Team: Charalampos Inglezos, Michailidou Maria

Project GitHub Repository

Air Quality Index (AQI) is a critical factor for human health, especially in Europe where many cities experience high pollution levels. Both short-term and long-term exposure to polluted air can cause serious health issues like stroke, lung cancers, asthma, and respiratory infections. The World Health Organization also links air pollution to type 2 diabetes, obesity, Alzheimer's, and dementia. The International Agency for Research on Cancer identifies PM2.5, a common air pollutant, as a leading cause of cancer. A recent global review found that chronic exposure can affect every organ in the body, complicating and exacerbating existing health conditions.

Children and adolescents are particularly at risk due to their developing bodies and immune systems. In 2020, air pollution was responsible for approximately 238,000 premature deaths in the EU. Despite a 45% reduction in premature deaths due to air pollution since 2005, significant sources of particulate matter still include residential and industrial activities. The EU and WHO have established guidelines and standards to combat air pollution, with the European Green Deal aiming to align EU air quality standards more closely with WHO recommendations. By 2030, the EU aims to reduce premature deaths from air pollution by more than 55% compared to 2005 levels. Air pollution affects people differently, with children, the elderly, and those with pre-existing conditions being more vulnerable. Socio-economic factors also play a role; poorer communities often face higher exposure due to proximity to busy roads or industrial areas. Regions with lower GDP per capita, particularly in Eastern and South-eastern Europe, experience higher levels of PM2.5 pollution due to the use of low-quality solid fuels for heating.

Mapping mortality rates against regional wealth in Europe shows that areas with higher pollution levels have higher rates of premature deaths. Besides death, air pollution causes significant morbidity, contributing to chronic diseases and disabilities, which burden both individuals and healthcare systems.

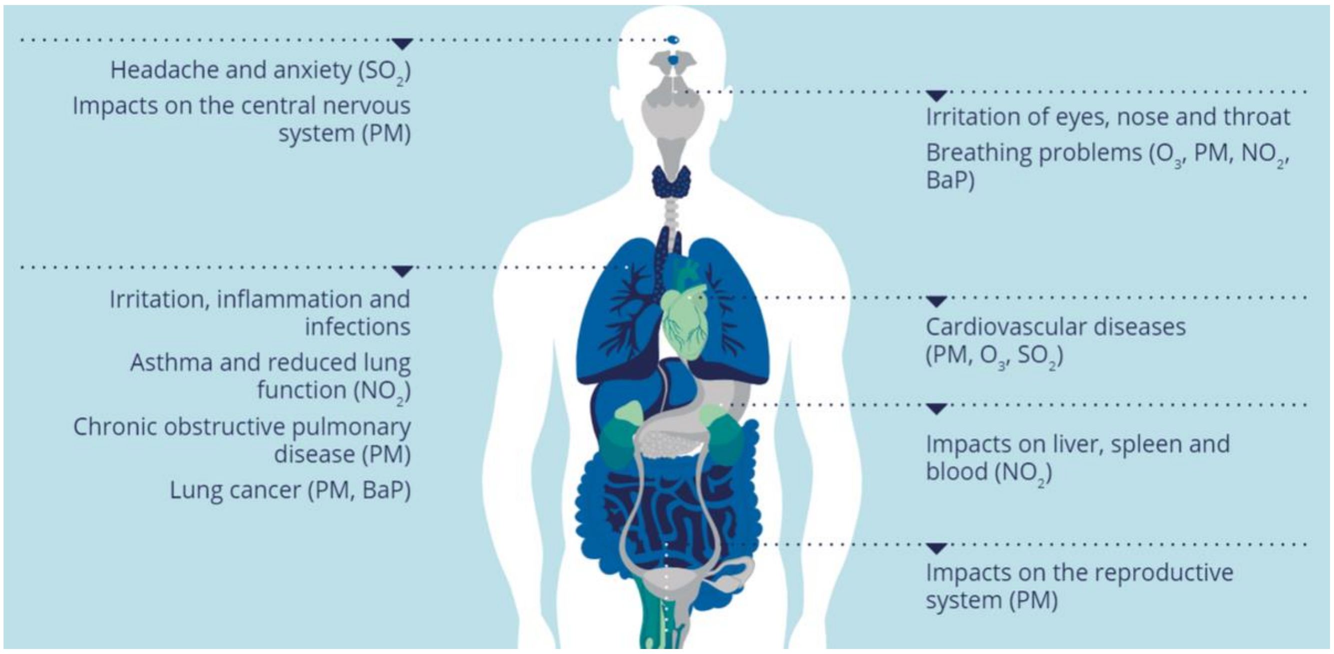

Headache and anxiety (SO_2)
Impacts on the central nervous system (PM)
Irritation of eyes, nose and throat
Breathing problems (O_3, PM, NO_2, BaP)
Irritation, inflammation and infections
Asthma and reduced lung function (NO_2)
Chronic obstructive pulmonary disease (PM)
Lung cancer (PM, BaP)
Cardiovascular diseases (PM, O_3, SO_2)
Impacts on liver, spleen and blood (NO_2)
Impacts on the reproductive system (PM)

Image copyright: EEA, "Healthy environment, healthy lives ," 2019

In 2019, exposure to PM2.5 led to significant health issues, including chronic obstructive pulmonary disease and diabetes. Efforts to reduce air pollution are crucial for improving public health, reducing healthcare costs, and achieving environmental goals. AQI serves as a vital tool for tracking and addressing these pollution levels to protect human health.

Impact of Air Pollution on Health in Europe in 2021

In 2021, air pollution had a significant impact on health across Europe, with thousands of deaths attributed to key pollutants. For PM2.5 fine particulate, above the WHO guideline level of 5 µg/m³ led to 253,000 deaths in the EU-27. When considering a larger group of 40 EEA countries, the number rose to 293,000 deaths.

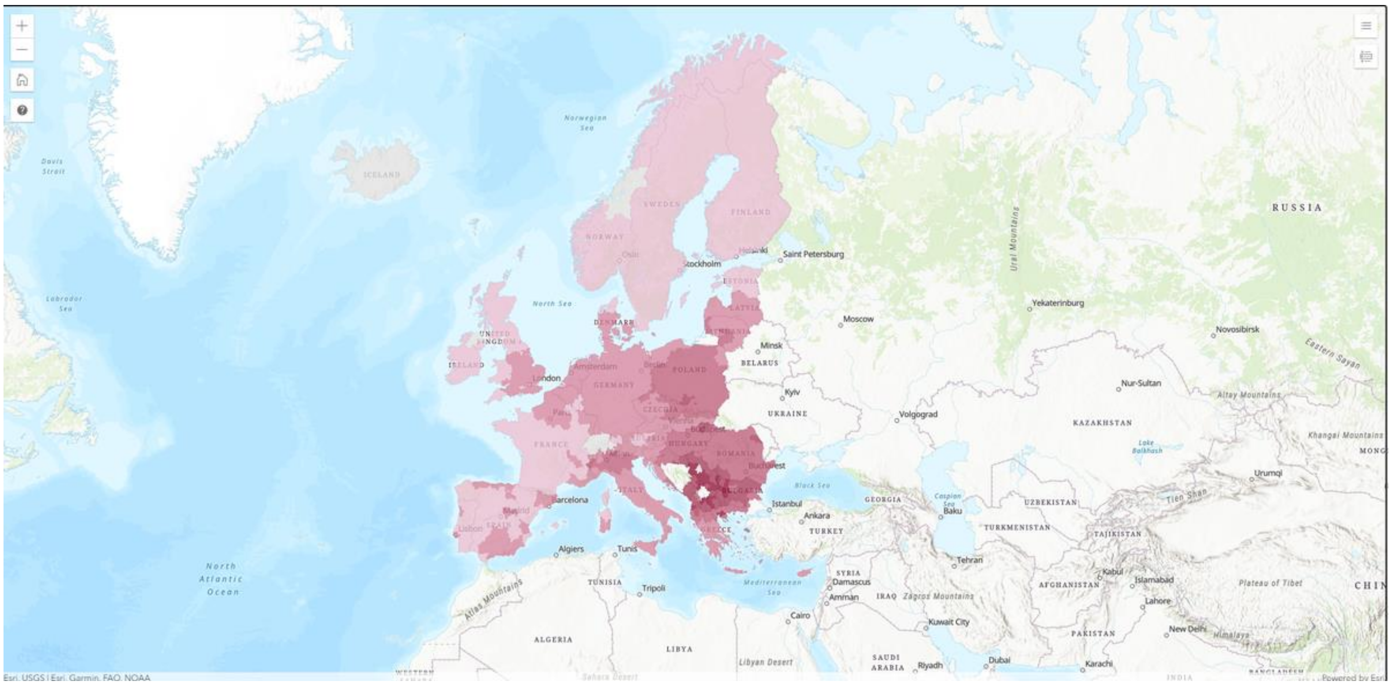


Image copyright: GDP per capita - Eurostat; Premature deaths, years of life lost and population-weighted concentrations are based on an EEA analysis of interpolated annual statistics of reported

Compared to 2020, there was a slight increase in deaths attributable to PM2.5 the increase is within the uncertainty intervals of the calculations, but it may also be partly due to higher exposure levels and the overall rise in European mortality, influenced by the COVID-19 pandemic.

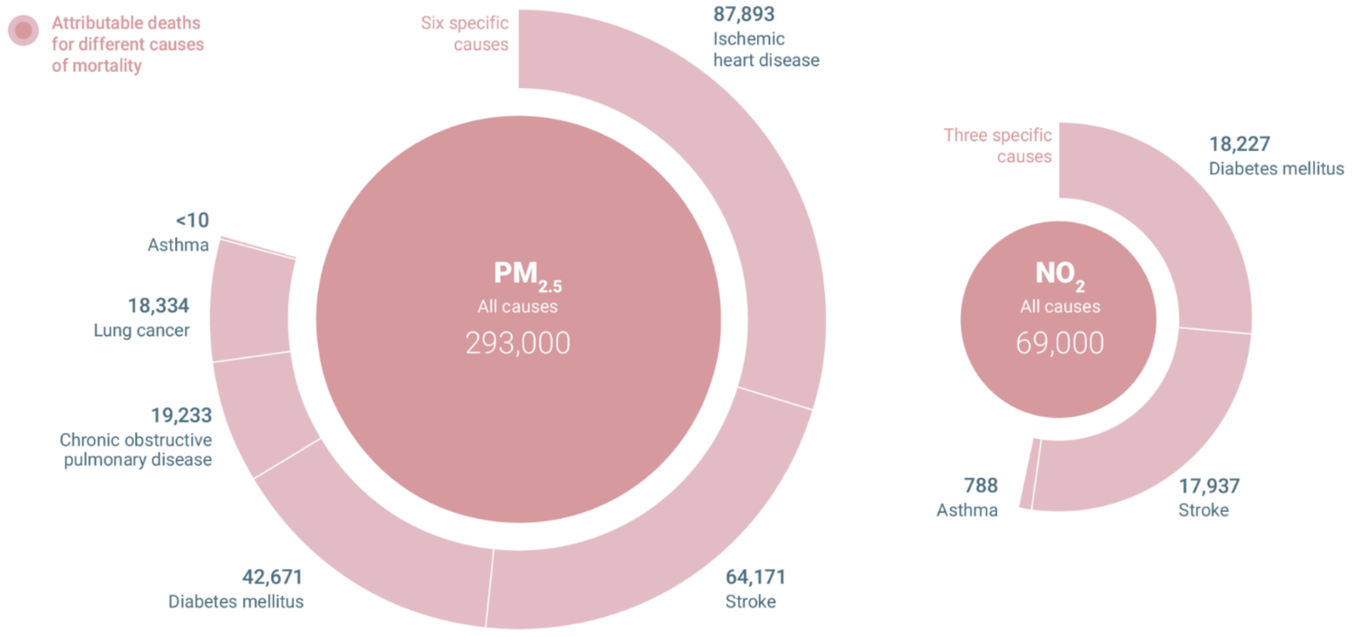
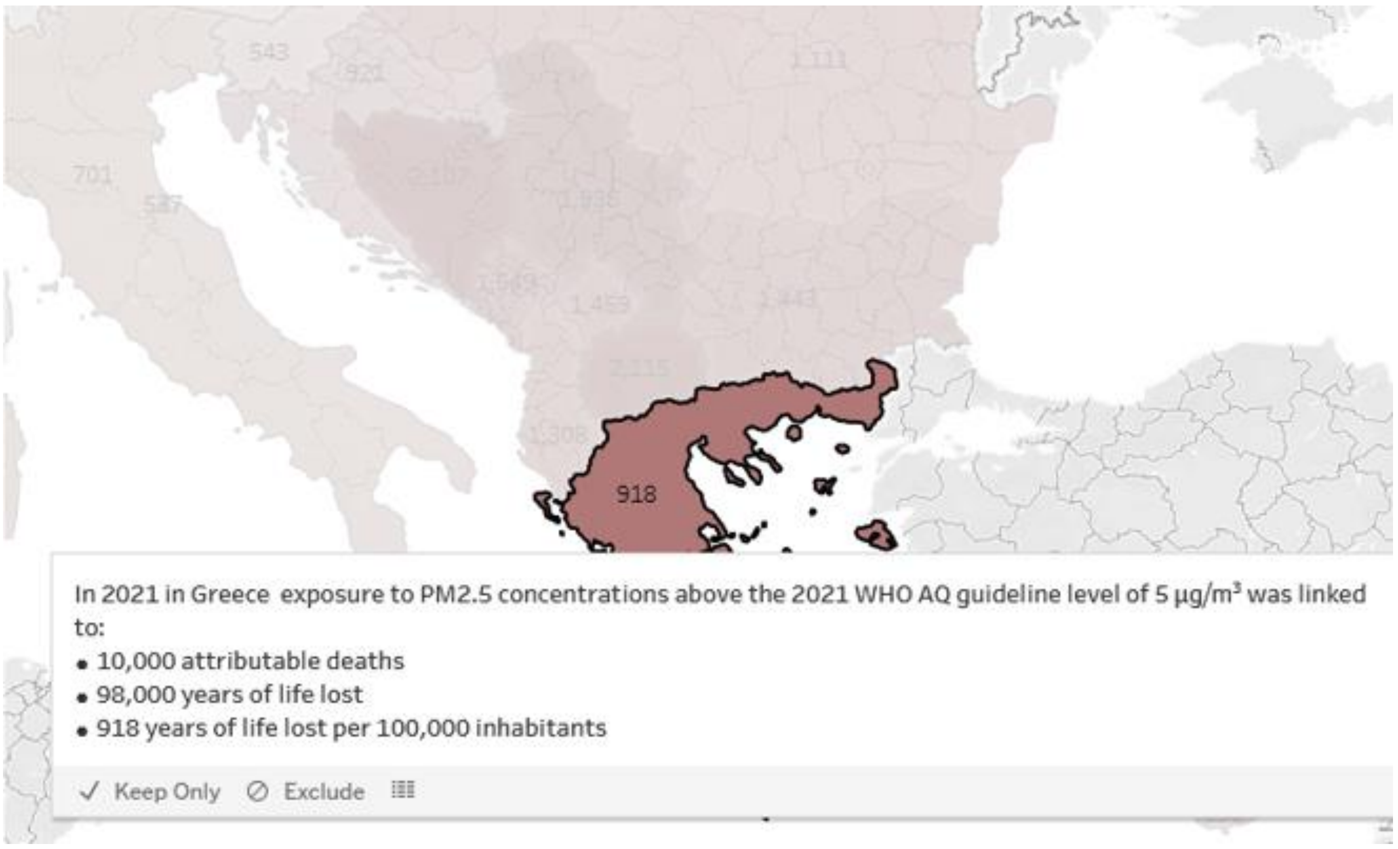


Image copyright:https://www.eea.europa.eu/publications/harm-to-human-health-from-air-pollution


In 2021 in Greece exposure to PM2.5 concentrations above the 2021 WHO AQ guideline level of 5 μg/m³ was linked to:

- 10,000 attributable deaths
- 98,000 years of life lost
- 918 years of life lost per 100,000 inhabitants

Image taken from: Interactive map: Associations between exposure to PM2.5, mortality and GDP per capita

Air pollution remains a critical health issue in Europe, contributing to hundreds of thousands of premature deaths annually. Addressing the levels of PM2.5, NO2, and O3 is vital for improving public health outcomes.

Industrial Device Design Process

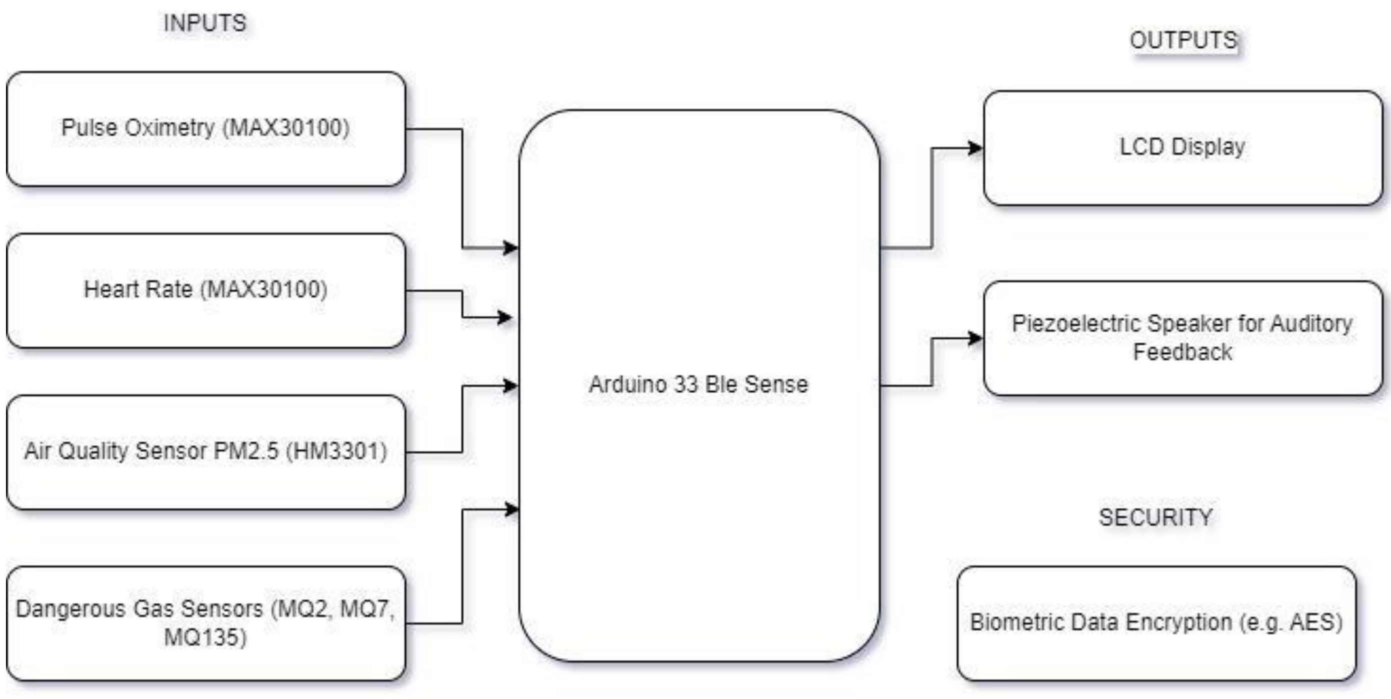
During the design of a Proof of Concept for the device, the following steps were undertaken:

1. **Development Platform**: The Arduino BLE 33 Sense was chosen for its suitability by the professor of this course and its ability to develop and run machine learning models.

2. **Hardware Design**: The focus was on selecting appropriate Arduino sensor and actuator modules, creating a PCB, and choosing components for a future commercial version.

4. **Testing**: Sensors were tested individually using an Arduino Uno before integrating them on a breadboard for final testing of the device's functionality. A simple schematic and 2-layer PCB were designed for better usability and reliability but wasn't ordered to be manufactured due to lack of time leading to the final presentation of the group projects.

Hardware Component Selection


INPUTS
Pulse Oximetry (MAX30100)
Heart Rate (MAX30100)
Air Quality Sensor PM2.5 (HM3301)
Dangerous Gas Sensors (MQ2, MQ7, MQ135)
Arduino 33 Ble Sense
OUTPUTS
LCD Display
Piezoelectric Speaker for Auditory Feedback
SECURITY
Biometric Data Encryption (e.g. AES)

Key sensors and actuators included:

- **MQ7 CO Sensor**: Detects carbon monoxide, a colorless and odorless gas, essential for indoor safety of industrial and home environments.

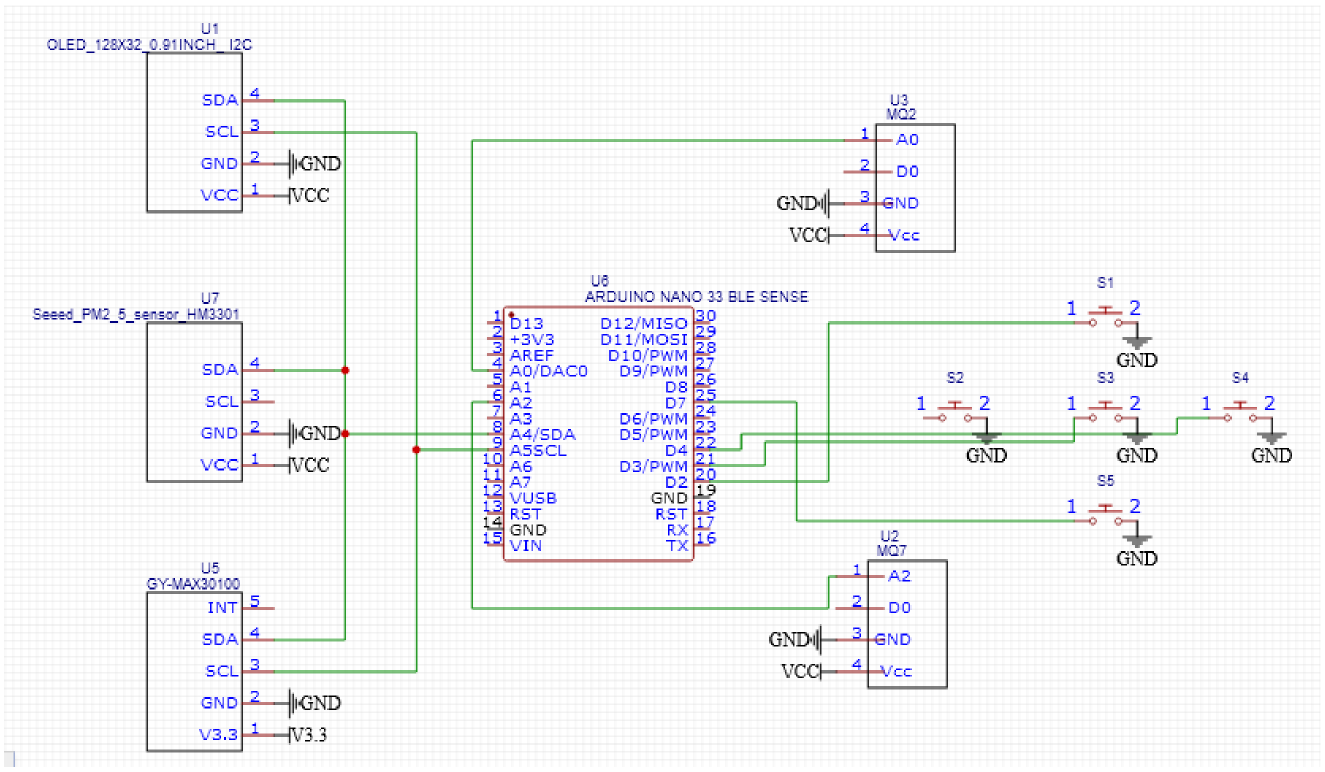
- **MQ2 Dangerous Gas Sensor**: Measures flammable gases such as LPG, butane, propane, and hydrogen, crucial for industrial and home safety.

- **MAX30100**: Monitors heart rate and blood oxygenation, important for assessing the impact of gas exposure and air pollution on human health in real time.

- **HM3301**: An air particle sensor that differentiates between PM10 and PM2.5 particles, vital for detecting air pollution levels and the particles which create that air pollution.

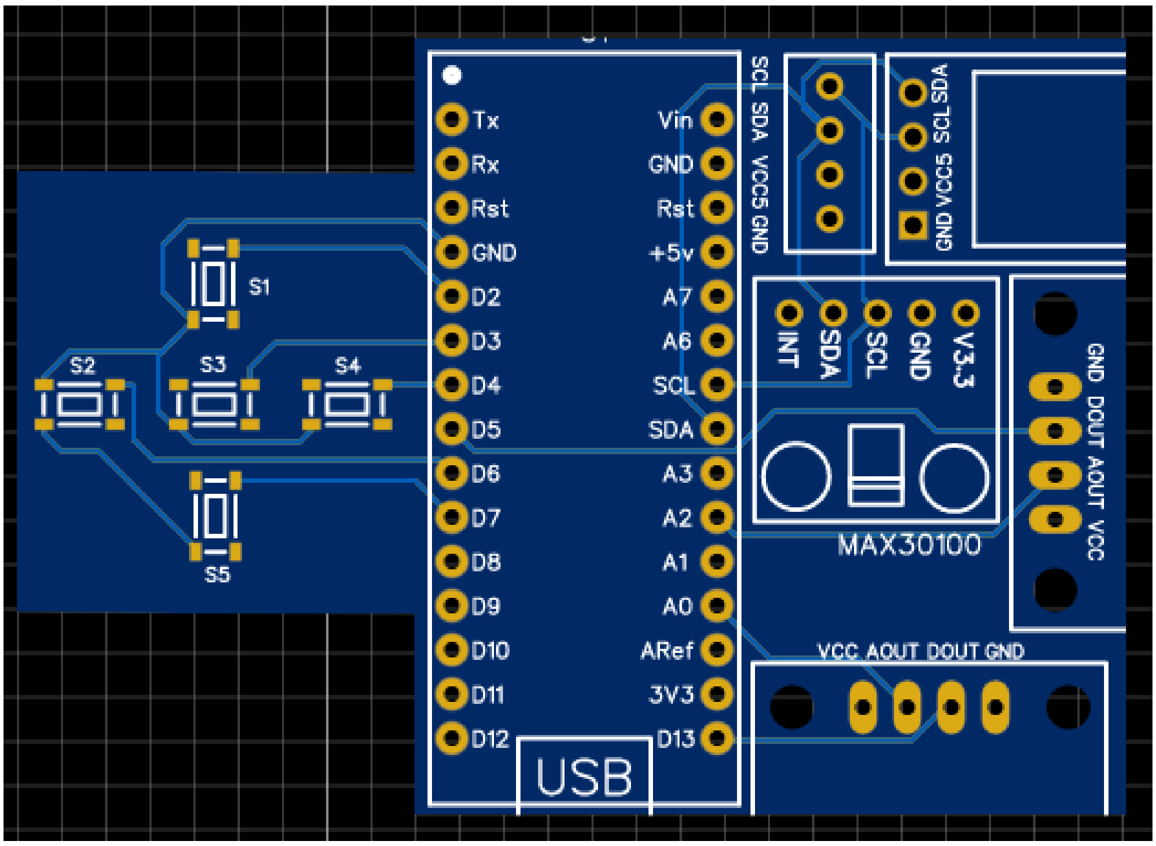
The before mentioned components were chosen to ensure the device could effectively monitor hazardous gases and air pollution, providing an important warning for industrial workers, firefighters and commercial users with respiratory issues.

PCB Shield Design: **Schematic**



A very simple schematic was created for the project, showing all the necessary connections needed for the device to work. At the left, we have all the 3.3v devices, consisting of the OLED screen, the air particle sensor and the pulse oximeter, while on the right we have the 5 button keypad, used for the navigation of the device's menus and the two analog sensors meant to provide warnings for high carbon monoxide or flammable gasses levels.

PCB Shield Design: **PCB**



The top layer or the signal layer of the PCB is shown, with the appropriate footprints of the microcontroller and all the devices being present. It should be noted that this two layer board consisting of a signal and ground layer, was created simply as a way to interact with the arduino modules in a better way than the breadboard and provide a more portable proof of concept of the device.

Bill of Materials:

Component	**Description**	**Cost (€)**
Arduino Nano 33 BLE Sense	Microcontroller	35
HM3301	Air Particle Size	30
GY-MAX30100	SpO_2, Heart Rate	5
MQ7	Monoxide carbon	3.5
MQ2	LPG, i-butane, propane	3.5
0.91inch OLED	I2C screen	6
Buzzer	Air Particle Size	0.5

Software Overview

Regarding the software implementation, the main components are the user – device interaction, the menu options, the risk determination feature, the flash memory operations and the security mechanisms concerning the password authentication and the data encryption. The device interface is user-friendly and intuitive, consisting of a small OLED screen for output and visual feedback, along with five buttons that serve for input. These buttons allow the user to enter passwords, navigate menus, and select an option. During password prompting, with the up and down buttons the user can select each character one-by-one from a wide set of available character that includes lower- and upper-case letters, numbers and special symbols.

During the first-time setup, the user is prompted to set a new password consisting of eight characters. This step ensures the device is secured against unauthorized access. Once the password is set, the device reboots and begins alternating between displaying pulse oximetry readings (bpm, SpO_2) and air quality measurements (PM2.5, MQ2, MQ7) per 5 sec (in two screen-pages). To access the main menu, the user must hold down the fifth button (menu button) at least for one second. The device then displays a "Please Wait…" message followed by a password prompt. Upon successful authentication, the user can view the ten menu options: 1. View Data, 2. Enable logging, 3. Disable logging, 4. Save data, 5. Lock data, 6. Unlock data, 7. Change password, 8. Delete records, 9. Logout, 10. Exit.

The logging functionality is crucial for ensuring data management and record keeping. The device logs five key values: bpm, SpO_2, PM2.5, MQ2, and MQ7 measurements. These five values, along with 1 byte for Risk state (Yes/No), 1 byte for Risk Cause (i.e. Heart) and 2 reserved bytes (end-of-record), all together comprise a single "record" instance (24 bytes). Logging can be performed automatically at predefined intervals or manually by the user. When "Enable Logging" is selected, the device automatically saves one record every predefined logging interval. If a risk state is detected, the device records data at (shorter) intervals specified by the risk state logging interval. The user can also manually save one record with the current sensor values by selecting the "Save Data" option. The sensor measurements can be filtered (SMA, EMA and DEMA support) for stable average readings over a predefined time window (set to 10 sec). If "Disable Logging" is selected, the last two logging mechanisms apply, not the automatic periodic first one.

The "Lock data" feature hides sensitive health data (bpm, SpO_2) from the main screen, while "Unlock data" restores the view and displays the health data as well. "Change password" allows the user to update the password, but deletes any stored records. The "Delete records" option lets the user to manually erase all the stored records from the flash memory, but keeps the credentials intact. "Logout" logs the user out by de-authentication and exits the menu, requiring re-authentication for future menu access, whereas "Exit" returns the user to the values display without logging out. Finally, the "View Data" option enables the user to view all the records saved in flash memory as decrypted and properly formatted readable text, with easy navigation through the records using the right button. The user at any time may exit the View Data submenu by pressing the menu button once.

A very important feature of essential biomedical interest is the risk determination, which identifies risk states based on sensor values consistently falling outside some predetermined thresholds. These thresholds were selected based on literature findings and are the following:

PM25_THRESHOLD	50
MQ2_THRESHOLD	160
MQ7_THRESHOLD	50
HEART_RATE_LOW_THRESHOLD	40
HEART_RATE_HIGH_THRESHOLD	100
HEART_RATE_MIN_VALID_THRESHOLD	25
SPO2_THRESHOLD	92
SPO2_MIN_VALID_THRESHOLD	60

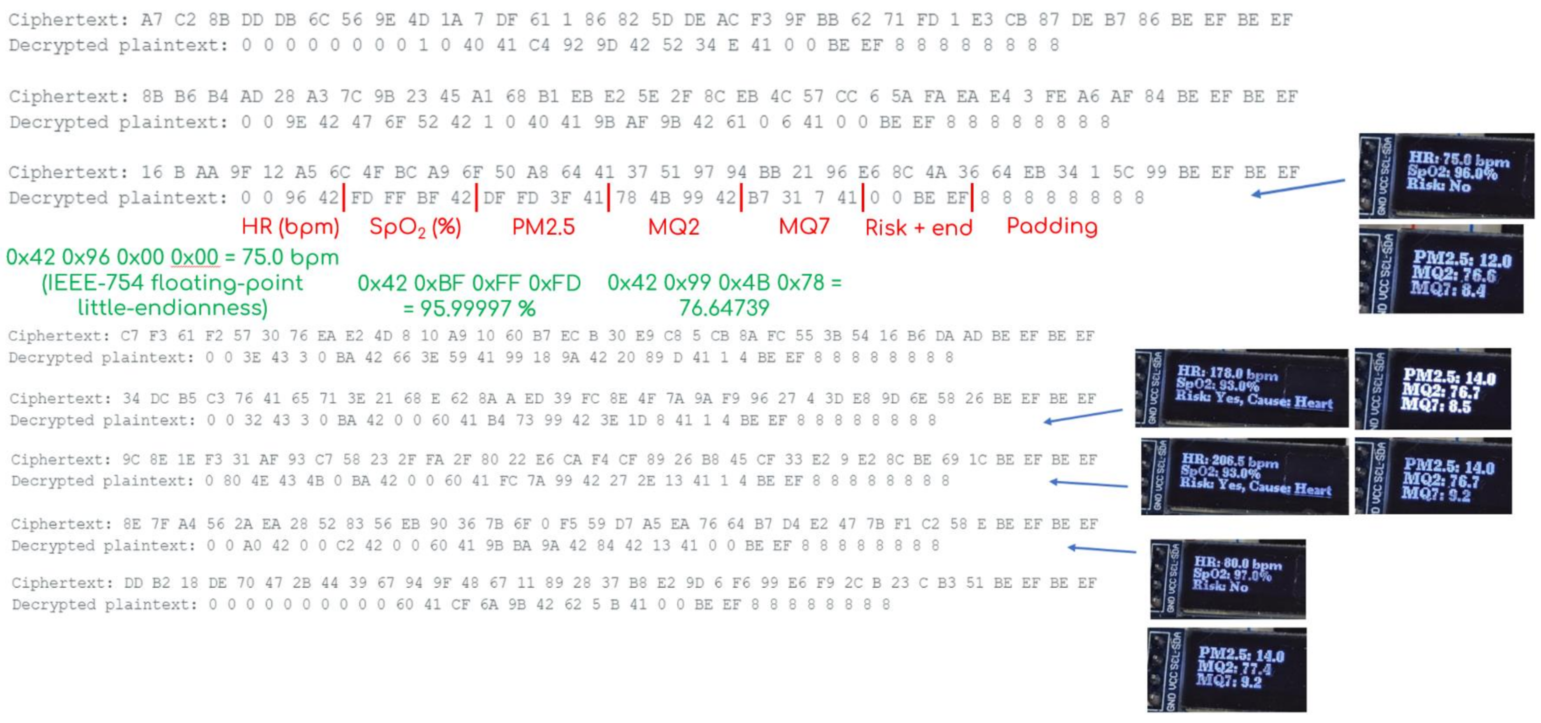
Besides the healthy range, we have also considered some validation thresholds that determine whether the measurement is valid or not to begin with, in order to be filtered and added to the averaging as final accepted value. For example, a heart rate of 0 or less than 25 bpm is not acceptable at all and is ignored, while a value of SpO_2 less than 60 is not possible even in extreme pathological conditions and thus must be discarded from filtering.

However, only few such out-of-healthy-range values may not necessarily indicate a risk state. That's why consistency is important. The sensors must measure consistently such values for a while in order to indicate a permanent risk state. A permanent risk state is triggered if more than half of the readings within a given risk window interval indicate a risk state. For example, for 100 ms READ_INTERVAL and 30 sec RISK_WINDOW → 30000/100 * 0.5 = 150 readings → 150 * 100 ms = 15 sec, which means that from the moment the first out-of-range value appears, the device will need 15 seconds to determine a permanent risk state (which corresponds to 150 readings of the sensors). When such a permanent risk state gets detected, a buzzer rings an alert after 5 sec for 2 seconds and every 5 seconds, until the risk state is resolved, while the device automatically begins logging data regularly in order to capture valuable information about the risk state (user's vitals, air quality during the risk state and cause of the risk state).

Due to ARM processor and Mbed OS, the classic AVR flash memory and AES libraries are not compatible. The "NanoBLEFlashPrefs" library was used for accessing the flash memory and for handling read/write operations, while Neil Thiessen's AES library was used for the AES operations (encryption, decryption). The user password is saved in flash memory, not as plaintext but as a hash of 32 bytes using SHA-256, derived by the combination of the raw password and a 16-bytes salt (random). The salt is also saved in flash memory, but unencrypted → 32 + 16 = 48 bytes belong to the credentials data area. When a password is entered, the device generates a new hash using the stored salt and the entered password, then compares it with the stored hash to authenticate the user. If they match, the password is correct and the authentication is successful. This is possible to do because SHA-256 is deterministic and generates always the same result output (hash) for the same inputs (raw password, salt). To enhance security, only five attempts are allowed for entering the correct password, otherwise a lockout period of one minute is enforced. During lockout, the user cannot enter any passwords or view the sensor values anymore or even enter the menu, and an on-screen countdown is displayed.

Data encryption is another crucial feature for data protection. All records are written in flash memory as encrypted data using AES-256 in CBC mode with IV, which requires a 32-byte key and a 16-byte initialization vector (IV). The password-salt hash itself serves as the key, and the last 16 bytes of this hash are used as the IV. Each raw record contains 5 float values (20 bytes) + 2 state bytes + 2 end-of-record bytes = 24 bytes. Raw data are necessarily padded to 32 bytes, since AES works on multiples of only 16-byte blocks. After padding, the data are finally encrypted and 4 end-of-record bytes ("BEEF") are appended (total of 36 bytes). The maximum number of records is set to 100 → max. 3600 bytes of encrypted data (which is less than the flash page size of 4kB). The end-of-record indicator is set to the custom "BEEF" both for raw data (once: 0xBE 0xEF) and for encrypted data (0xBE 0xEF 0xBE 0xEF, twice to avoid memory alignment issues) to easily determine the number of records.

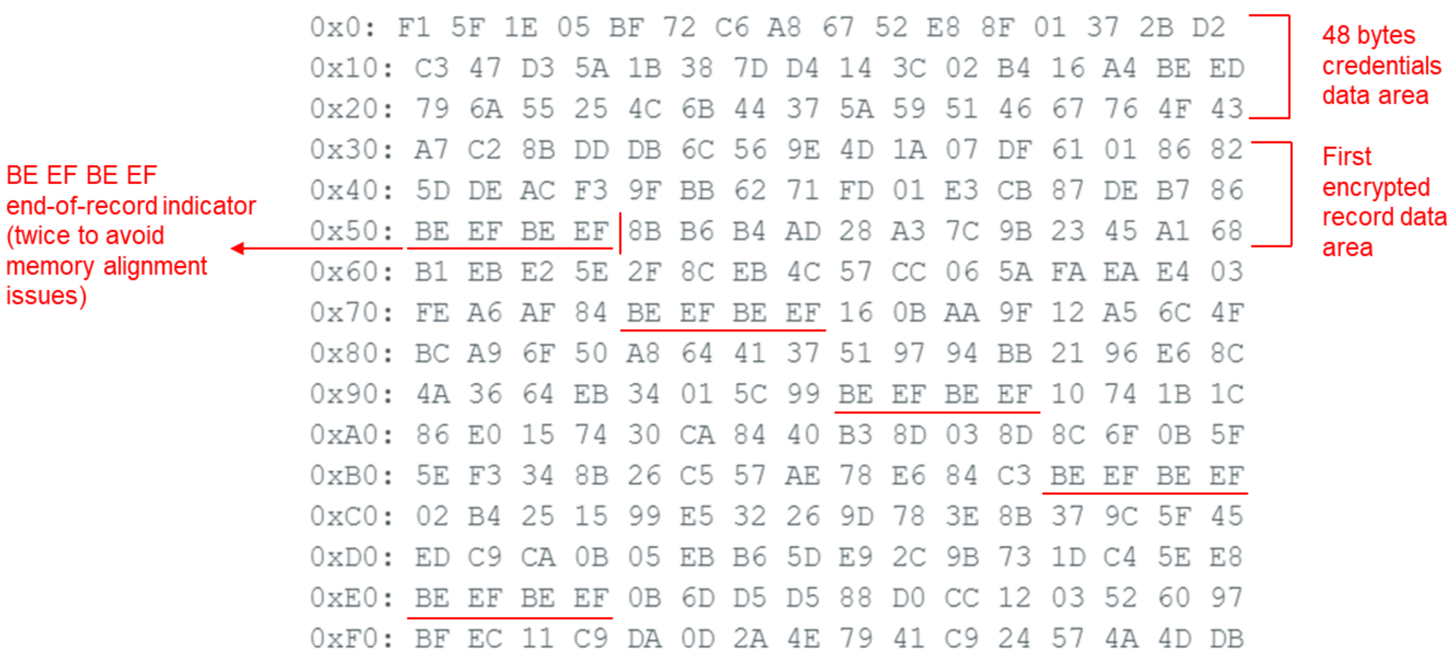
Example 1: Ciphertext vs Plaintext



```
Ciphertext: A7 C2 8B DD DB 6C 56 9E 4D 1A 7 DF 61 1 86 82 5D DE AC F3 9F BB 62 71 FD 1 E3 CB 87 DE B7 86 BE EF BE EF
Decrypted plaintext: 0 0 0 0 0 0 0 0 1 0 40 41 C4 92 9D 42 52 34 E 41 0 0 BE EF 8 8 8 8 8 8 8 8

Ciphertext: 8B B6 B4 AD 28 A3 7C 9B 23 45 A1 68 B1 EB E2 5E 2F 8C EB 4C 57 CC 6 5A FA EA E4 3 FE A6 AF 84 BE EF BE EF
Decrypted plaintext: 0 0 9E 42 47 6F 52 42 1 0 40 41 9B AF 9B 42 61 0 6 41 0 0 BE EF 8 8 8 8 8 8 8 8

Ciphertext: 16 B AA 9F 12 A5 6C 4F BC A9 6F 50 A8 64 41 37 51 97 94 BB 21 96 E6 8C 4A 36 64 EB 34 1 5C 99 BE EF BE EF
Decrypted plaintext: 0 0 96 42|FD FF BF 42|DF FD 3F 41|78 4B 99 42|B7 31 7 41|0 0 BE EF|8 8 8 8 8 8 8 8
```

```
Ciphertext: C7 F3 61 F2 57 30 76 EA E2 4D 8 10 A9 10 60 B7 EC B 30 E9 C8 5 CB 8A FC 55 3B 54 16 B6 DA AD BE EF BE EF
Decrypted plaintext: 0 0 3E 43 3 0 BA 42 66 3E 59 41 99 18 9A 42 20 89 D 41 1 4 BE EF 8 8 8 8 8 8 8 8

Ciphertext: 34 DC B5 C3 76 41 65 71 3E 21 68 E 62 8A A ED 39 FC 8E 4F 7A 9A F9 96 27 4 3D E8 9D 6E 58 26 BE EF BE EF
Decrypted plaintext: 0 0 32 43 3 0 BA 42 0 0 60 41 B4 73 99 42 3E 1D 8 41 1 4 BE EF 8 8 8 8 8 8 8 8

Ciphertext: 9C 8E 1E F3 31 AF 93 C7 58 23 2F FA 2F 80 22 E6 CA F4 CF 89 26 B8 45 CF 33 E2 9 E2 8C BE 69 1C BE EF BE EF
Decrypted plaintext: 0 80 4E 43 4B 0 BA 42 0 0 60 41 FC 7A 99 42 27 2E 13 41 1 4 BE EF 8 8 8 8 8 8 8 8

Ciphertext: 8E 7F A4 56 2A EA 28 52 83 56 EB 90 36 7B 6F 0 F5 59 D7 A5 EA 76 64 B7 D4 E2 47 7B F1 C2 58 E BE EF BE EF
Decrypted plaintext: 0 0 A0 42 0 0 C2 42 0 0 60 41 9B BA 9A 42 84 42 13 41 0 0 BE EF 8 8 8 8 8 8 8 8

Ciphertext: DD B2 18 DE 70 47 2B 44 39 67 94 9F 48 67 11 89 28 37 B8 E2 9D 6 F6 99 E6 F9 2C B 23 C B3 51 BE EF BE EF
Decrypted plaintext: 0 0 0 0 0 0 0 0 0 0 60 41 CF 6A 9B 42 62 5 B 41 0 0 BE EF 8 8 8 8 8 8 8 8
```

Example 2: Encrypted Memory Dump



```
0x0:  F1 5F 1E 05 BF 72 C6 A8 67 52 E8 8F 01 37 2B D2
0x10: C3 47 D3 5A 1B 38 7D D4 14 3C 02 B4 16 A4 BE ED
0x20: 79 6A 55 25 4C 6B 44 37 5A 59 51 46 67 76 4F 43
0x30: A7 C2 8B DD DB 6C 56 9E 4D 1A 07 DF 61 01 86 82
0x40: 5D DE AC F3 9F BB 62 71 FD 01 E3 CB 87 DE B7 86
0x50: BE EF BE EF 8B B6 B4 AD 28 A3 7C 9B 23 45 A1 68
0x60: B1 EB E2 5E 2F 8C EB 4C 57 CC 06 5A FA EA E4 03
0x70: FE A6 AF 84 BE EF BE EF 16 0B AA 9F 12 A5 6C 4F
0x80: BC A9 6F 50 A8 64 41 37 51 97 94 BB 21 96 E6 8C
0x90: 4A 36 64 EB 34 01 5C 99 BE EF BE EF 10 74 1B 1C
0xA0: 86 E0 15 74 30 CA 84 40 B3 8D 03 8D 8C 6F 0B 5F
0xB0: 5E F3 34 8B 26 C5 57 AE 78 E6 84 C3 BE EF BE EF
0xC0: 02 B4 25 15 99 E5 32 26 9D 78 3E 8B 37 9C 5F 45
0xD0: ED C9 CA 0B 05 EB B6 5D E9 2C 9B 73 1D C4 5E E8
0xE0: BE EF BE EF 0B 6D D5 D5 88 D0 CC 12 03 52 60 97
0xF0: BF EC 11 C9 DA 0D 2A 4E 79 41 C9 24 57 4A 4D DB
```

48 bytes credentials data area
First encrypted record data area
BE EF BE EF end-of-record indicator (twice to avoid memory alignment issues)

Verification – Integration – Testing

Apart from the initial hardware testing with Arduino Uno to ensure each sensor was working properly and the I2C was sending data, we ran some sanity check tests for the software as well to make sure that each major software component alone was functional and was working with the expected behavior. To achieve this, we ran some sanity tests with basic code examples for i) AES encryption/decryption, ii) password hashing, iii) password hashing with applying also AES to records data, iv) pulse-oximeter, PM2.5, MQ2, MQ7 sensors, v) OLED screen, vi) flash memory operations: erase password and write to / read from flash memory. And of course, after starting from these simple examples we concluded with integrating all these together successfully. Representative use cases (for setting first password, authentication through correct password and displaying menu options, viewing data records and risk state scenario with buzzer alarm) are demonstrated in the video files available online here (link is also included in GitHub readme file).

Problems – Challenges

During the development of our industrial device, we encountered several significant challenges that influenced our design and implementation choices. In total, the main challenges were the following:

1. **Library Support for Arduino Nano BLE 33 Sense**: One of the major hurdles we faced was the limited support for the Arduino Nano BLE 33 Sense within the existing Arduino libraries. Many libraries that we initially planned to use were either incompatible or required extensive modifications to work with this specific development board, especially the flash memory operations and AES libraries.

2. **Peripheral Integration**: Due to the constraints imposed by the limited library support, we had to carefully select peripherals that were compatible with the Arduino Nano BLE 33 Sense. Ensuring that these components operated at 3.3 volts was crucial to maintain compatibility and avoid potential issues with power regulation and communication. This required thorough testing and validation of each peripheral to ensure they met our project's requirements, which we were able to do with an Arduino uno before we were provided with the appropriate microcontroller. Besides the voltage level, for I2C communication establishment we had to use pull-up resistors to work, and in software the pulse-oximeter peripheral needs to be reinitialized whenever a delay() is used, as it seems to desynchronize the beat detection callback of the internal driver (MAX30100 library). Concerning in specific the pulse-oximetry sensor, due to its optical nature of operation, this sensor is very sensitive to ambient light scattering and reflection (for example interference due to lights or camera lens above sensor), which can result in "ghost" readings if not properly shielded/covered (this sensor shall be in contact with human skin and not exposed to ambient light).

3. **App Development Constraints**: Initially, we considered using platforms like MIT App Inventor or Android Studio to develop a companion app for our device, however, we decided against this due to the additional complexity it would introduce. Developing an app would have required significant time and expertise in android development (at least at the complexity level our device targets and requires for all the aforementioned features), which were beyond the scope of our current project, since by design we decided to create a closed system without the need for external mobile device and Bluetooth connectivity (hence the on-board OLED screen and input buttons approach). Instead, we focused on ensuring that the device itself was robust and reliable, prioritizing the core functionalities over additional features which could be added later on.

4. **PCB Design and Testing**: Creating a reliable hardware setup was another challenge, since integrating all the sensors into a breadboard proved the complexity of the connections needed and its weak ability for portability of the device. That's why, a simple schematic and a 2-layer PCB were designed to facilitate easier assembly and testing of the final prototype.

It is worth mentioning that the most time-consuming problems we faced was finding the proper libraries for flash memory operations and AES cryptography and integrating them into a single working system all together along with the rest of the code. A side problem we had was with the decryption process, which could be a result of a race condition in memory access, but this was fortunately handled and is now resolved. A major problem we had initially was developing the menu concept with only five buttons, to implement the menu transitions, the screen-pages to display, the button navigation and password-characters-entering scheme using only two buttons instead of a convenient keypad for example (we had to work with what we had available in hand and in time).

The menu concept was very exhausting to develop and initially error-prone, requiring many days of code development and debugging to be fully operational, yet still intuitive for the end-user. From the architecture-design perspective, the design choices for the implementation of the various menu options were a cumbersome process, in order to have a complete system with all the necessary features the end user might need. These useful menu options and features offer flexibility and allow the user to make the most out of the device. What we mean is that this system was developed (at least at the software level) considering the actual needs of the end user and not only the basic requirements of this assignment. The code development required almost 16 days (~192 work hours) and debugging with testing took almost another 6 days (~72 work hours), while documentation-reporting took 4 days (~32 work hours).

Future improvements

For future enhancements, we could explore the integration of LittleFS as an efficient filesystem for flash memory operations with wear leveling abilities. Besides flash memory, we should also consider the option of an external SD card storage for logging even more records at a higher rate. Furthermore, we could try more complex encryption key derivation methods such as the widely used PBKDF2-HMAC or at least other complex custom patterns rather than simply setting the key to be the hash itself and the IV to be the last 16 bytes of this hash. Moreover, we should save the number of password-entering attempts in flash memory as well, in order to handle malicious resets or power off – power on cycles that would try to circumvent the lockout protection mechanism, which would allow brute force attacks gaining illegal access to the system. In addition, a machine learning model could be deployed to determine dynamically the risk state instead of using the predetermined thresholds, which will require to collect hundreds of records data, to export them and label them manually as risk or not, to import them into a python ML/DL model to train, and then import the trained model into the system to classify the state as risk or not during runtime. All these improvements aim to enhance the device's reliability, security, and user experience. Finally, we can proceed with the PCB manufacturing and testing in real-world situations (real device testing) as a miniaturized portable-wearable device, the successor to the prototype presented in this report, as a complete end product.

Bibliography

1. https://www.eea.europa.eu/themes/air/air-quality-concentrations/air-quality-standards#:~:text=To%20assess%20how%20this%20has,particulate%20matter%20(PM10%20and%20PM2.&text=Air%20quality%20is%20assessed%20by,pollutants%20in%20the%20ambient%20air
2. https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX%3A32008L0050
3. https://eur-lex.europa.eu/eli/dir/2004/107/oj
4. https://www.eea.europa.eu/en/topics/in-depth/air-pollution/eow-it-affects-our-health
5. https://www.eea.europa.eu/publications/environmental-burden-of-cancer/air-pollution
6. https://www.eea.europa.eu/publications/unequal-exposure-and-unequal-impacts
7. https://www.eea.europa.eu/publications/harm-to-human-health-from-air-pollution/
8. https://os.mbed.com/users/neilt6/code/AES/
9. https://www.arduino.cc/reference/en/libraries/nanobleflashprefs/
10. https://github.com/Dirk-/NanoBLEFlashPrefs
11. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsdk_nrf5_v17.1.0%2Flib_fds.html
12. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf52840%2Fmemory.html
13. https://www.h-schmidt.net/FloatConverter/IEEE754.html
14. https://github.com/littlefs-project/littlefs