- SCK
#define MISO    19   // GPIO19 -- MISO
#define MOSI    27   // GPIO27 -- MOSI
#define SS      18   // GPIO18 -- CS
#define RST     23   // GPIO14 -- RESET
#define DI0     26   // GPIO26 -- IRQ(Interrupt Request)
#define BAND    868E6
```

```
21
22 bool messageIsReceived;
23 byte localAddress = 0xF1;     // address of this device
24
25 // Function to run in setup(), sets LoRa
26 void LoRa_Sender::firstRun(){
27   vspi.begin(SCK, MISO, MOSI, SS); //SCLK, MISO, MOSI, SS
28   LoRa.setSPI(vspi);
29   LoRa.setPins(SS,RST,DI0);// set CS, reset, IRQ pin
30   delay(50);
31   if (!LoRa.begin(868E6)) {     // initialize ratio at 868 MHz
32     Serial.println("LoRa init failed. Check your connections.");
33     delay(2000);
34     ESP.restart();      // if failed, restart
35   }
36   Serial.println("LoRa init succeeded.");
37 }
38
39 // Create packet with the message and send via LoRa
40 void LoRa_Sender::sendMessage(String message, long timeToTyToSend, long
    ↪  timeUntilNextRead, byte nodeToConnect){
41   long firstMessage = millis();
42   messageIsReceived = false;
43   // stop trying to send message after 25s trying to receive feedback
     ↪  of the message sent
44   while (messageIsReceived == false && (millis() - firstMessage <
     ↪  timeToTyToSend)){
45     LoRa.beginPacket();       // start packet
46     LoRa.write(nodeToConnect);      // add destination address
47     LoRa.write(localAddress);     // add sender address
48     LoRa.write(message.length()); // add payload length
49     LoRa.print(message);      // add payload
50     LoRa.endPacket();        // finish packet and send it
51     Serial.println("Sent: " + message);
52     delay(10);
53     long firstTrial = millis();
54    // after 2 seconds not receiving feedback, send the packet again
55     while (messageIsReceived == false && (millis() - firstTrial) < 2000
       ↪  ){
56       onReceive(LoRa.parsePacket(), nodeToConnect);
57       delay(10);
58     }
59   }
60   Serial.println(messageIsReceived);
```

```
61    while((millis() - firstMessage) < timeUntilNextRead){}; // ensure
      ↪ time between readings
62  }
63
64  // Feedback function
65  void LoRa_Sender::onReceive(int packetSize, byte nodeToConnect) {
66    if (packetSize == 0) return;          // if there's no packet, return
67    // read packet header bytes:
68    int recipient = LoRa.read();    // recipient address
69    byte sender = LoRa.read();    // sender address
70    byte incomingLength = LoRa.read(); // incoming msg length
71    String incoming = "";
72    while (LoRa.available()) {incoming += (char)LoRa.read();}
73    if (incomingLength != incoming.length()) {   // check length for
      ↪ error
74      Serial.println("error: message length does not match length");
75      return;           // skip function
76    }
77    // if the recipient isn't this device or broadcast,
78    if (recipient != localAddress && recipient != 0xFF) {
79      Serial.println("This message is not for me.");
80      return;          // skip function
81    }
82    if (incoming == "0xF3 read 0xF1")
83    {
84      Serial.println(incoming);
85      messageIsReceived = true;
86    }
87  }
```

A.2 C++ code uploaded in the MCU of the Node 2

Table A.2 presents the list of files used in programming code of Node 2.

Table A.2: Organisation of programming code files for the Node 2

Function	Description
`Node_2.ino`	Main file
`Gravity_BME680.h`	Header file of BME680 code. Includes function and class declarations
`Gravity_BME680.cpp`	Implementation file of BME680's class. This file contains the reading routines of the BME680
`LoRa_Sender.h`	Header file of LoRa code. Includes function and class declarations
`LoRa_Sender.cpp`	Implementation file of Lora's class. This file contains LoRa communication routines

A.2.1 Node2.ino

```
/*
 * Node 2 - Sensors:  - DFRobot Gravity BME680
 * Created by João Paulo Sousa, FEUP, 28th January 2021
 */

#include <Arduino.h>
#include "stdint.h"
#include <Wire.h>
#include "Gravity_BME680.h"
#include "Lora_Sender.h"

// Declare objects and variables
Gravity_BME680 gravity_bme680; // DFRobot Gravity BME680
LoRa_Sender lora_sender;
Gravity_BME680_DATA_struct sensorData;
int counter = 0;

// Sleeping mode
#define uS_TO_S_FACTOR 1000000  /* Conversion factor for micro seconds
 ↪  to seconds */
#define TIME_TO_SLEEP  600        /* Time ESP32 will go to sleep (in
 ↪  seconds) */
RTC_DATA_ATTR int bootCount = 0;

void setup() {
  Serial.begin(115200);           // start serial for output
  Wire.begin();                 // join i2c bus with address #4
  Serial.println("--- System init ---");
  delay(10);
  gravity_bme680.firstRun();
  lora_sender.firstRun();
  delay(3000);
  getData();
  delay(1000);
  // Defining sleep mode time
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  bootCount++;
  Serial.println(" >>>>> COLLECTING DATA (NODE 2): SAMPLE #" +
   ↪  String(bootCount));
  getData();
  char json[250];
  sprintf(json, "{\"2t\":%02.02f, \"2h\":%02.02f, \"2p\":%02.02f}",
   ↪  sensorData.Temp_C, sensorData.RH_perc, sensorData.Pressure_Pa);
```

```
40    showDataSerial();
41    lora_sender.sendMessage(json,25000,25000);
42    Serial.println("Going to sleep now");
43    Serial.flush();
44    esp_deep_sleep_start();
45    Serial.println("This will never be printed");
46  }
47
48  // Get data from sensors
49  void getData(){
50    sensorData = gravity_bme680.get_Gravity_BME680Data(); //
      ↪  BME680_DATA_struct. Contains T, RH, Pressure
51  }
52
53  // Show data in Serial Monitor
54  void showDataSerial(){
55    Serial.println("Temperature: " + String(sensorData.Temp_C, 2) + "
      ↪  °C");
56    Serial.println("Pressure: " + String(sensorData.Pressure_Pa/1000, 2)
      ↪  + " kPa");
57    Serial.println("RH: " + String(sensorData.RH_perc, 2) + " %");
58  }
59  void loop() {
60
61  }
```

A.2.2 Gravity_BME680.h

```
1  /*
2   * Gravity_BME680.h - DFRobot Gravity BME680
3   * Created by João Paulo Sousa, FEUP, 26th January 2021
4   * Using DFRobot library
5  */
6
7  #ifndef Gravity_BME680_h
8  #define Gravity_BME680_h
9
10 // Create a struct to save sensor data
11 struct Gravity_BME680_DATA_struct
12 {
13   float Temp_C;
14   float Pressure_Pa;
15   float RH_perc;
16 };
```

```
17
18 class Gravity_BME680
19 {
20   public:
21     void firstRun();
22     Gravity_BME680_DATA_struct get_Gravity_BME680Data();
23 };
24 #endif
```

A.2.3 Gravity_BME680.cpp

```
1  /*
2   * Gravity_BME680.cpp - DFRobot Gravity BME680
3   * Created by João Paulo Sousa, FEUP, 26th January 2021
4   * Using DFRobot library
5  */
6
7  #include "Gravity_BME680.h"
8  #include <Wire.h>
9  #include <Arduino.h>
10 #include "DFRobot_BME680_I2C.h"
11
12 Gravity_BME680_DATA_struct gravity_bme680_data;
13 DFRobot_BME680_I2C bme(0x77);  //0x77 I2C address
14
15 // Function to run in setup()
16 void Gravity_BME680::firstRun(){
17   uint8_t rslt = 1;
18   Serial.begin(115200);
19   while(!Serial);
20   delay(1000);
21   Serial.println();
22   while(rslt != 0) {
23     rslt = bme.begin();
24     if(rslt != 0) {
25       Serial.println("bme680 begin failure");
26       delay(2000);
27     }
28   }
29   Serial.println("bme680 begin successful");
30 }
31
32 // Get sensor data
33 Gravity_BME680_DATA_struct Gravity_BME680::get_Gravity_BME680Data(){
```

```
34    bme.startConvert();
35    delay(1000);
36    bme.update();
37    gravity_bme680_data.Temp_C=bme.readTemperature()/100;
38    gravity_bme680_data.Pressure_Pa=bme.readPressure();
39    gravity_bme680_data.RH_perc=bme.readHumidity()/1000;
40    return gravity_bme680_data;
41  }
```

A.2.4 LoRa_Sender.h

This file is the same as the file in the Section A.1.8.

A.2.5 LoRa_Sender.cpp

This file is the same as the file in the Section A.1.9.

A.3 C++ code uploaded in the MCU of the Node 3

Table A.3 presents the list of files used in programming code of Node 3.

Table A.3: Organisation of programming code files for the Node 3

Function	Description
Node_3.ino	Main file
MIKROE_3056.h	Header file of MIKROE-2767 code. Includes function and class declarations
MIKROE_3056.cpp	Implementation file of MIKROE-3056's class. This file contains the reading routines for NH_3, CO and NO_2
SHINYEI_PPD42.h	Header file of SHINYEI PPD42 code. Includes function and class declarations
SHINYEI_PPD42.cpp	Implementation file of SHINYEI PPD42's class. This file contains the reading routines for particulate matter
Sparkfun_CCS811_BME280.h	Header file of Sparkfun SEN-14348 code. Includes function and class declarations
Sparkfun_CCS811_BME280.cpp	Implementation file of Sparkfun SEN-14348's class. This file contains the readings of the BME280 and CCS811
LoRa_Receiver.h	Header file of LoRa code. Includes function and class declarations
LoRa_Receiver.cpp	Implementation file of Lora's class. This file contains LoRa communication routines
MQTT_Client.h	Header file of MQTT configuration code. Includes function and class declarations
MQTT_Client.cpp	Implementation file of MQTT configuration class. This file contains Wi-Fi and MQTT routines

A.3.1 Node3.ino

```
1  /*
2   * Node 3 - Sensors:  - Air Quality 5 click (MIKROE-5065) - NO2, CH
3   *                    - Sparkfun combo CCS811+BME280 - T, RH, TVOC, CO2
4   *                    - Shinyei PPD42: Particulate matter
5   * Created by João Paulo Sousa, FEUP, 26th January 2021
6   */
```

```
7
8  #include "MIKROE_3056.h"
9  #include "SHINYEI_PPD42.h"
10 #include "Sparkfun_CCS811_BME280.h"
11 #include "MQTT_Client.h"
12 #include "LoRa_Receiver.h"
13
14 #include <Arduino.h>
15 #include "stdint.h"
16 #include <Wire.h>
17
18 // Declare objects and variables
19 MIKROE_3056 mikroe_3056;
20 SHINYEI_PPD42 shinyei_ppd42 = SHINYEI_PPD42();
21 Sparkfun_CCS811_BME280 sparkfun_combo;
22 MQTT_Client mqtt_client;
23 LoRa_Receiver lora_receiver;
24
25 SpFun_COMBO_DATA_struct sensorData;
26 float PM_ppL;
27 float NH3_ppm;
28 float CO_ppm;
29 float NO2_ppm;
30 int counter = 0;
31
32
33 void setup() {
34   Serial.begin(115200);           // start serial for output
35   Wire.begin();                 // join i2c bus with address #4
36   Serial.println("--- System init ---");
37   delay(10);
38   sparkfun_combo.firstRun();
39   delay(3000);
40   sensorData = sparkfun_combo.get_SparkfunComboData();
     ↪  //SpFun_COMBO_DATA_struct. Contains CO2, T, RH, TVOC, Pressure
41   delay(1000);
42   mqtt_client.firstRun();
43   lora_receiver.firstRun();
44 }
45
46 //Get data from sensors
47 void getData(){
48   PM_ppL = shinyei_ppd42.getPM_ppL(5000);
49   NH3_ppm = mikroe_3056.getNH3_ppm();
50   CO_ppm = mikroe_3056.getCO_ppm();
```

```
51    NO2_ppm = mikroe_3056.getNO2_ppm();
52    sensorData = sparkfun_combo.get_SparkfunComboData();
      ↪ //SpFun_COMBO_DATA_struct. Contains CO2, T, RH, TVOC, Pressure
53  }
54
55  //Show data in Serial Monitor
56  void showDataSerial(){
57    Serial.println("CO2: " + String(sensorData.CO2_ppm) + " ppm");
58    Serial.println("TVOC: " + String(sensorData.TVOC_ppb) + " ppb");
59    Serial.println("Temperature: " + String(sensorData.Temp_C, 2) + "
      ↪ °C");
60    Serial.println("Pressure: " + String(sensorData.Pressure_Pa/1000, 2)
      ↪ + " kPa");
61    Serial.println("RH: " + String(sensorData.RH_perc, 2) + " %");
62    Serial.println("NO2: " + String(NO2_ppm, 2) + " ppm");
63    Serial.println("CO: " + String(CO_ppm*1000) + " ppb");
64    Serial.println("NH3: " + String(NH3_ppm*1000) + " ppb");
65    Serial.println("PM: " + String(PM_ppL,2) + " pcs/L");
66  }
67
68  //Send data via MQTT
69  void sendDataMQTT(){
70    char json[250];
71    //Atribui para a cadeia de caracteres "json" os valores referentes a
      ↪ temperatura e os envia para a variável do ubidots correspondente
72    sprintf(json, "{\"3t\":%02.02f, \"3h\":%02.02f, \"3p\":%02.02f,
      ↪ \"3co2\":%02.02f, \"3tvoc\":%02.02f, \"3pm\":%02.02f,
      ↪ \"3no2\":%02.02f, \"3co\":%02.02f, \"3nh3\":%02.02f}",
      ↪ sensorData.Temp_C, sensorData.RH_perc, sensorData.Pressure_Pa,
      ↪ sensorData.CO2_ppm, sensorData.TVOC_ppb, PM_ppL, NO2_ppm,
      ↪ CO_ppm*1000, NH3_ppm*1000);
73
74    mqtt_client.testConnection();
75    delay(2500);
76    if(mqtt_client.sendValues(json))
77    {
78      Serial.println("Successfully sent data");
79      Serial.println();
80    }
81    else {Serial.println("Failed to send sensor data");}
82  }
83
84  void loop() {
85
86    counter++;
```

```
87    Serial.println(" >>>>> COLLECTING DATA (NODE 3): SAMPLE #" +
      ↪  String(counter));
88    getData();
89    if (counter > 2) // when counter = 1, data are not fine. And, we want
      ↪  more time for some stability in data
90    {
91      showDataSerial();
92      sendDataMQTT();
93    }
94     //Reserve time to receive messages from another devices
95     long firstTrial = millis();
96     lora_receiver.onReceive(firstTrial, 600e3);
97  }
```

A.3.2 MIKROE_3056.h

```
1  /*
2   * MIKROE_3056.h - Air Quality 5 click
3   * Created by João Paulo Sousa, FEUP, 25th January 2021
4  */
5
6  #ifndef MIKROE_3056_h
7  #define MIKROE_3056_h
8
9  class MIKROE_3056
10 {
11   public:
12     float getNO2_ppm(float valueR0 = 15000);
13     float getNH3_ppm(float valueR0 = 1000000);
14     float getCO_ppm(float valueR0 = 1000000);
15 };
16 #endif
```

A.3.3 MIKROE_3056.cpp

```
1  /*
2   * MIKROE_3056.h - Air Quality 5 click
3   * Created by João Paulo Sousa, FEUP, 25th January 2021
4  */
5
6  #include "MIKROE_3056.h"
7  #include "HardwareSerial.h"
8  #include <Wire.h>
9
```

```
// Chanel data
const uint16_t _AIRQ5_DATA_CHANNEL_NO2 = 0x4000;
const uint16_t _AIRQ5_DATA_CHANNEL_NH3 = 0x5000;
const uint16_t _AIRQ5_DATA_CHANNEL_CO  = 0x6000;

uint16_t setConfig = 0x8583;

// Get NO2 value
float MIKROE_3056::getNO2_ppm(float valueR0){
  // convert digital to analog value
  // send configuration data and channel to read
  Wire.begin();
  delay(10);
  Wire.beginTransmission(72); // transmit to device 0x48=72; 0x4D = 77
  delay(10);
  uint8_t writeReg1[ 3 ];
  uint16_t _data = setConfig | _AIRQ5_DATA_CHANNEL_NO2;
  writeReg1[ 0 ] = 0x01; //_AIRQ5_REG_POINTER_CONFIG      = 0x01
  writeReg1[ 1 ] = _data >> 8;
  writeReg1[ 2 ] = _data & 0x00FF;
  Wire.write(writeReg1,3);
  delay(50);
  Wire.endTransmission();    // stop transmitting
  delay(20);
  // send info about the pointer to read
  Wire.beginTransmission(72);
  delay(50);
  uint8_t writeReg2[1];
  writeReg2[0] = 0x00; //_AIRQ5_REG_POINTER_CONVERT     = 0x00;
  Wire.write(writeReg2,1);
  delay(20);
  Wire.endTransmission();
  // request data
  uint8_t readReg [ 2 ];
  uint16_t dataValue;
  delay(50);
  Wire.requestFrom(72, 2, 1);
  delay(20);
  if (Wire.available()){
    readReg[0] = Wire.read();
    readReg[1] = Wire.read();
  }
  delay(15);
  Wire.endTransmission();
  dataValue = readReg[ 0 ];
```

```
55    dataValue <<= 8;
56    dataValue |= readReg[ 1 ];
57    dataValue >>= 4;
58    float voltageValue = (float(dataValue) / 2047) * 3.3; // convert to
       ↪  voltage value
59    float valueRs = ((voltageValue) / (3.3 - voltageValue)) * 15000; //
       ↪  resistence created in sensor, 15000 ohm is the value of resistor
       ↪  in voltage divider
60    float ratio = valueRs/valueR0;
61    float result = pow(ratio, 1.007) / 6.855;
62    return result;
63  }
64
65  // Get CO value
66  float MIKROE_3056::getCO_ppm(float valueR0){
67    // convert digital to analog value
68    // send configuration data and channel to read
69    Wire.begin();
70    delay(10);
71    Wire.beginTransmission(72); // transmit to device 0x48=72; 0x4D = 77
72    delay(10);
73    uint8_t writeReg1[ 3 ];
74    uint16_t _data = setConfig | _AIRQ5_DATA_CHANNEL_CO;
75    writeReg1[ 0 ] = 0x01; //_AIRQ5_REG_POINTER_CONFIG      = 0x01
76    writeReg1[ 1 ] = _data >> 8;
77    writeReg1[ 2 ] = _data & 0x00FF;
78    Wire.write(writeReg1,3);
79    delay(50);
80    Wire.endTransmission();    // stop transmitting
81    delay(20);
82    // send info about the pointer to read
83    Wire.beginTransmission(72);
84    delay(50);
85    uint8_t writeReg2[1];
86    writeReg2[0] = 0x00; //_AIRQ5_REG_POINTER_CONVERT     = 0x00;
87    Wire.write(writeReg2,1);
88    delay(20);
89    Wire.endTransmission();
90    // request data
91    uint8_t readReg [ 2 ];
92    uint16_t dataValue;
93    delay(50);
94    Wire.requestFrom(72, 2, 1);
95    delay(20);
96    if (Wire.available()){
```

```
97      readReg[0] = Wire.read();
98      readReg[1] = Wire.read();
99    }
100   delay(15);
101   Wire.endTransmission();
102   dataValue = readReg[ 0 ];
103   dataValue <<= 8;
104   dataValue |= readReg[ 1 ];
105   dataValue >>= 4;
106
107   float voltageValue = (float(dataValue) / 2047) * 3.3; // convert to
      ↪ voltage value
108   float valueRs = ((voltageValue) / (3.3 - voltageValue)) * 1000000; //
      ↪ resistence created in sensor, 1000000 ohm is the value of
      ↪ resistor in voltage divider
109   float ratio = valueRs/valueR0;
110   float result = pow(ratio, -1.179) * 4.385;
111   return result;
112 }
113
114 //Get NH3 value
115 float MIKROE_3056::getNH3_ppm(float valueR0){
116   // convert digital to analog value
117   // send configuration data and channel to read
118   Wire.begin();
119   delay(10);
120   Wire.beginTransmission(72); // transmit to device 0x48=72; 0x4D = 77
121   delay(10);
122   uint8_t writeReg1[ 3 ];
123   uint16_t _data = setConfig | _AIRQ5_DATA_CHANNEL_NH3;
124   writeReg1[ 0 ] = 0x01; //_AIRQ5_REG_POINTER_CONFIG      = 0x01
125   writeReg1[ 1 ] = _data >> 8;
126   writeReg1[ 2 ] = _data & 0x00FF;
127   Wire.write(writeReg1,3);
128   delay(50);
129   Wire.endTransmission();    // stop transmitting
130   delay(20);
131   // send info about the pointer to read
132   Wire.beginTransmission(72);
133   delay(50);
134   uint8_t writeReg2[1];
135   writeReg2[0] = 0x00; //_AIRQ5_REG_POINTER_CONVERT     = 0x00;
136   Wire.write(writeReg2,1);
137   delay(20);
138   Wire.endTransmission();
```

```
139   // request data
140   uint8_t readReg [ 2 ];
141   uint16_t dataValue;
142   delay(50);
143   Wire.requestFrom(72, 2, 1);
144   delay(20);
145   if (Wire.available()){
146     readReg[0] = Wire.read();
147     readReg[1] = Wire.read();
148   }
149   delay(15);
150   Wire.endTransmission();
151   dataValue = readReg[ 0 ];
152   dataValue <<= 8;
153   dataValue |= readReg[ 1 ];
154   dataValue >>= 4;
155   float voltageValue = (float(dataValue) / 2047) * 3.3; //convert to
      ↪ voltage value
156   float valueRs = ((voltageValue) / (3.3 - voltageValue)) * 1000000;
      ↪ //resistence created in sensor, 1000000 ohm is the value of
      ↪ resistor in voltage divider
157   float ratio = valueRs/valueR0;
158   float result = pow(ratio, -1.67) / 1.47;
159   return result;
160 }
```

A.3.4 SHINYEI_PPD42.h

```
1  /*
2   * SHINYEI_PPD42.h - Shinyei PPD42, Particulate matter
3   * Created by João Paulo Sousa, FEUP, 26th January 2021
4   * Adapted from SHINYEI example
5  */
6
7  #ifndef SHINYEI_PPD42_h
8  #define SHINYEI_PPD42_h
9
10 class SHINYEI_PPD42
11 {
12   public:
13     SHINYEI_PPD42();
14     float getPM_ppL(unsigned long sampletime_ms = 2000);
15 };
16 #endif
```

A.3.5 SHINYEI_PPD42.cpp

```
/*
 * SHINYEI_PPD42.cpp - Shinyei PPD42, Particulate matter
 * Created by João Paulo Sousa, FEUP, 26th January 2021
 * Adapted from SHINYEI example
*/


#include "SHINYEI_PPD42.h"
#include "HardwareSerial.h"
#include <Arduino.h>
#include <Wire.h>

int pin = 25;
unsigned long duration;
unsigned long starttime;
unsigned long lowpulseoccupancy = 0;

SHINYEI_PPD42::SHINYEI_PPD42() {
  pinMode(pin,INPUT);
}

// Retuns PM in pieces per liter
float SHINYEI_PPD42::getPM_ppL(unsigned long sampletime_ms) {
  unsigned long duration;
  unsigned long starttime = millis(); // get the current time
  unsigned long lowpulseoccupancy = 0;
  float ratio = 0;
  float concentration = 0;
  bool isSampling = true;

  while(isSampling){ //loop during sampling time
    duration = pulseIn(pin, LOW);
    lowpulseoccupancy = lowpulseoccupancy + duration;
    if ((millis()-starttime) >= sampletime_ms) // after end of the
     ↪  sampling time
    {
      ratio = lowpulseoccupancy/(sampletime_ms*10.0);
      concentration = 1.1 * pow(ratio,3) - 3.8 * pow(ratio,2) + 520 *
       ↪  ratio + 0.62; // using spec sheet curve
      concentration = concentration / 0.283; // converting pcs/0.01cf
       ↪  to pcs/liter
      starttime = millis();
      isSampling = false; //stop
```

```
41      }
42    }
43    return concentration;
44  }
```

A.3.6 Sparkfun_CCS811_BME280.h

This file is the same as the file in the Section A.1.6.

A.3.7 Sparkfun_CCS811_BME280.cpp

This file is the same as the file in the Section A.1.7.

A.3.8 LoRa_Receiver.h

```
1  /*
2   * LoRa_Receiver.h - receive information from nodes and send a feedback
    ↪  message
3   * Created by João Paulo Sousa, FEUP, 3nd February 2021
4   * Based on 'LoRa Duplex' created by Tom Igoe
5  */
6
7  #include <Arduino.h>
8  #include "stdint.h"
9
10 #ifndef LoRa_Receiver_h
11 #define LoRa_Receiver_h
12
13 class LoRa_Receiver
14 {
15   public:
16     void firstRun( void );
17     void sendFeedback(byte nodeThatConnected);
18     void onReceive(long firstTrial, long timeToListen = 17500);
19     void onReceiveCycle();
20 };
21 #endif
```

A.3.9 LoRa_Receiver.cpp

```
1  /*
2   * LoRa_Receiver.cpp - receive information from nodes and send a
    ↪  feedback message
3   * Created by João Paulo Sousa, FEUP, 3nd February 2021
```

```
 * Based on 'LoRa Duplex' created by Tom Igoe
*/

#include "LoRa_Receiver.h"
#include "MQTT_Client.h"
#include "HardwareSerial.h"
#include <SPI.h>
#include <LoRa.h>

SPIClass vspi(VSPI); // we need to create a spi object because we will use two spi channels (for LoRa and for MIKROE-2767 (ozone 2 click)

#define SCK     5    // GPIO5  -- SCK
#define MISO    19   // GPIO19 -- MISO
#define MOSI    27   // GPIO27 -- MOSI
#define SS      18   // GPIO18 -- CS
#define RST     23   // GPIO14 -- RESET
#define DI0     26   // GPIO26 -- IRQ (Interrupt Request)
#define BAND    868E6

bool messageIsReceived;
byte localAddress = 0xF3;     // address of this device
MQTT_Client mqtt_client_lora;

// Function to run in setup(), sets LoRa
void LoRa_Receiver::firstRun(){
  vspi.begin(SCK, MISO, MOSI, SS); //SCLK, MISO, MOSI, SS
  LoRa.setSPI(vspi);
  LoRa.setPins(SS,RST,DI0);// set CS, reset, IRQ pin
  delay(50);
  if (!LoRa.begin(868E6)) {  // initialize ratio at 868 MHz
    Serial.println("LoRa init failed. Check your connections.");
    delay(2000);
    ESP.restart();        // if failed, restart
  }
  Serial.println("LoRa init succeeded.");
}

// Send feedback if receives messages from another node
void LoRa_Receiver::sendFeedback(byte nodeThatConnected){
  long firstMessage = millis();
  messageIsReceived = false;
  String message;
  if (nodeThatConnected == 0xF1){
    message = "0xF3 read 0xF1";
```

```
48    }
49    else if (nodeThatConnected == 0xF2){
50      message = "0xF3 read 0xF2";
51    }
52    while ((millis() - firstMessage < 1500)){  // send new feedback
      ↪  message each 1500 ms
53      LoRa.beginPacket(); // start packet
54      LoRa.write(nodeThatConnected); // add destination address
55      LoRa.write(localAddress);  // add sender address
56      LoRa.write(message.length()); // add payload length
57      LoRa.print(message); // add payload
58      LoRa.endPacket();  // finish packet and send it
59      delay(10);
60    }
61  }
62
63  // Receive packets form another sensors
64  void LoRa_Receiver::onReceive(long firstTrial, long timeToListen) {
65    while ((millis() - firstTrial) < timeToListen ){
66      onReceiveCycle();
67    }
68  }
69
70  // Verify packet received and send to MQTT
71  void LoRa_Receiver::onReceiveCycle(){
72    int packetSize = LoRa.parsePacket();
73    if (packetSize == 0){
74      //Serial.print(".");
75      return;           // if there's no packet, return
76    }
77    // read packet header bytes:
78    int recipient = LoRa.read();          // recipient address
79    byte sender = LoRa.read();            // sender address
80    byte incomingLength = LoRa.read();    // incoming msg length
81    String incoming = "";
82    String newAvailable = "";
83    while (LoRa.available()) {
84      incoming += (char)LoRa.read();
85    }
86    char json[250] = "";
87    for (int i = 0 ; i < incoming.length() ; i++ ){
88      json[i]=incoming[i];
89    }
90
```

```
91    if (incomingLength != incoming.length()) {   // check length for
       ↪  error
92      Serial.println("error: message length does not match length");
93      return; // skip function
94    }
95    // if the recipient isn't this device or broadcast,
96    if (recipient != localAddress && recipient != 0xFF) {
97      Serial.println("This message is not for me.");
98      return; // skip function
99    }
100   //if message is for this device, or broadcast, print details:
101   Serial.println("Received from: 0x" + String(sender, HEX) + ", Sent
       ↪  to: 0x" + String(recipient, HEX) + ", Length: " +
       ↪  String(incomingLength));
102   sendFeedback(sender);
103   mqtt_client_lora.testConnection();
104   delay(200);
105   // Serial.println("Sent to MQTT: " + String(json));
106   if(mqtt_client_lora.sendValues(json))
107   {
108     Serial.println("Successfully sent data");
109     Serial.println();
110   }
111   else
112   {
113     Serial.println("Failed to send sensor data");
114     Serial.println();
115   }
116 }
```

A.3.10 MQTT_Client.h

```
1  /*
2   * MQTT_Client.h - Connecting ubidots via MQTT
3   * Created by João Paulo Sousa, FEUP, 27th January 2021
4   * Adapted from
     ↪  https://www.fernandok.com/2018/11/introducao-ao-protocolo-mqtt-com.html
5  */
6
7  #ifndef MQTT_Client_h
8  #define MQTT_Client_h
9
10 class MQTT_Client
11 {
```

```
12   public:
13     void reconnect();
14     void testConnection();
15     bool mqttInit();
16     void firstRun();
17     bool sendValues(char json[250]);
18 };
19 #endif
```

A.3.11 MQTT_Client.cpp

```
1  /*
2   * MQTT_Client.cpp - Connecting to ubidots via MQTT
3   * Created by João Paulo Sousa, FEUP, 27th January 2021
4   * Using the library for mqtt https://github.com/knolleary/pubsubclient
5  */
6
7  #include <WiFi.h>
8  #include <PubSubClient.h>
9  #include <Arduino.h>
10 #include "MQTT_Client.h"
11
12 #define WIFISSID "*****************" // Wi-Fi SSID
13 #define PASSWORD "*****************" // Wi-Fi Password
14 #define TOKEN "*****************" // Ubidots token
15 #define DEVICE_ID "*****************" // Device ID
16 #define SERVER "things.ubidots.com" // MQTT broker server
17 #define PORT 1883
18 #define TOPIC "/v1.6/devices/node_3"
19
20 WiFiClient ubidots; // WiFiClient Object for the Wi-Fi connection
21 PubSubClient client(ubidots); // PubSubClient object for
   ↪ publish-subscribe
22
23 // Reconnect MQTT connection
24 void MQTT_Client::reconnect()
25 {
26   int counterTrials = 0;
27
28   while (!client.connected()) //Loop until MQTT is established
29   {
30     Serial.println("Attempting MQTT connection...");
31     if (client.connect(DEVICE_ID, TOKEN,""))  // Try to connect
32       Serial.println("Connected");
```

```
33      else
34      {
35        Serial.print("failed, rc=");
36        Serial.print(client.state());
37        Serial.println(" try again in 2 seconds");
38        delay(2000);
39        counterTrials++;
40        if (counterTrials>30){ //if there are no succes in the operation,
           ↪  try to reconnect the device
41          ESP.restart();
42        }
43      }
44    }
45    Serial.println("Reconnected");
46  }
47
48  // Test connection. If it doesn't connect, try to reconnect
49  void MQTT_Client::testConnection(){
50    if(!client.connected())
51      reconnect();
52  }
53
54  // Initialise MQTT connection
55  bool MQTT_Client::mqttInit()
56  {
57    WiFi.begin(WIFISSID, PASSWORD);
58    int counter = 0;
59    while (WiFi.status() != WL_CONNECTED)  // Try to connect to Wi-Fi. If
       ↪  there is no success, restart device
60    {
61      delay(1000);
62      Serial.println("Establishing connection to Wi-Fi..");
63      if (counter>15){ESP.restart();}
64      counter++;
65    }
66    Serial.println("Connected to network");
67    client.setServer(SERVER, PORT); // Sets MQTT server
68    while(!client.connect(DEVICE_ID, TOKEN, "")) // Connects with Ubidots
69    {
70        Serial.println("MQTT - Connect error");
71        return false;
72    }
73    Serial.println("MQTT - Connect ok");
74    return true;
75  }
```

```
76
77 // Function to run in setup(), connect to MQTT
78 void MQTT_Client::firstRun(){
79   if(!mqttInit())
80   {
81     delay(3000);
82     Serial.println("MQTT Failed!");
83     ESP.restart();
84   }
85   Serial.println("MQTT OK");
86 }
87
88 // Send data via MQTT
89 bool MQTT_Client::sendValues(char json[250])
90 {
91   if(!client.publish(TOPIC, json)){return false;} // if no success,
     ↪  returns false
92   Serial.println("Sent via MQTT: " + String(json));
93   return true; // if well successed, returns true
94 }
```