UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Sistem embedded pentru monitorizarea poluării aerului

Nicolae-Andrei Vasile

Coordonator științific:
Conf. Dr. Ing. Dan Stefan Tudose

BUCUREȘTI

2021

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

DIPLOMA PROJECT

Wearable Air Quality Monitor

Nicolae-Andrei Vasile

Thesis advisor:
Conf. Dr. Ing. Dan Stefan Tudose

BUCHAREST

2021

CONTENTS

SINOPSIS

Proiectul vizează monitorizarea calității aerului în vederea informării utilizatorului cu referire la nivelul de poluare la care acesta este expus, oferind date despre mediu în timp real, și este alcătuit din trei componente principale: sistem embedded, o aplicație mobilă și servicii de backend.

Sistemul urmăreste implementarea tuturor caracteristicilor unui dispozitiv embedded, fiind conceput special pentru colectarea și procesarea parametrilor aerului cât și pentru transmiterea acestora la cererea utilizatorului. Aplicația mobilă expune date primite de la dispozitivul embedded, oferind de asemenea funcționalități pentru analiza acestora. Aceasta, la rândul ei, se folosește de servicii de backend pentru accesarea si stocarea informațiilor colectate în vederea utilizărilor ulterioare.

Soluția expune toate componentele necesare utilizatorului pentru a fi corect informat despre nivelul de poluare dintr-o anumtă zonă, prezentând o structură a aplicației intuitivă și ușor de folosit.

ABSTRACT

The project aims to monitor the quality of air to inform the user about the level of pollution to which he is exposed, providing real-time environmental data, and consists of three main components: embedded system, mobile application and backend services.

The system follows the implementation of all the characteristics of an embedded device, being designed specifically for collecting and processing environmental parameters and for transmitting them upon user's request. The mobile application exposes data received from the embedded device and provides various functionalities for analysis as well. In turn, it uses backend services to access and store information collected for future use.

The solution exposes all the components needed to the user to be properly informed about the level of pollution in a certain area, presenting an intuitive and easy-to-use application structure.

1 INTRODUCTION

1.1 Background

With the technological evolution that is constantly exceeding expectations by enlarging the spectrum of possibilities and opening new opportunities, the society is relying more and more on the resulted devices. It has become common that such operational systems to be incorporated in many of the objects that people use on daily basis, such as smartphones, smart TVs, smart refrigerators, autonomous vacuum cleaners, even the concept of the intelligent homes being a topic of interest.

The current technological state allows to build complex systems for several purposes, from representing forms of entertainment, such as game stations, to autonomous robots used in production lines, has already expanded in almost every existent domain, providing new solutions to current issues, facilitating fields with necessary equipment, dealing with world pollution, and many others.

The medical domain has also adopted technological approaches in various ways. One example is represented by the many interventions nowadays conducted using technologic devices for increased precision, better solutions, and other benefits. More and more health monitoring devices also have appeared and are adopted by the public.

1.2 Problem

Air pollution is a significant problem that affects the entire world, a staggering 91% of the people breathing polluted air, which exceeds WHO (World Health Organization) limits [1]. Pollutants are a major contribution to chronic conditions, such as pulmonary and cardiovascular diseases, and are present on daily basis more in urban rather than rural environments. Such issue needs to be addressed properly for the rate of related mortality, reaching 4.2 million deaths every year [1], to slowly decrease and the constant negative effects that affect the health of the public to diminish.

1.3 Proposed Solution

A viable solution is presented to the public specifically to address the very relevant problem of the air pollution, putting accent on minimizing the exposure to polluted environment rather than treating the possible symptoms of diseases caused by it.

We propose an application that measures the air quality in a specific location by collecting and analyzing data from the environment in real time. Air pollutants such carbon and nitrogen oxides and many others can be monitored and processed by the device. Further, data can be available to the user’s smartphone almost immediately by receiving the information from the wearable device.

1.4 Paper Structure

The paper is structured in 6 main chapters, presenting the researching and development process of the proposed solution. The “Introduction” chapter is responsible with familiarizing the reader with the overall context of the paper, including the problem, solutions, and the background of the project. It is followed by the “Requirements Analysis and Specifications” chapter, where we discuss and clarify the expectations, as well as the requirements of the project. The “Market Study / Existent Approaches” describes the present state of the market and the already implemented solutions related to the stated problem. Having the results of the previously mentioned chapters, we describe in more detail the proposed solution and implementation in the “Proposed solution / Implementation details” chapter. Further, after having implementation ready, a results evaluation and case study is desired to check the correctness and the general opinion regarding the solution. The “Case Study / Results Evaluation” chapter provides a detailed description of the mentioned points. We end by summarizing all the components of the paper described prior, such as the implementation of the requirements, the structure of the solution, testing process, in the “Conclusions” chapter. Future work is also presented in the last chapter.

2 REQUIREMENT ANALYSIS AND SPECIFICATIONS

It is required to know from the very beginning what are the expectations towards the project for important reasons, such as clarifying what are the conditions, what functionalities it needs to contain, as well as providing a plan for the development process accordingly, to pursue the optimal implementation. Therefore, a requirements analysis is in place to set the proper goals and to provide a list of functionalities that will define exactly the behavior of the project.

2.1 Analysis

There are many variables that must be considered for the structure to be defined, taking in consideration the user perspective, from physical to cognitive and emotional factors [2]. As it is critical to sustain the desire of the user towards the application through design and ease of interaction, is it required for the project to implement characteristics such as simplicity, reliability, resistance, responsiveness, mobility and others, that can influence the user acceptance and engagement with the application on a positive manner [3]. A diagram of the described requirements is presented below.

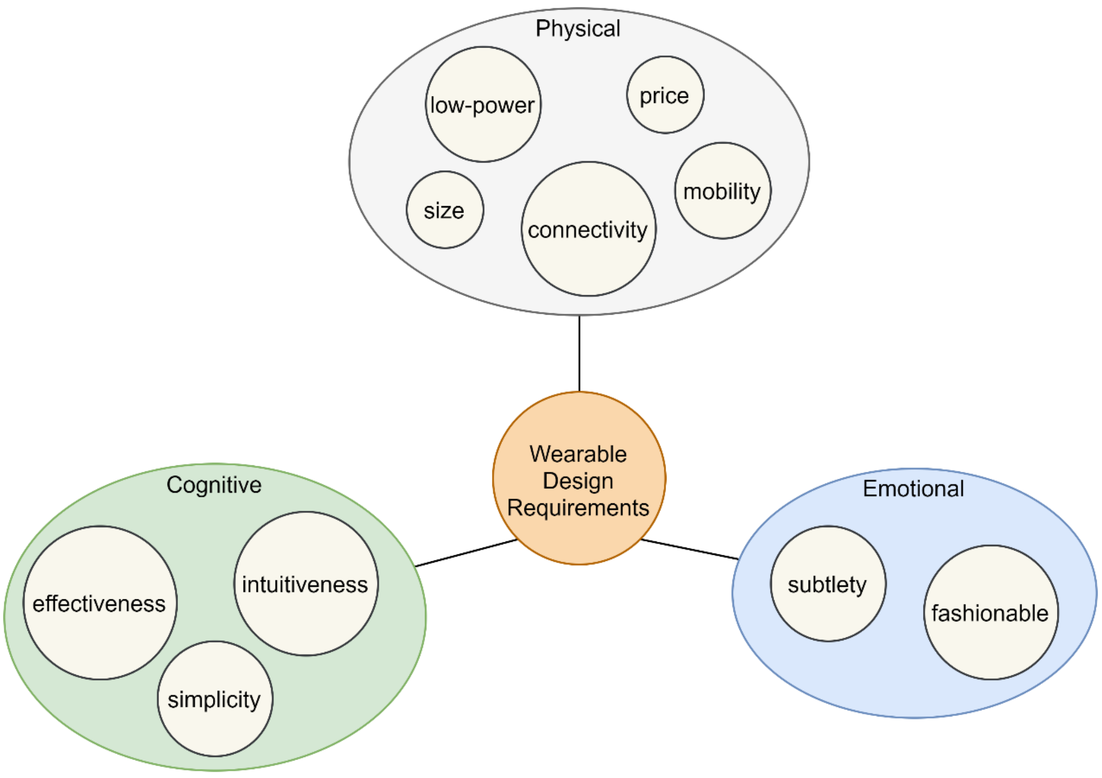


Figure 2.1 Wearable Design Requirements [3]

A fundamental challenge in designing a wireless device is represented by the physical limitations and available solutions in the development process [2]. Looking at the requirements of a sensor-based wireless device, there are several parameters that need to be considered, such as cost and size of the application, resources, mobility of the device, connectivity, communication, lifetime of the power supply and others [2], [4]. For that matter, the wearable device takes precedence above other possible components in the development process, for instance visualization tools or data storage.

The mobility is an essential aspect of a wearable device [2], ensuring that the equipment can be utilized independently from a stationary power source, such as an USB cable, and adds a dynamic aspect to the application [4]. Although it is of great interest to implement such capability in wearable devices, the power consumption becomes another big criterion in choosing the components for an IoT implementation [5], [6]. Many of such devices are powered by batteries and changing the energy source or charging the equipment all the time can decrease the interest in the device [4]. Therefore, it is desired that the power consumption of the circuit be minimal.

The connectivity aspect is yet another fundamental part as it enables different components to communicate, and the choice of protocol for the implementation can vary from application to application, depending on the requirements. For example, a Bluetooth connection is better equipped to sustain close-range communication, not exceeding 100m, between portable equipment that rely on low energy consumption [6], whereas Wi-Fi is designed for long-range connections, consuming a considerable amount of energy, therefore, powerful power supplies are required [7], [8].

Size and price also must be considered as they play an important role in the engagement of possible users with the application. Although the two parameters can vary according to the nature of the project, increased sizes can affect the maneuverability for the wearable, or can bring discomfort to the user [3]. Also, devices are desired to be affordable in accordance with the available functionalities and performance.

The cognitive and emotional ergonomics follow the previously discussed field and are better suited for the component that interacts mostly with the user, that being the visualization and control tool. Multiple options are available, such as OLED screens embedded into the wearable [9], computers or smartphones. Although an OLED display would fulfil the minimal requirements of the solution, it provides a limited interface for the user to work with, as well as affecting the overall size of the device and the PCB (Printed Circuit Board) complexity. A smartphone application is better suited in this situation, being able to easily connect and control the device and bringing more comfort and simplicity to the project [10].

Therefore, we conclude the structure of the project is composed of two physical parts: a wearable device that measures the data and redirects it for visualization and a smartphone that implements the required application for display and access to data storage, as well as device control.

The wearable device is the main component of the project, as the solution relies mainly on the air quality data collected from the environment. It is required to ensure that the measurement of the environmental data, the processing and the communication capabilities are performing optimally. To pursue the best performances for our application, an efficient, low-power, yet powerful microcontroller is required as the main processing unit for the device, as well as the other relying components, keeping in mind that the size of the packages and the price play an important role as well.

The smartphone application is the secondary component of the project, and it is needed for data visualization and wearable control, as well as providing access to data storage for future statistics. It is required to implement all the necessary parameters discussed in this chapter, and alongside the measuring component, to provide an optimal and friendly interaction support between the user and the environment.

2.2 Conclusion

There are multiple factors that can represent challenges in designing a sensor-based wireless application, and can take various forms, for instance hardware limitations created by size restrictions, price or capabilities of the underlying components, or requirements from the perspective of potential users and more. They also can influence the interest of the user regarding the project. Therefore, they must be considered when designing and implementing the solution.

The structure of the project is defined with the previously described requirements considered, containing a wearable device for measuring and processing data to redirect it afterwards to the visualization component, and the smartphone application, being also a control tool for the device and providing access to a data storage. All the other components are built around the wearable, as it represents the only source of environmental data and optimizing its specifications and capabilities in regards with the solution idea would maximize the performance of the overall project.

3 MARKET STUDY / EXISTENT APPROACHES

Many examples of air quality monitoring are available, having different approaches, but mainly relying on sensor-based IoT wireless gadgets connecting to different devices such as smartphones, tablets [10], [11], or computers [7], [12]. As the main part of the solution is the wearable device, we aim to focus our attention on the processing unit, the microcontroller, and to further conduct a market study to find the optimal option to fulfil the previously mentioned requirements.

3.1 State of the Art

Table 3.1 provides a comparative analysis between four microcontrollers frequently used in IoT applications based on their specifications, such as CPU and memory details, connectivity capabilities and I/O protocols. Please note that there are many other options available on the market, but due to insufficient functionalities, large modules, or the increased price, they are not selected in this situation.

Table 3.1 Microcontrollers for IoT applications [5]

Chip	ESP32 [13]	ESP8266 [14]	CC3200 [15]	ATmega328 [16]
Price				
Details:				
Packages	QFN 6x6/5x5	QFN 5x5	QFN 9x9	MA, PN
CPU	1/2 x Xtensa® 32-bit LX6	1 x Tensilica L106 32-bit	1 x ARM Cortex-M4 32-bit	1 x AVR 8-bit
RAM	520 KB	160 KB	256 KB	2 KB
Flash	Up to 16 MB	Up to 16 MB	16 MB	32 KB (in-system)
Operating Voltage	3.3V (2.3V - 3.6V)	3.3V (2.5V - 3.6V)	3.3V (2.1V – 3.6V)	2.7V – 5.5V
Connectivity:				
Wi-Fi	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	-
Bluetooth	v4.2 BR/EDR + BLE	-	-	-
I/O:				
UART	3	2	2	1
GPIO	34	17	27	23
I2C	2	1	1	1
SPI	4	2	1	2
PWM	8	4	4	6
ADC	18 (12-bit)	1 (10-bit)	4 (12-bit)	6 (10-bit)
DAC	2 (8-bit)	-	-	1 (10-bit)

Further, more detailed explanations are required for the microcontroller of choice, considering technical details, communication capabilities, size, price and available software.

As discussed in chapter 2, an important characteristic is the dimensions of the available packages it comes into [5], [6]. As the trend for IoT devices is to become smaller, size restrictions play an important role in choosing the right chip for an IoT application. The SimpleLink CC3200 [15] is a high-performance Wi-Fi certified SoC manufactured by Texas Instruments. According to the datasheet [15], the available package for this chip is QFN, measuring 9mm x 9mm, and it can be classified as one of the bigger chips on the market, occupying a considerable area on the board in comparison with other available microcontrollers, such as the ATmega328P or ESP family.

An alternative to the SimpleLink CC3200 solution from Texas Instruments is the ESP8266 [14] and ESP32 [13] manufactured by Espressif Systems. They are low-cost, low-energy powerful SoCs that implement all the necessary tools for a wide variety of applications, such as wearable electronics, smart appliances, sensor hubs, and many others. Although many similarities can be seen between the microcontrollers, the ESP8266 and ESP32 packages, such as QFN 5x5 measuring just 5mm x 5mm and QFN 6x6 measuring 6mm x 6mm, are considerably smaller, making the alternatives eligible for smaller IoT applications.

As the communication between devices is done mainly in a wireless manner, it is essential for the chip to have RF capabilities for receiving and transmitting data and provide the implementation of standard communication protocols, for instance IEEE 802.11 (WLAN), IEEE 802.15 (Bluetooth), and others, the connectivity aspect of IoT applications being essential [4], [6], [10].

For example, ATmega328P [16] is a SoC (System-on-Chip) manufactured by Atmel and serves as the microcontroller for the Arduino Uno development board. Although its small package sizes, measuring 6.8mm x 3.3mm, technical specifications and low power consumption characteristics make the chip very appealing for IoT applications, the lack of RF capabilities is a big downside for this chip. Additional modules are required, besides ATmega328P, to implement the communication functionalities, increasing the size and the total price of the board and complicating the layout.

The SimpleLink CC3200 and ESP8266 provide a solution to this issue, the two microcontrollers having RF capabilities and implementing 802.11 b/g/n Wi-Fi functionalities. Although the two chips contain the required hardware and can facilitate connections between IoT devices without any additional IC (integrated chip), they are limited by the single available connection type, the IEEE 802.11 (WLAN). Given a situation where another protocol is better suited, such as IEEE 802.15 (Bluetooth), neither of the previously mentioned options can provide the required hardware support and implementation.

The ESP32 family is an integrated low-power, high-performance SoC designed by Espressif Systems as an improvement to its predecessor, the ESP8266 family, and presents an alternative to the communication constraints. As the chip implements both Wi-Fi (802.11

b/g/n) and Bluetooth (v4.2 BR/EDR + BLE) functionalities, it is one of the more versatile options on the market.

For minimal energy consumption, many SoCs implement different power modes and other functionalities to minimize the current draw from the source. All the microcontrollers presented in Table 3.1 contain low-power functionalities, such as clock gating and multiple low-power modes. For example, the ESP32 and ESP8266 implement five modes, reducing power consumption by turning off or pausing parts of the chip, such as RF module, CPU, and others. The SimpleLink CC3200[1] implements four of such power modes in the same manner as presented above.

To correctly pursue an IoT implementation for a specific microcontroller, the software support and the available frameworks are essential. Espressif Systems provides multiple options for embedded software development on ESP chips, such as ESP-IDF development framework [17] and Arduino modules [18], using C and C++ as programming languages. Texas Instruments provides its own Software Development Kit (SDK)[2] for the CC3200 microcontroller, having many support offerings, as well as the Arduino Software IDE (Integrated Development Environment)[3] designed for embedded development on Arduino boards. All the available software support includes the necessary tools for flashing the device, detailed documentation, and code examples.

Analyzing the data for each of the presented options, we can conclude that the ESP32 microcontroller is the obvious choice in this situation. The low-power, high-performance and communication characteristics, the size, price, and variety of available packages, alongside the software support and development frameworks, make the chip the perfect choice for this application.

3.2 Related work

Even though the ESP32 family has been released in August 2016, a considerable amount of related work is featuring the microcontroller, using it in a variety of IoT applications, development boards, and modules.

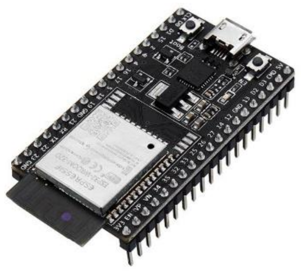
One of the examples of usage of the ESP32 is the ESP32-WROOM-32[4] module manufactured by Espressif Systems, having at its core the ESP32-D0WDQ6 [13] chip. It contains a variety of I/O interfaces, such as SPI, UART, I^{2}C, PWM, GPIO, ADC and DAC. It also

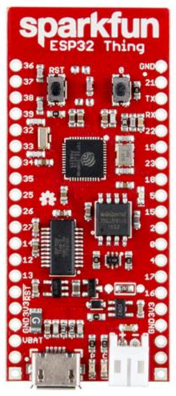
[1] B. Gilboa, "SimpleLink™ CC3100/CC3200 Wi-Fi Internet-on-a-chip™ Networking Sub-system Power Management", Texas Instruments, Appl. Report. SWRA462, Sept. 2014
[2] "CC3200SDK, SimpleLink Wi-Fi CC3200 Software Development Kit (SDK)", Texas Instruments Incorporated, [Online]. Available: https://www.ti.com/tool/CC3200SDK [Accessed: May. 20, 2021].
[3] "Arduino Software (IDE)", Arduino, [Online]. Available: https://www.arduino.cc/en/software [Accessed: May 20, 2021].
[4] Espressif Systems, "ESP32-WROOM-32", ESP32-WROOM-32 datasheet, Aug. 2016 [Revised Feb. 2021].

integrates a 40MHz crystal oscillator, and with the help of the 4MB SPI flash memory, it can be programmed accordingly. The ESP32-WROOM-32 is the main module of the entry-level development board ESP32-DevKitC[5], containing I/O pins, a micro-USB port, an USB-to-UART bridge, RESET and EN buttons also.

Another example is the ESP32 Thing[6] manufactured by SparkFun, designed specifically for building IoT projects. It is a developing platform for the ESP32 chip, equipping it with all the necessary functionalities, to support and run IoT applications. According to the technical documentation, it implements various components, such integrated 26MHz crystal oscillator, integrated SPI flash memory, micro-USB port, USB-to-UART FS231X integrated chip, auto-reset additional circuitry for the flashing sequence, and a LiPo charger that enables the board to be powered by batteries.

The image below shows the two presented ESP32-based development boards.





(1) (2)

Figure 3.1 (1) ESP32-DevKitC, (2) SparkFun ESP32 Thing

Looking towards more specific use cases of the ESP32 chips, it can be observed that there are many scientific papers that document various IoT implementations, proving the versatility of the microcontroller [6], [9].

One of such papers [9] describes how different platforms using ESP32 can be used for data processing and gives practical examples and explanations for the chosen implementation. With the usage of IoT sensors, the microcontroller can measure and

[5] "ESP32-DevKitC", ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD, [Online]. Available: https://www.espressif.com/en/products/devkits/esp32-devkitc/overview [Accessed: June 5, 2021].
[6] "SparkFun ESP32 Thing", SparkFun Electronics ®, [Online]. Available: https://www.sparkfun.com/products/13907 [Accessed: June 5, 2021]

process environmental data, and redirect the results towards a specific component for visualization, such as an OLED Display connected via I^{2}C, or a smartphone application. There are also ESP32-based development boards, that integrate such OLED displays, for instance ESP-WROVER-KIT[7].

Although the ESP32 microcontroller is vastly used in the IoT industry, it can be used outside of the mentioned domain also [6]. An implementation of a "smartphone-based oscilloscope" that intends to use the ESP-WROOM-32 module as the hardware support and a smartphone application for data display and control unit can be achieved, the ADC capabilities of the chip measuring and processing the input data accordingly.

Aside from the scientific contributions regarding ESP32, many books about the microcontroller are available as well, translated in multiple languages, such as English, German, Chinese, Russian and others, according to the official site[8]. They serve as a fundamental introduction[9] or a more application-focused support[10] to the ESP32 development process, targeting a broad audience, from passionate beginners, who want to learn more about the ESP32 chip, to advanced users, with experience in other platforms such as Arduino, Raspberry Pi, or even ESP8266. They also contain multiple code examples and explanations for each part, and step-by-step guides for designing entire applications. There are many options to choose from regarding the ESP32 related books, from a general overview of the development world of the chip to a more specific domain.

3.3 Conclusion

Based on the comparative analysis of the available market options for the device MCU and considering the various types of related work regarding the ESP32 family, we can safely conclude that the ESP32 microcontroller is the perfect choice in this situation. The communication capabilities, low-cost, low-power and high-performance characteristics, as well as the competitive size and price of the packages, bring versatility and robustness to a wide variety of applications. Moreover, the software support, development frameworks and modules, containing all the necessary tools, and the available documentation make the development process easier for programming and running embedded applications.

[7] "ESP-WROVER-KIT", ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD, [Online]. Available: https://www.espressif.com/en/products/hardware/esp-wrover-kit/overview [Accessed: May 21, 2021].
[8] Espressif Systems, "Books", ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. Available: https://www.espressif.com/en/ecosystem/iot-college/books [Accessed: May 21, 2021].
[9] A. Kurniawan, "Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32", Packt Publishing, 2019.
[10] A. McDivitt, "ESP32 Wi-Fi Scanner in Arduino IDE on Windows 10", Access Resource Group, 2019.

4 PROPOSED SOLUTION / IMPLEMENTATION DETAILS

In this chapter we propose an application that measures the air quality in a specific location by collecting and analyzing data from the environment in real time. Air pollutants such as carbon and nitrogen oxides and many others can be monitored and processed by the device. Further, the required data can be available to the user's smartphone almost immediately by receiving the information from the wearable device.

The project contains 3 main parts:

- Wearable device
- Smartphone application
- Backend service for data storage

Each of the above-mentioned components is further described in a more detailed manner.

4.1 Wearable Device

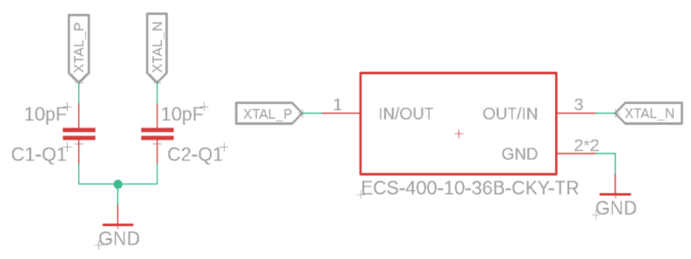
The wearable device is the component that interacts directly with the environment, collecting data through the available sensors and transmitting it to the smartphone application for visualization. The device is structured in various modules, containing multiple components, to assure that the required functionalities are correctly implemented. The mentioned modules are further discussed below, separately.

4.1.1 Microcontroller

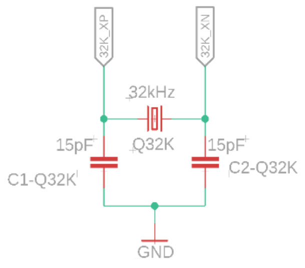
The ESP32-U4WDH chip is selected as the main processing unit of the board, based on the technical analysis of the available market options concluded in chapter 3. It is the central component of the wearable device. Therefore, the auxiliary modules are built according to its specifications.

The CPU of the SoC requires different clock inputs for the available power modes. For example, a high-frequency clock is needed when the chip is running in normal power consumption parameters, such as an external crystal oscillator or the 8MHz internal oscillator of the chip . On the other hand, an external 32kHz crystal oscillator or the low-frequency internal clocks are required for the low power modes [13].

As the WiFi/Bluetooth features of the ESP32 support only a 40MHz clock [13], a crystal oscillator of the same value is chosen for the external CPU clock source. A capacitor is added on each of the crystal's I/O pins and tied to GND. The value of these capacitors are determined by the datasheet, being 10pF each. The capacitors are placed as close as possible to their assigned pins of the crystal.



(1)



(2)

Figure 4.1 (1) 40MHz External Crystal Oscillator,

(2) 32kHz External Crystal Oscillator

The crystal oscillator is a sensitive component, therefore, it is placed far from the clock pins of the chips, at a distance of 3.5mm, and a strong guard of ground vias is added around it to maintain the high-frequency clock signal. Also, no routes are traced below the crystal to avoid unwanted interferences between the top and bottom planes [19].

A 32kHz crystal oscillator is also added to enable the possibility of integrating low-power modes for the application. A capacitor is placed at each pin of the crystal, as close to it on the board. The value for each capacitor is determined by the recommendations of the datasheet.

Further, additional circuitry is needed to optimize the power consumption in deep-sleep mode. An external capacitor, with a value of 10nF, and a RC circuit are required on the CAP1, respectively CAP2 pins [13], [19]. If the mentioned components are not implemented, the power consumption in deep-sleep mode is increased. Therefore, to have the availability of the optimized deep-sleep mode, they are added to the ESP32 schematic.

Another essential functionality for this application is the RF (radio-frequency) capability of ESP32. An antenna is required to transmit and receive the radio signals from other devices, Groundtrace 15.2mm and U.FL type being used for the specified task. The figure below presents the schematic for the antenna in this application:

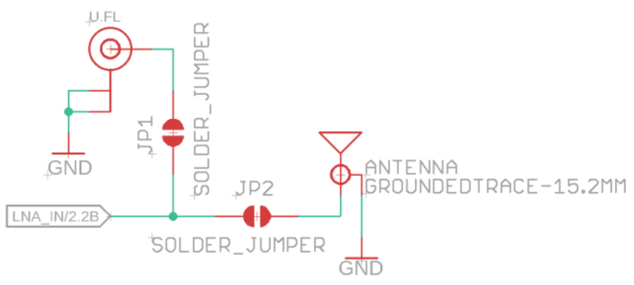
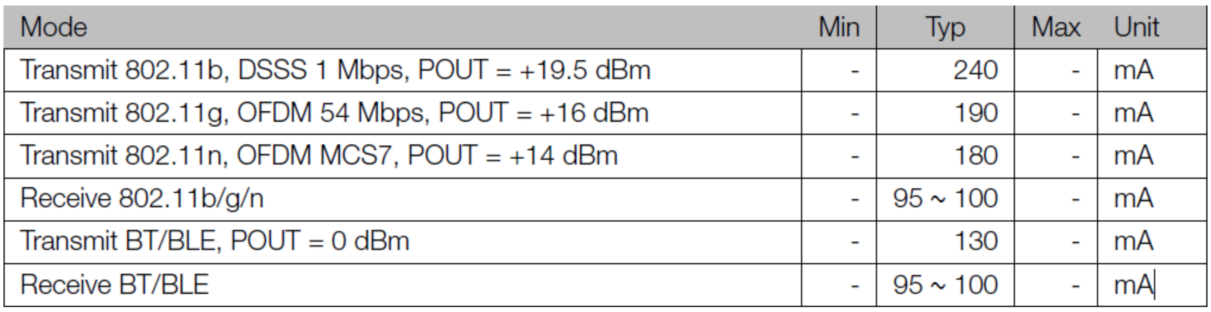


Figure 4.2 Antenna

A strong guard of ground vias is added between the groundtrace antenna and the rest of the board to achieve minimal interference between them. The RF trace is also guarded by ground vias on each side and placed far from the 40MHz crystal clock to avoid the antenna to externally couple with the mentioned oscillator. Also, no high-frequency routes from the bottom plane are crossed with the RF trace [19].

The chip must be powered accordingly by the power supply to work as intended, therefore, the necessary input current for the microcontroller must be determined. The below image shows the consumption of the active mode of the chip. It has the highest battery-draining parameters, having all the built-in functionalities of the microcontroller enabled [13].



Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

Figure 4.3 RF Power Consumption Specifications [13]

We can observe that the Wi-Fi module is the highest battery consumer, while the Bluetooth Classic/LE draws around 100~130mA. As the power consumption is an important feature for this project, the application communicates via Bluetooth LE with other devices. Therefore, the Wi-Fi component of the microcontroller is not used, to increase the power source life.

Knowing the average current consumption of the functionalities of ESP32 that are going to be implemented in the application, an estimate of the output current limit of the power module can be chosen. A value of 0.3A covers the power consumption use cases of the application and is selected as the upper limit for this implementation and will be used in further calculations.

4.1.2 Power Module

The power module is required to feed the board with continuous 3.3V voltage and a minimum 300mA. The board is also designed to be able to support a 1.5V AAA battery or an micro-B USB cable as the energy source.

The synchronous boost converter TPS61025 is used to ensure proper alimentation, as it has the specifications to fulfil the power requirements device [20]. The recommended circuit for the TPS61025 part is used in this application, the schematic for this module being presented in the following image.

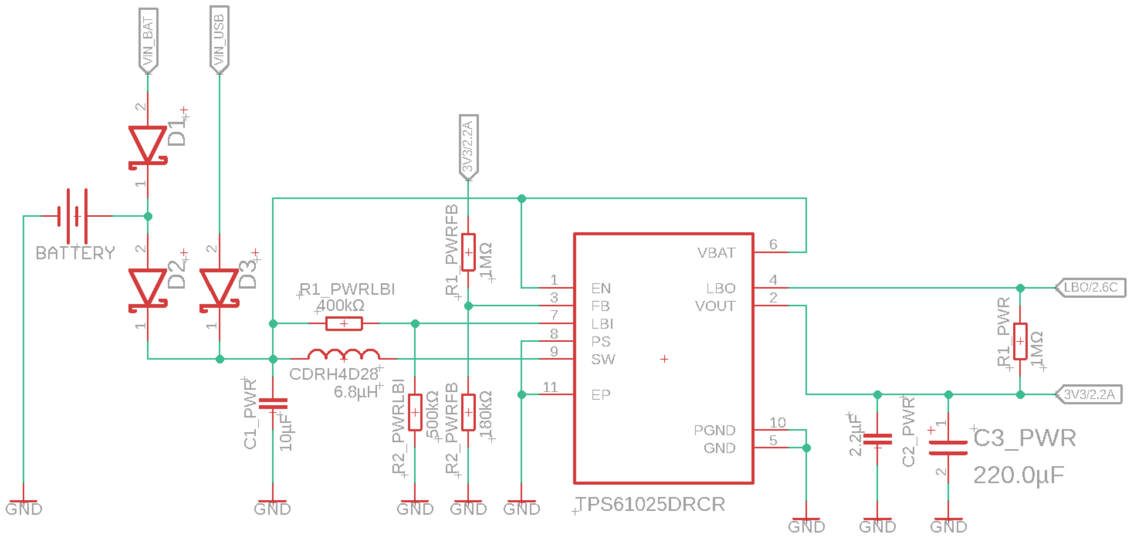


Figure 4.4 Synchronous boost converter

Further, the values of the used components need to be determined. The input noise-filtering capacitor C_1 is set to 10µF, as per datasheet recommendation. As the values for the R_4 and R_2 are given (180kΩ, respectively 500kΩ) [20], as well as the LBI threshold, the other resistor values for each voltage divider of the power module, R_1 and R_3, each of them connected to the LBI, respectively FB pins, can be calculated, as seen in (1) and (2) [20].

$$R_3 = R_4 \times \left(\frac{V_O}{V_{FB}} - 1\right) = 180k\Omega \times \left(\frac{3.3V}{0.5V} - 1\right) = \ 1M\Omega \tag{1}$$

$$R_1 = R_2 \times \left(\frac{V_{BAT}}{V_{LBI-threshold}} - 1\right) = 500k\Omega \times \left(\frac{0.9V}{0.5V} - 1\right) = 400k\Omega \tag{2}$$

The inductance value is calculated to be able to choose the inductor accordingly. Prior to (4) [21], which computes the value of the inductor, (3) [21] determines the inductor current limit based on the output current, the output voltage, the input voltage, and the efficiency of the boost converter, for the application to function properly. The below graph presents the efficiency parameter of the boost converter regarding the output current.

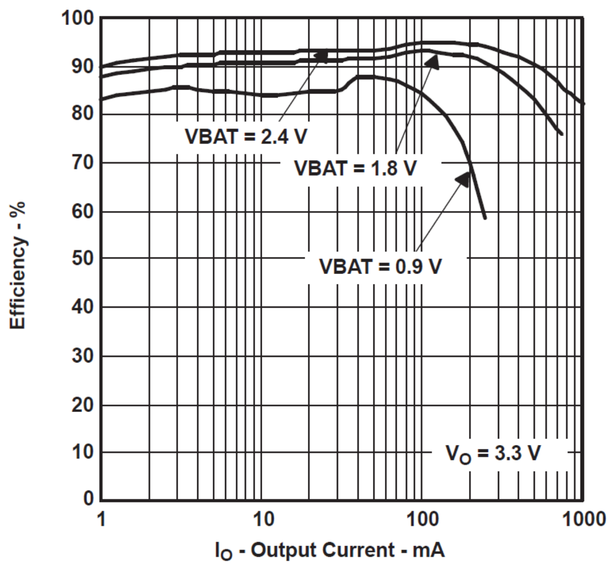


Figure 4.5 TPS61025 Efficiency vs Output Current [20]

It can be observed that the efficiency of the boost converter maintains a value above 80% even at 0.9V input voltage, when the battery is discharged, declining rapidly at output currents higher than 150mA. Therefore, an estimate of 80% efficiency will be used for further calculations. As the efficiency parameter has been evaluated, the value for the required inductor can be determined.

$$I_L = I_{OUT} \times \frac{V_{OUT}}{V_{IN} \times \eta} = 0.3A \times \frac{3.3V}{0.9V \times 0.8} = 1.375A \tag{3}$$

$$L = \frac{V_{IN} \times (V_{OUT} - V_{IN})}{\Delta I_L \times f \times V_{OUT}} = \frac{V_{IN} \times (V_{OUT} - V_{IN})}{20\% \times I_L \times f \times V_{OUT}} = \\ = \frac{0.9V \times (3.3V - 0.9V)}{20\% \times 1.375A \times 600 \times 10^3 Hz \times 3.3V} = \frac{2.16}{544500} = \\ = 4\mu H \tag{4}$$

As the minimum value of the inductor is evaluated to 4μH, the recommended value of 6.8μH present in most typical applications is used.

Further, the output capacitor needs to be selected[11] as well to minimize the ripple of the output voltage of the module. (5) [21] determines the minimum value of the output capacitor, with a desired maximum output voltage ripple of $5mV$.

[11] J. Arrigo, "Input and Output Capacitor Selection", Texas Instruments, Appl. Report. SLTA055, Feb. 2006

$$C_{OUT(min)} = \frac{I_{OUT} \times (V_{OUT} - V_{IN(\min)})}{f \times \Delta V \times V_{OUT}} = \\ = \frac{0.3A \times (3.3V - 0.9V)}{600 \times 10^3 Hz \times 5mV \times 3.3V} = \frac{0.72}{9900} = 73\mu F \tag{5}$$

A 220µF capacitor is used for this application. The chosen type is ceramic, with X5R as dielectric material. It is a good alternative [21], over the tantalum options that have a significant quiescent current, which will unnecessarily drain the battery. An additional 2.2µF output ceramic capacitor is required alongside the 220µF capacitor, according to the datasheet.

The LBO output pin notifies the microcontroller when the input power has fallen below the threshold, which is programmed with the $R_{1/2}$ voltage divider on the LBI input pin. To not be left floating, a pull-up resistor with the recommended value of 1MΩ [20] has been added to the specified pin.

4.1.3 Voltage Regulation Module

As the device has the possibility to be powered by a battery, a voltage supervisor is recommended, according to the datasheet of ESP32 [13]. In case the output voltage from the power source decreases below 2.3V, the CHIP_PU pin is pulled LOW by the voltage regulator, protecting the SoC.

The BD48E27G-TR voltage detector is used in this situation. The schematic follows a typical open drain output type application circuit [22].

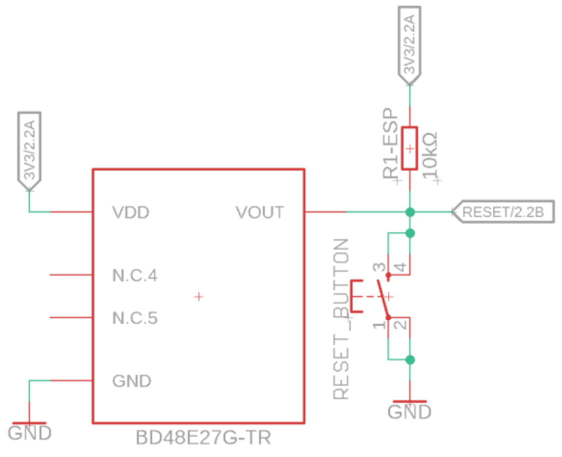


Figure 4.6 BD48E27G-TR Voltage Detector

In addition, a RESET tactile switch is added to pull down the CHIP_PU signal, resetting the chip whenever it is required. A decoupling capacitor with a value of 100pF, as recommended in the datasheet [22], is also present near the CHIP_PU pin for noise filtering. The capacitor is also mandatory for the auto-reset circuit [23], which will be discussed further in the following sub-chapter.

4.1.4 USB Module

The USB module provides one of the two ways of powering the board, connecting it to a device via a micro-B type USB cable, and allows the programmer to download and flash the software into the ESP32 microcontroller via UART protocol. The FT231XS-R [24] part, USB to full handshake UART IC (integrated circuit), is used as the main component for this module.

The image below provides the schematic of the USB module of the application.

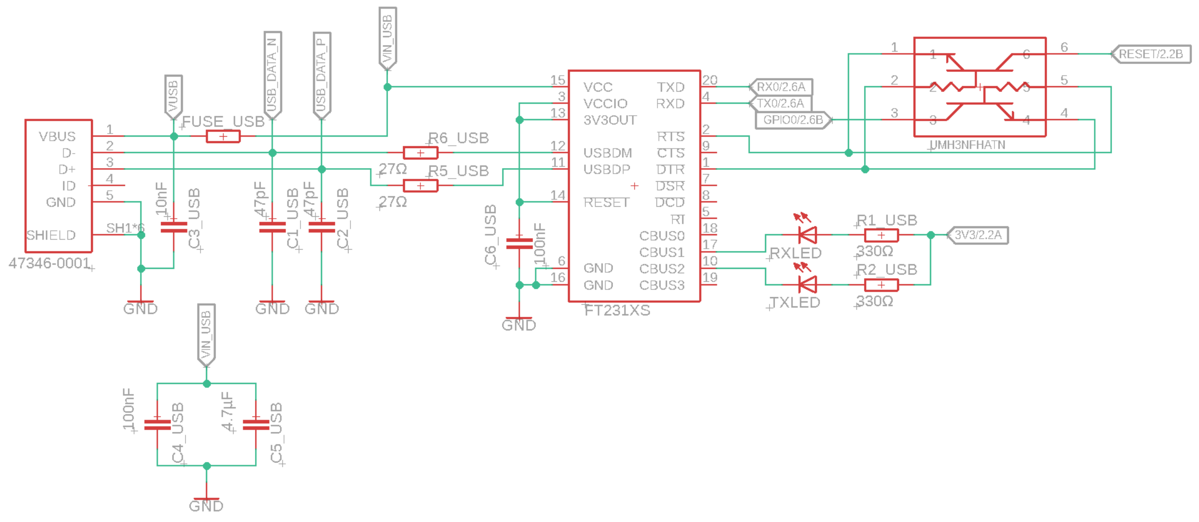


Figure 4.7 USB to full handshake UART IC

As the IC is required to be functional only when the USB cable is connected to flash the ESP32 chip, the typical USB bus powered configuration is used for this module. In this situation, the module is not drawing current from the battery to keep the chip in a quiescent state when it is not required.

Because the module follows the schematic recommendations of the datasheet, the additional parts and their values are determined in the same manner and are detailed below.

A ferrite bead is added on the VBUS connection to mitigate the EMI noise that may radiate the power supply line and should sustain a minimum current draw of 300mA. Two decoupling capacitors are added on the VBUS connection, near the VCC pin of the IC. The selected values for the two capacitors C_1 and C_2 are 100nF, respectively 4.7µF. Another capacitor (C_5) with a value of 10nF is present on the side of the USB port, connected to the VBUS line also.

The RESET# pin allows the IC to be reset by an external source, being described as active LOW. Therefore, it is tied to the 3V3OUT pin as no external circuitry for reset is needed. A required decoupling capacitor is connected to and placed near the 3V3OUT pin as well.

Each of the USBDM and USBDP pins have a 27Ω resistor in series and a 47pF bypass capacitor as well, as presented in the datasheet.

The chip enters the serial bootloader when GPIO0 pin is held LOW on reset, as well as GPIO2 being left unconnected (floating) or being pulled down to GND. The esptool.py [18] connects the EN pin of the ESP32 to RTS and the GPIO0 pin to DTR automatically to enter the serial bootloader, unlike other serial terminal programs, which pull RTS and DTR low while opening the serial port, holding the chip in reset.

The USB module integrates an additional circuit [23] to avoid the chip being held in reset when both RTS and DTR are asserted (found on development boards as well). The UMH3N FHATN[12] dual digital transistor is used in this situation to replicate the required circuit for the above-mentioned problem. A MODE tactile switch is also added for the booting sequence of the microcontroller, connected GPIO0 pin, as well as a pull-up 10kΩ resistor.

Given the fact that the 40MHz external crystal oscillator is placed near the TX pin, a 499Ω resistor has been added to suppress the external clock harmonics, as seen in the below figure [19].

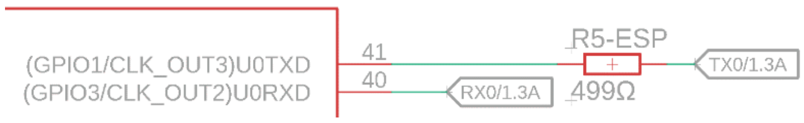


Figure 4.8 TX Series Resistor

Further, the FT231XS-R part contains 4 CBUS (Control Bus) I/O pins with pre-programmed definitions, which can be configured by setting the bits in the internal MTP memory. The table below presents the default factory configurations for the CBUS lines of the module.

[12] Rohm Semiconductors, "EMH3/UMH3N/IMH3A General purpose (dual digital transistor)", UMH3N FHATN datasheet, [Revised Oct. 2015]

Table 4.1 Default Internal MTP Memory Configuration [24] for CBUS lines

Parameter	Value	Notes
CBUS0	TXDEN	Default configuration of CBUS0 – Transmit data enable for RS485
CBUS1	RXLED#	Default configuration of CBUS1 – Receive LED drive
CBUS2	TXLED#	Default configuration of CBUS2 – Transmit LED drive
CBUS3	SLEEP#	Default configuration of CBUS3 – SLEEP#. Logic 0 when the device is in suspend

The CBUS1 and CBUS2 lines are used in this scenario for one LED diode to indicate the transmission of data (TXLED#) and the other LED diode to signal receiving of data. Further, (6) determines a minimal resistance value that must be taken in consideration for the resistor of each LED (RX and TX). As most of the LED diodes that are available on the market have a forward current limit of 20mA, this value is used in (6) as well.

$$R = \frac{V_{CC}}{I_{forward\ current\ limit}} = \frac{3.3V}{20mA} = 165\Omega \qquad (6)$$

As the minimum value for the R_3 and R_4 is 165Ω, resistors between 270Ω and 330Ω are chosen for this application.

4.1.5 Sensor module

The physical device interacts with the environment using the sensor module, which is described further. It contains two sensors, BME680, a four-in-one integrated environmental sensor, and VEML6030, a high accuracy ambient light sensor. Each of the mentioned components are discussed separately.

BME680 [25] is a low power environmental sensor designed by Bosch, which has the capabilities to measure four categories of parameters regarding gas, temperature, humidity, and pressure. The small size, measuring only 3mm x 3mm, the low power consumption and the wide variety of applications for it, make the BME680 a very good environmental sensing alternative for battery powered IoT solutions. The figure below provides the typical application schematic for the sensor in this application:

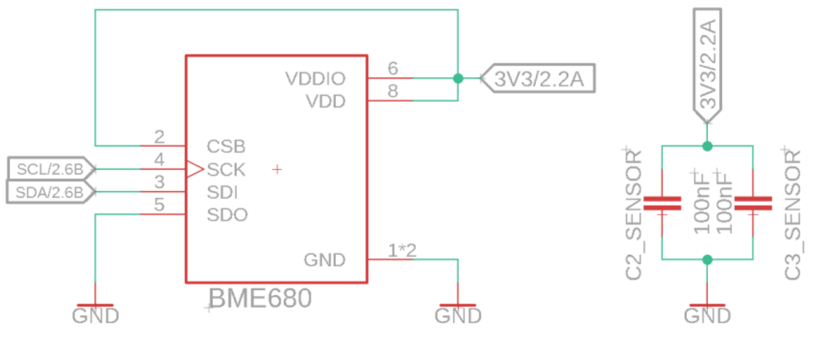


Figure 4.9 BME680 Environmental Sensor

BME680 implements two possible peripheral protocols with which it can exchange data with the microcontroller: I^{2}C and SPI. As both sensors used by this module have I^{2}C capabilities, the protocols are used to communicate with the ESP32 chip. As described in the pin description from the datasheet, the SDO is tied to GND for the device to be recognized with the default address. A decoupling capacitor is also placed near each power supply pin for noise filtering.

The VEML6030 [26] is a high accuracy ambient light sensor available in very small packages, measuring only 2mm x 2mm, representing a good alternative for size constrained IoT applications. Below the typical application schematic is provided, as described in the datasheet:

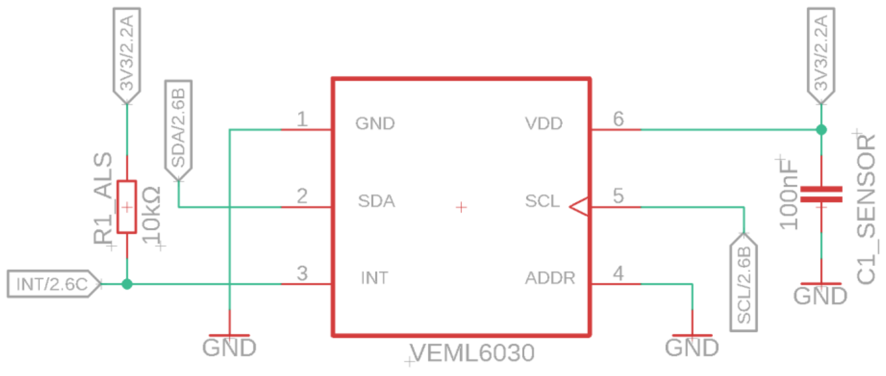


Figure 4.10 VEML6030 Ambient Light Sensor

The ambient light sensor incorporates an I^{2}C interface for communication with other devices and an interrupt feature. Two default addresses are available for the I^{2}C communication in this situation and are defined by the ADDR pin [26]. For simplicity, the 0x10 slave address has been chosen for this component, and can be achieved by pulling ADDR down to GND. A pull-up resistor is added for the interrupt pin, as it is an open drain output, and a decoupling capacitor is also placed near the power supply pin for noise filtering.

As every component of the sensor module communicates via I^2C interface with the microcontroller and the fact that the number of components is limited due to board size constraints and board design parameters (such as the battery holder pin placement, restrictions, and design recommendations [19]), the BME680 and VEML6030 sensors use one pull-up 4.7kΩ resistor for each of the two communication wires: SDA and SCL.

4.1.6 Auxiliary components

The board also disposes of multiple LED diode structures to make the implementation and debugging process easier for the developer. The remaining unattached pins are directed towards pinheads placed at the very margin on both sides. Figure 8.1 and Figure 8.2 provide the layout for the PCB (Printed Circuit Board) layers of the embedded system.

The hardware design of the wearable component is implemented, and the device is materialized. Further, the connectivity aspect is discussed regarding the other components of the project, as well as describing the implementation details for each of them.

4.2 Connectivity

The connection with the smartphone application needs to be developed to ensure the means of communication between the two parts. The smartphone application is required to provide a user-friendly manner of data visualization to the customer, since the wearable part does not satisfy the necessary requirements for this task. Also, it is also intended to remotely act as a control unit for the physical device, performing actions such as rebooting the wearable, requesting data and others.

As stated in chapter 2, the power consumption of the microcontroller is extremely important, especially when the physical component is battery-powered. The two components, the wearable device and the smartphone, are also required to be wirelessly connected, as a direct connection would be most of the time frustrating and would interfere with the ease of movement when operating the devices. Given the specified restrictions and the nature of the project, being mainly focused on transmitting and receiving sensing data, the Bluetooth Low Energy (BLE) is the obvious choice as the communication protocol in this situation. Although Bluetooth Low Energy (BLE) restricts other functionalities, such as communication range or the amount of data transmitted, focusing mainly on consuming the minimal amount of power from the source, it suits perfectly with the intended project in this situation, as sensor based IoT applications rely on short bursts of data transmissions and minimal energy usage [27].

The Generic Attribute Profile (GATT) model is used by the Bluetooth Low Energy protocol to establish the means of communication over a BLE connection, structuring profile and user value data in a hierarchical manner, as seen in the below figure.

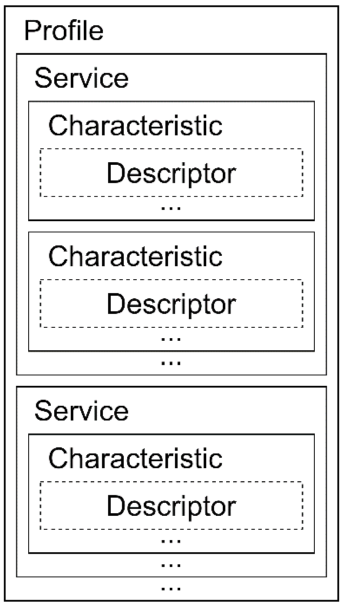


Figure 4.11 GATT Profile Structure [27]

The descriptor has the lowest placement in the hierarchy and defines a characteristic value. A characteristic describes the type of a group of descriptors and can contain none or multiple of such entries. The characteristics are further encapsulated into collections called services, that describe a particular behavior for the contained components. At the highest level of the hierarchy lies the profile. An application can have multiple GATT profiles, implementing different application behavior for each of them.

The components of the GATT profile structure are identified by a Universally Unique ID (UUID), assigned to each of the present ones in the application. There are two types of such UUIDs: 16-bit UUIDs, can be used when predefined Bluetooth SIG attributes [13] are implemented, such as Heart Rate Sensor, and 128-bit UUIDs, for custom services and characteristics.

Further, we discuss the assigned roles and detail the implementation for each of the participating devices. Two roles are available in a GATT implementation:

[13] "Assigned Numbers", Bluetooth SIG, Inc., [Online]. Available: https://www.bluetooth.com/specifications/assigned-numbers/ [Accessed: Jun 16, 2021]

- Client
- Server

The client sends requests to the server to access a specific attribute or to perform an operation. In turn, the server responds accordingly providing the required data or notifying the operation being completed. It also can update a connected client via notifications in certain situations, such as a change of value for a parameter of interest. Based on the described functionalities, the GATT server role is assigned to the wearable device, and the smartphone represents the GATT client.

4.2.1 Wearable device

The wearable component of the project acts as a server for the connected devices, transmitting environmental data and executing control operations issued by the user upon request. It follows the structure of a GATT Server Service Table[14], being the recommended version of implementation. It sets the structure of the profiles and services from the beginning and disables the possibility of unwanted changes, unlike the simple GATT Server[15], which allows the clients to introduce characteristics themselves.

A single profile is used in this situation, encapsulating the predefined Environmental Sensing Service, that contains the following characteristics:

- User Control Point – characteristic for receiving control actions from the user.
- Gas – data regarding air pollution parameters measured by BME680.
- Temperature – temperature measured by BME680.
- Humidity – humidity measured by BME680.
- Pressure – atmospheric pressure measured by BME680.
- Perceived Lightness – ambient light measured by VEML6030.

The application implements the following GATT profile arrangement, using predefined services and characteristics:

[14] "GATT Server Service Table Example Walkthrough", GitHub, Inc., [Online]. Available: https://github.com/espressif/esp-idf/blob/1d7068e/examples/bluetooth/bluedroid/ble/gatt_server_service_table/tutorial/Gatt_Server_Service_Table_Example_Walkthrough.md [Accessed: Jun 16, 2021]

[15] "GATT Server Example Walkthrough", GitHub, Inc., [Online]. Available: https://github.com/espressif/esp-idf/blob/1d7068e/examples/bluetooth/bluedroid/ble/gatt_server/tutorial/Gatt_Server_Example_Walkthrough.md [Accessed: Jun 16, 2021]

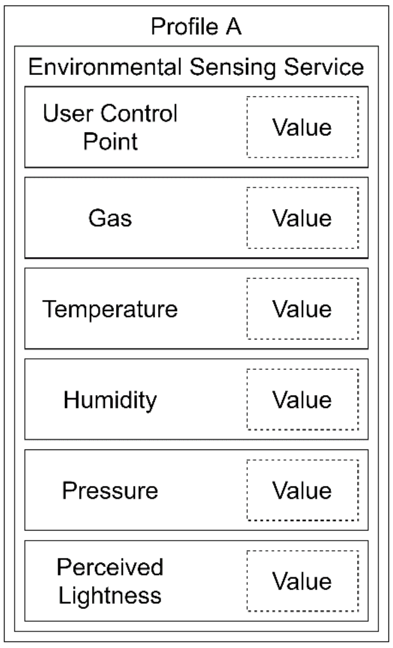


Figure 4.12 GATT Profile Implementation

The wearable application initializes all the necessary prerequisites upon startup, such as the ESP32 Bluetooth Controller and the Bluedroid Stack [28]. The GAP protocol is responsible for the discovering, establishing and managing BLE connections between devices [27], while The GATT protocol defines the model for the communication, as described previously. Therefore, GAP and GATT callbacks are defined as well. The initialization process ends with handles for each of the mentioned characteristics being set to be executed when a request regarding an attribute is received. Further, the application now can listen for requests from the clients and provide a response accordingly or execute a particular action.

The embedded software for the ESP32 is designed and developed using the ESP-IDF Software Development Kit (SDK) [17], the wearable device and an additional ESP32-DevKitC board.

Further, the visualization component is discussed, describing in detail the implementation of the smartphone application.

4.2.2 Smartphone

The smartphone is the secondary component of the solution. It represents the GATT client, and it is designed to send requests of various types to the server and receive responses accordingly. Due to development requirements and available testing equipment, the Android platform is chosen as the support environment for the application.

From a structural perspective, there are two main components:

- Scanning activity
- Control activity

Each of the mentioned activities are further presented in more detail. The image below shows a brief representation of the Android application.

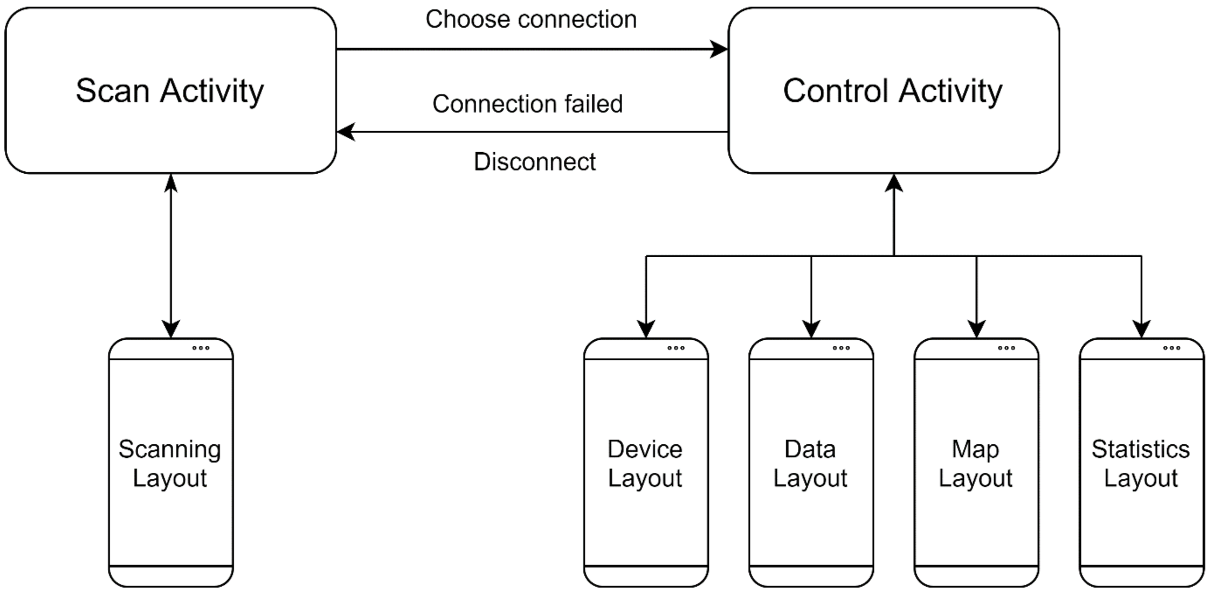


Figure 4.13 Android Application Structure

The scanning activity is the first step of the interaction process between the smartphone and the wearable device. The application performs a search to find nearby devices with Bluetooth capabilities and displays the results, the user being able to select the entry of choice to move forward.

After the desired device is chosen, the control activity, which is responsible for the connection and the communication between devices, displaying the wearable characteristics, environmental data and statistics regarding the air quality, as well as measurements of different nearby locations using Google Maps services, is started.

If the connection between the two devices is not successful, the application returns to the scanning activity. Otherwise, it pursues with the initialization of the required structures for the control activity. The custom Bluetooth Low Energy service, responsible for the connection and communication components, is further called.

Having multiple layouts with different functionalities, a fragment is assigned for each one of them for controlling purposes. Each fragment of the control activity is presented to ensure a better understanding of the visualization logic.

The device fragment is set as default when the control activity starts, displaying information about the connected device. Two possible actions that can be performed for the wearable, disconnect and restart. When choosing to disconnect to the device, the smartphone simply sends a disconnect GATT request and returns to the scanning activity. Further, by pressing the restart button, the application sends a GATT request that notifies the device that a reboot is required, rebooting the device.

The data fragment is the next implemented entity for the application, being responsible for displaying environmental data. For each of the earlier mentioned characteristics (gas, temperature, humidity, pressure and ambient light), a GATT request is issued to update the associated field from the layout accordingly. Once all the values have been updated, the data is made available for visualization and it is stored for future uses on the cloud, which will be discussed later. Such action can be done by pressing the "refresh data" button on the right side of the top bar. The layout is structured to suggest a hierarchical interpretation regarding the importance of the data measurements. The more important ones occupy a larger portion of the screen, while, as the characteristics become less impactful, the size decreases, the position being further down towards the bottom of the layout.

The map fragment uses Google Maps services[16] to display the current location of the smartphone. It also provides elements that describe the air quality in a certain area from previous measurements, using the available data stored in the cloud. The fragment proceeds to send HTTP requests to the backend service to collect all the available data from a particular perimeter, and display it on the map, alongside the time of each measurement. Each retrieved data is associated with a marker on the map, colored according with the IAQ (Index for Air Quality) parameter.

The last view of the control activity is implemented by the statistics fragment. It presents previous data measurements to the user, using line charts to show the trend of the air pollution.

All the fragments are further managed by the navigation bar present at the bottom of the screen to ensure that the user interaction with the application is fast, intuitive and easy to use. The top bar also presents information about the current layout and contains different elements, in regards with each fragment, for example, the "refresh data" button from the data fragment.

One of the downsides of the Bluetooth LE integration on the Android platform is that there is no queueing mechanism to manage BLE operations. Therefore, performing such operations in an overwhelming fashion can lead to unexpected behavior, or even failure [29]. To prevent the application of future bugs and unexpected corner case failures regarding the

[16] "Google Maps Platform", Google LLC, [Online]. Available: https://cloud.google.com/maps-platform [Accessed: Jun. 16, 2021]

execution of BLE operations, a priority queue is implemented. A flowchart of the queueing mechanism is presented below.

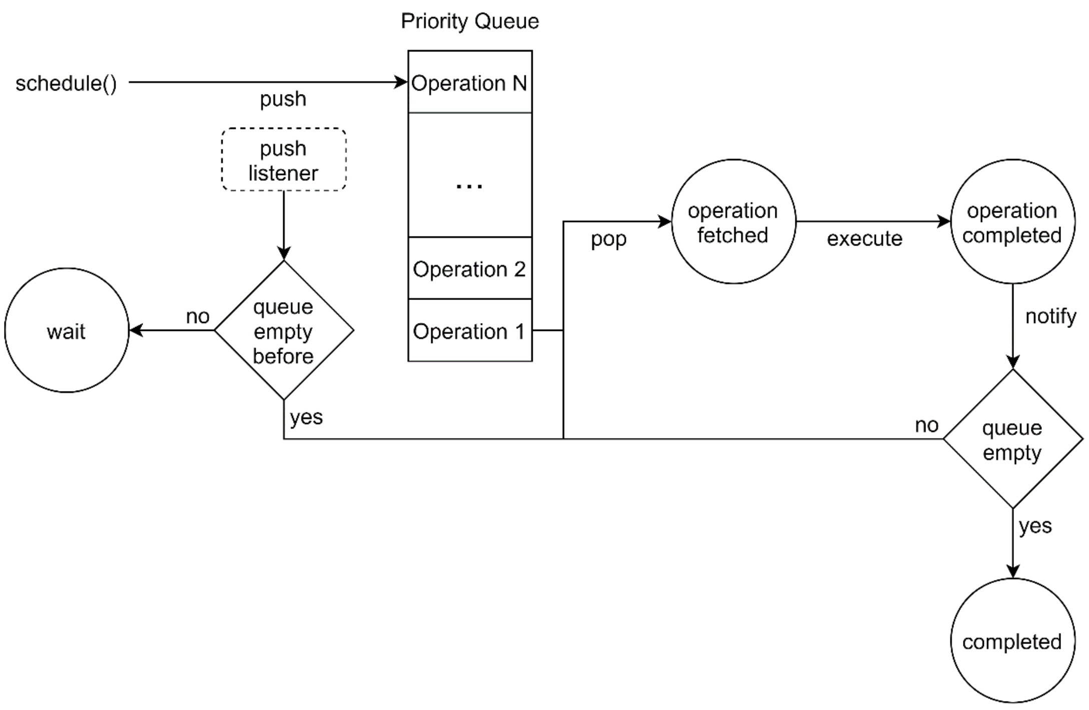


Figure 4.14 Queueing Mechanism for BLE Operations

Using the Command design pattern, all the BLE operations used in the logic of the applications are encapsulated and can be run using the implemented queueing mechanism.

When a GATT request is made, the operation is scheduled to be executed in asynchronous manner, after all the other BLE operations already present in the queue have finished and have signaled their completion.

Further, it is added in the priority queue, triggering the “push listener”, that automatically dispatches the operations for execution. If the queue was empty before the last addition, it proceeds to execute the recently added operation. If the queue was not empty before the last addition, then the execution of the currently added operation is postponed. In this situation, there are other operations waiting to be executed as well, or an operation that is executing at the time, therefore, the last added one is scheduled to run after all the other operations have finished, preventing unwanted overlaps.

The queue extracts the next item accordingly, and proceeds with its execution. After the operation has finished, a notify event is issued to signal the queue that it can begin the next iteration. Further, if there are not any other operations to be executed, the queueing mechanism has completed the current assignments and waits for other operations to be scheduled. If there are other entries that are waiting to be executed, it proceeds with the next operation.

Using the presented feature, we ensure that all BLE operations are executed in asynchronous manner, therefore, avoiding possible errors and unexpected behavior from the application.

Given the fact that the cloud is the place of choice for storing the data measured by the wearable device and multiple features require access to it, the resulting backend component is further discussed.

4.3 Persistent Data Storage

As stated previously, the application requires data storage that exists outside of the other components to save data according to a specific location and date for future usage. The data needs to be available at any time, and not be lost due to external events. Therefore, persistent storage is implemented, where measurements are stored and can be accessed easily at any time throughout a backend application.

The backend application is designed and implemented with the help of String Framework[17], consisting in a simple model of data that encapsulates all the required fields to save the necessary information about a measurement, such as:

- ID – used to locate the entry in the database. The field is auto generated upon measurement registration.
- Location – unique identifier that specifies the location where the measurement has been performed.
- Gas – gas characteristic.
- Temperature – temperature characteristic.
- Humidity – humidity characteristic.
- Pressure – pressure characteristic.
- Light – light characteristic.
- Time – the date and time when the measurement has been performed.

The application follows the standard structure of a Spring Boot Application. It implements a model for the transferred data, a controller that binds different mappings with the associated functionality, servers that implement the means of retrieving and manipulating data of the database.

Multiple methods for manipulating the data are implemented for getting, registering, updating and deleting entries, each of them bound to its respective mapping. The queries

[17] "Spring Framework", VMware, Inc. or its affiliates, [Online]. Available: https://spring.io/projects/spring-framework [Accessed: Jun. 17, 2021]

are made regarding the unique location of each entry ensuring that there cannot be two measurements on the same spot. The user can change the current data for a particular location in case another entry is present in the database.

MongoDB is utilized as the persistent storage for the data, the database being managed by MongoDB Atlas[18] and running on one of the clusters available for customers. Taking in consideration that the database is not running on a local device, but rather on an assigned server, the data is persistent, it is always available and can be modified at any time.

As the smartphone application requires access to the previously stored data whenever suits the user, the backend application needs to be kept running in a continuous fashion as well, like the previously discussed database. For this matter, the Google Cloud Platform[19] (GCP) is used. Deploying the application to a cluster results in consistency in the availability of the database.

4.4 Conclusion

The presented solution aims to combat the air pollution issue and provides the user with all the necessary tools to monitor the environmental parameters. It is structured in three main components:

- The wearable device
- The smartphone application
- Persistent data storage

The means of communication between the two present physical devices (wearable and smartphone) are described by the GATT protocol, working over a Bluetooth Low Energy connection to minimize the power consumption regarding the transmission of data.

The wearable device is the principal component of the project, and it implements the means of measuring the state of the environment. It represents the GATT server in the Bluetooth LE network, listening for requests from the clients and responding accordingly. Based on the market study conducted on chapter 3 as well as looking at the relevant related work, the ESP32 microcontroller is chosen as the main processing unit. Further, because the performance of the chip reflects the overall performance of the entire device, it is required to design and develop the wearable around the ESP32, optimizing the parameters of its execution. For this matter, all the other parts are selected in accordance with the requirements of the mentioned microchip.

[18] "MongoDB Atlas", MongoDB, Inc., [Online]. Available: https://www.mongodb.com/cloud/atlas [Accessed: Jun. 17, 2021]
[19] "Google Cloud", Google LLC, [Online]. Available: https://cloud.google.com/ [Accessed: Jun. 17, 2021]

The smartphone application is the secondary part of the described solution, designed for exposing the measured data for display. In this situation, the smartphone represents the GATT client, sending requests and waiting for responses or notifications from the server. It implements different features for the user to have a better understanding if the parameters and visualize data easily, such as statistics and measurements overview according to locations on the map. It also provides means of controlling the wearable device and the related connection.

The solution also proposes a persistent data storage for measurements to be saved and reused in the future for statistical purposes. It is represented by a backend service that is deployed on the cloud platform to ensure a persistent database and continuous availability of the service.

Further, a case study is required to evaluate the results of the project, as well as the correctness of its implementation.

5 CASE STUDY / RESULTS EVALUATION

In this chapter we aim to evaluate the results of the previously described implementation as well as conduct a case study to review the general point of view of the public towards the proposed solution.

As the project contains multiple parts that work together to achieve the describes goals, it is important to ensure that the anatomy of the solution is adequate for usage. Figure 5.2 shows the general point of view of the public regarding the related structure.

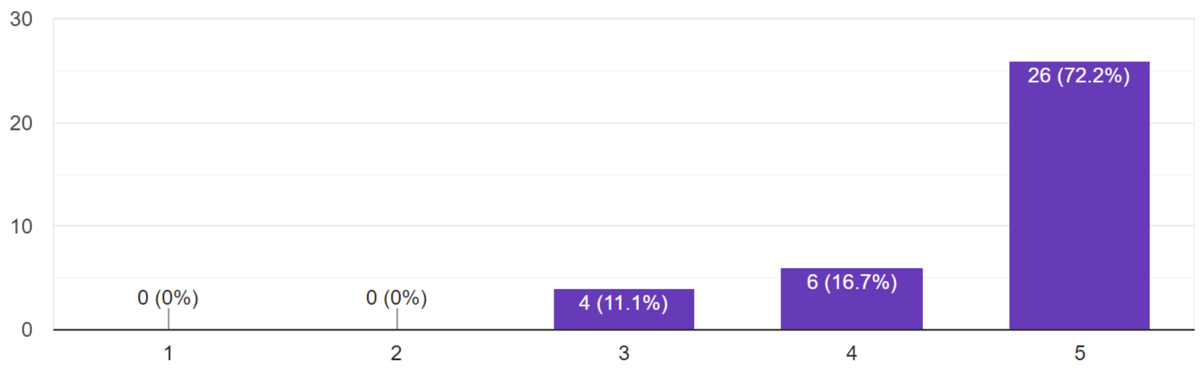


Figure 5.1 Solution structure feedback

As Figure 5.2 presents, 26 (72.2%) participants voted the structure being compliant with the related objectives, 6 of them (16.7%) incline towards the structure being adequate and 4 (11.1%) are neutral. Therefore, we can conclude that the anatomy of the solution is adequate, according to the study, the customer being included in the development process of each component.

5.1 Wearable

The solution relies on the communication between the embedded system and the mobile device to correctly provide the necessary tools for air quality monitoring. Given the fact that the mobility is an important feature for a wearable device, as described in chapter 2, the Bluetooth Low Energy technology is used for connectivity purposes. Although a Bluetooth LE connection provides an advantage over other implementations using an USB cable for communication for example [10], it also presents additional concerns, such as energy consumption management. Like the mobility of the embedded system, the small size of the component contributes as well to the maneuverability aspect of the device.

Another advantage of using the mentioned technology is that the wearable does not have to have access to the internet to provide the requested measurements, unlike a Wi-Fi connection [6], [11], yet it is recommended to have the mobile device connected to the Internet for cloud and map features to work as intended.

Figure 5.4 show the point of view of the participants regarding the characteristics of the embedded system described in chapter 2.

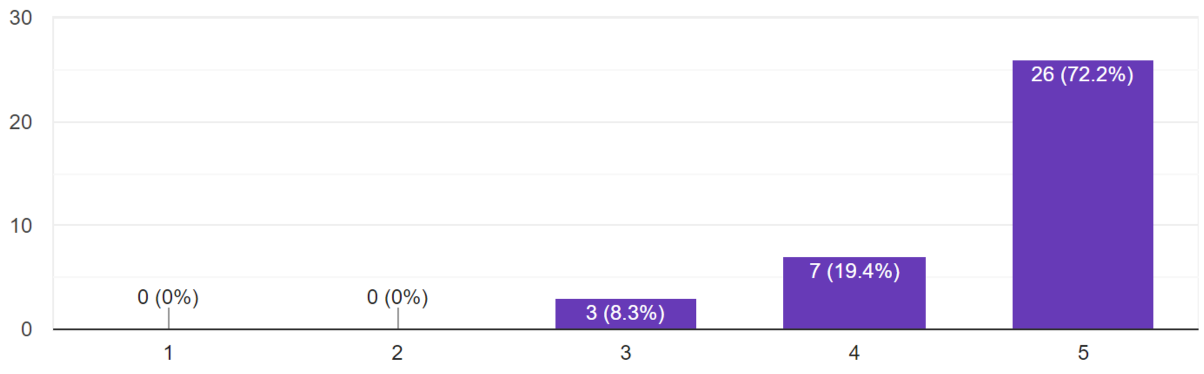


Figure 5.2 Embedded system characteristics feedback

Out of 36 participants, 26 (72.2%) believe that characteristics such as size, mobility, connectivity, low-power, are essential for an embedded device, 7 (19.4%) voted the presented characteristics to be important and 3 (8.3%) share neutral ground. Therefore, the structure of the device satisfies the requirements of possible customers, implementing the desired functionalities.

5.2 Mobile application

Given the fact that the customer interacts mainly with the mobile application, it is very important to be included in the development process. Constant feedbacks are essential in the application to meet the user requirements and expectations.

The implemented features of the mobile application are the result of a study conducted with possible customers, initial features being proposed to the participant and them being required to provide feedback regarding the importance of said elements. The below figure presents the results of the previously mentioned case study.

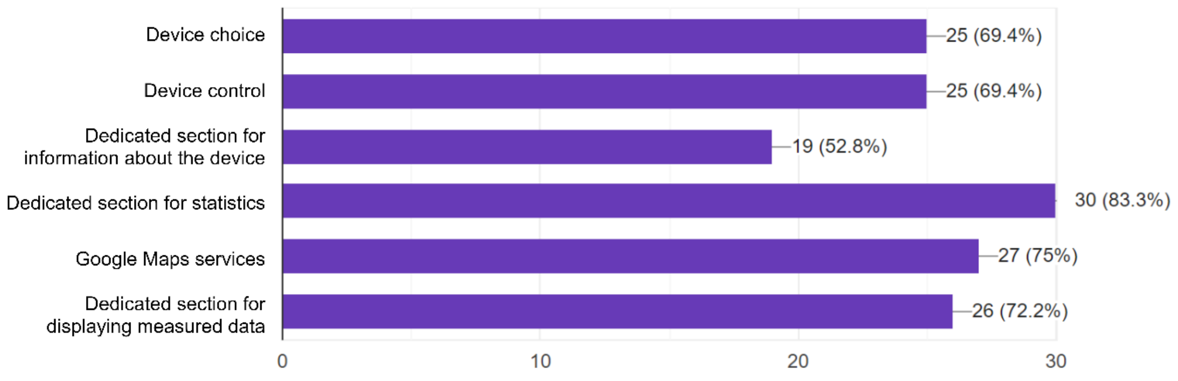


Figure 5.3 Mobile application features feedback

The Android features can be organized in hierarchical manner, from the least to the most important ones, according to the case study. The dedicated section for displaying information about the device is placed on the bottom, with 52.8% pick rate. The device choice and control are of the same priority for the participants, collecting 25 votes each out of 36 (69.4% pick rate). The most important features are represented by the dedicated section for displaying measured data, with a 72.2% pick rate, google maps services, reaching 75% pick rate, and the dedicated section for statistics about previous measurements, collecting 28 out of 36 votes.

As the presented features collected more than 50% votes each, they all are integrated and represent functionalities of the mobile application.

Further, another feedback from possible customers is required, to observe the general response of the public regarding the design and appearance of the mobile application.

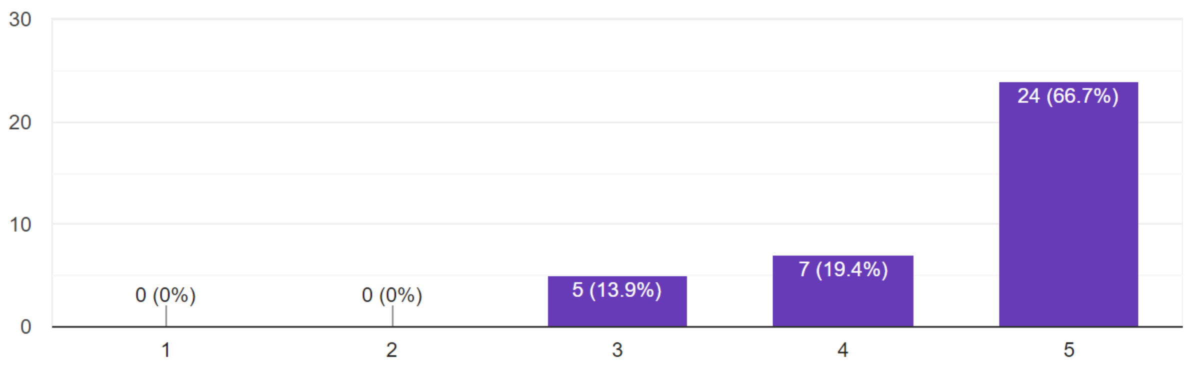


Figure 5.4 Mobile application design feedback

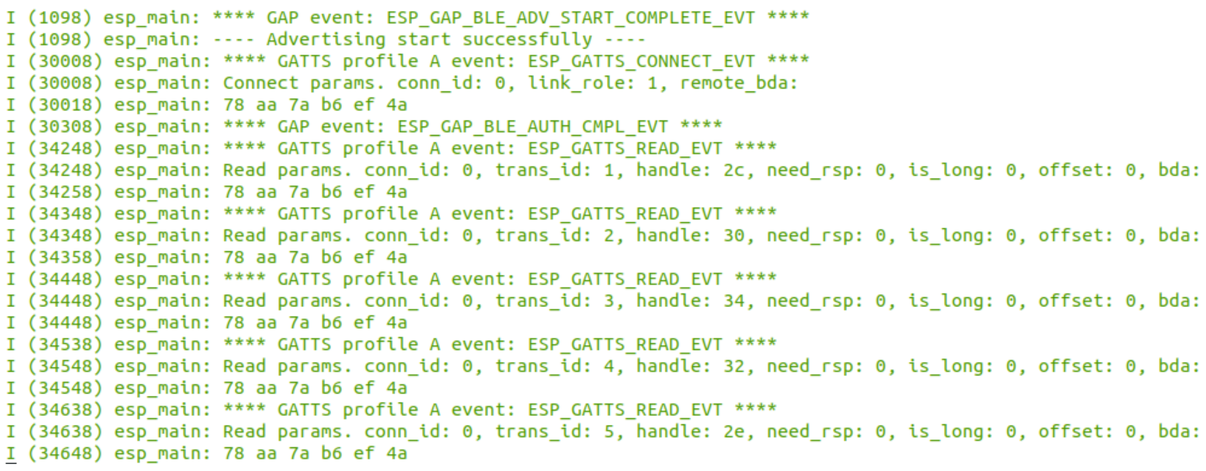
Figure 5.4 shows the feedback from the customers regarding the design of the mobile application, having five possible options of choice, from the design not being appropriate, to being suited to the objectives and structure of the application. The overall point of view is positive, as all the votes are positioned towards the acceptance of the design. Out of 36 participants in the case study, 5 (13.9%) share a neutral opinion, 24 (66.7%) voted the compliance of the design with the related objectives and structure and 7 (19.4%) are placed between the two previously mentioned groups.

5.3 Correctness

To confirm the correctness of the implementation, a test is further conducted, implementing the following procedure:

1) The user connects to the available embedded system
2) New measured environmental data is requested using the mobile application
3) Statistics and map features are reviewed

The wearable is responsible for receiving and processing the requests from the clients and execute operations accordingly. The below picture represents a log section from the embedded device during the test.



```
I (1098) esp_main: **** GAP event: ESP_GAP_BLE_ADV_START_COMPLETE_EVT ****
I (1098) esp_main: ---- Advertising start successfully ----
I (30008) esp_main: **** GATTS profile A event: ESP_GATTS_CONNECT_EVT ****
I (30008) esp_main: Connect params. conn_id: 0, link_role: 1, remote_bda:
I (30018) esp_main: 78 aa 7a b6 ef 4a
I (30308) esp_main: **** GAP event: ESP_GAP_BLE_AUTH_CMPL_EVT ****
I (34248) esp_main: **** GATTS profile A event: ESP_GATTS_READ_EVT ****
I (34248) esp_main: Read params. conn_id: 0, trans_id: 1, handle: 2c, need_rsp: 0, is_long: 0, offset: 0, bda:
I (34258) esp_main: 78 aa 7a b6 ef 4a
I (34348) esp_main: **** GATTS profile A event: ESP_GATTS_READ_EVT ****
I (34348) esp_main: Read params. conn_id: 0, trans_id: 2, handle: 30, need_rsp: 0, is_long: 0, offset: 0, bda:
I (34358) esp_main: 78 aa 7a b6 ef 4a
I (34448) esp_main: **** GATTS profile A event: ESP_GATTS_READ_EVT ****
I (34448) esp_main: Read params. conn_id: 0, trans_id: 3, handle: 34, need_rsp: 0, is_long: 0, offset: 0, bda:
I (34448) esp_main: 78 aa 7a b6 ef 4a
I (34538) esp_main: **** GATTS profile A event: ESP_GATTS_READ_EVT ****
I (34548) esp_main: Read params. conn_id: 0, trans_id: 4, handle: 32, need_rsp: 0, is_long: 0, offset: 0, bda:
I (34548) esp_main: 78 aa 7a b6 ef 4a
I (34638) esp_main: **** GATTS profile A event: ESP_GATTS_READ_EVT ****
I (34638) esp_main: Read params. conn_id: 0, trans_id: 5, handle: 2e, need_rsp: 0, is_long: 0, offset: 0, bda:
I (34648) esp_main: 78 aa 7a b6 ef 4a
```

Figure 5.5 Embedded system log entries

After the connection is established, the user requests all the required data from the server. Each READ event is handled by the assigned handler and is associated with one of the available characteristics on the GATT server and the requests are performed asynchronously, using the GATT operation queueing mechanism implemented on the mobile application.

Further, if we look on the side of the smartphone, we can observe that the device has indeed connected, and the data is updated and displayed accordingly, as shown in Figure 8.3 sections (1) and (2). Section (3) of the same figure show the review of the Google Maps and statistics functionalities.

The Google Maps services offer the benefit of viewing previous measurements from specific locations, the data being updated and uploaded to the database using the implemented backend services of the solution. The marker associated with the data requested for the test is green and is placed to the location of the measurement accordingly. Although the green marker represents the current data, the orange one is hovered to present the consistency of the database, the associated measurement being performed previously.

Finally, the dedicated section for statistics displays previous local measurements using line charts. In this case, only one data is shown, as it is the only one available and positions it in the correct category regarding the IAQ (Index for Air Quality).

5.4 Performance

Further, we aim to analyze the performance of the solution and observe how the system is responding to different situations.

One of the essential parameters that needs to be considered when conducting a performance analysis of such project is the signal strength of the connection. The below figure shows the variation of the signal according to the distance between the embedded system and the mobile device.

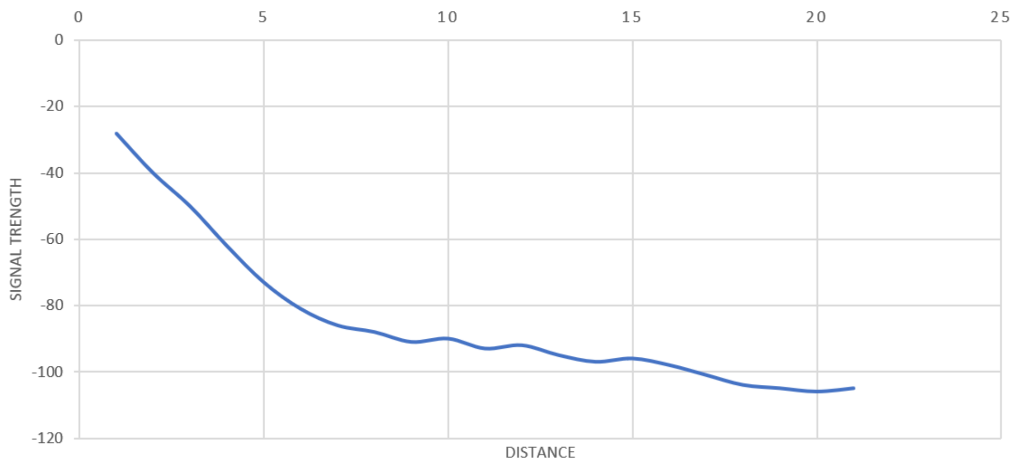


Figure 5.6 Signal strength of the connection

The line chart is composed of data collected from 10 iterations of the same procedure, structured to incrementally measure the signal strength at various distances. It can be observed that the signal strength of the connection queakens as the distance between the two objects grows. Values around -32 dBm are measured at the 1m mark, decreassing constantly, until the -105 dBm mark is exceeded, representing no signal. Any value lower than the no signal limit leads to connection loss and incapability of the Bluetooth scanning mechanism to label the device as eligible for connection, therefore, it is not displayed.

We can further analyze the time variation of the data transfer based of various distances between the wearable and smartphone as well. 12 iterations are elected to be executed, each one requesting data from the system. The request event has the exact behaviour as the correctness test has previously described, 5 individual characteristics being read from the server asynchronously. The duration starts when the first request is issued and ends when the last characteristic has been received by the mobile application. The below figure shows the results of the mentioned procedure.

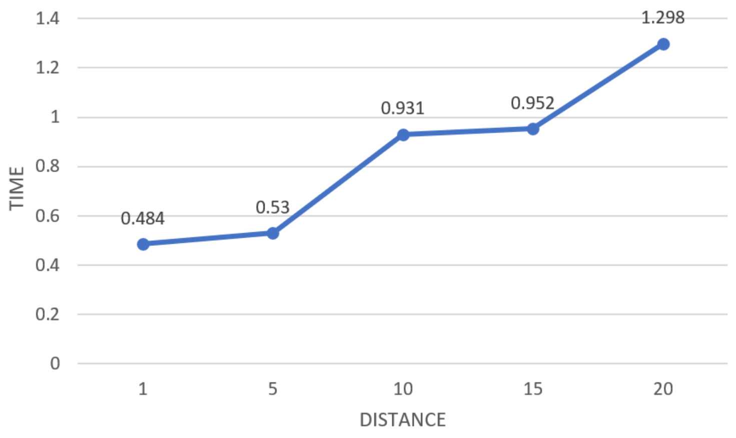


Figure 5.7 Duration of data transfer

The line chart shows the increasing trend of the duration, an average of 0.484s being registered at the 1m mark, on the other end, a value of almost 1.3s being recorded near the no signal limit. Given the fact that the values do not exceed 1s until the signal strength queakens to the point of breaking the connection, the results can be considered satisfactory regarding the responsiveness of the embedded system.

5.5 Conclusion

We can conclude that the proposed solution is implemented correctly, as the presented test describes, implementing all the technical requirements specified in chapter 2. It can be considered an ESP32 based solution, implementing Bluetooth Low Energy capabilities and having all the necessary tools for achieving the stated goals.

Further, the conducted studies have shown that the general point of view towards the overall implementation of the project is positive, and all the user requirements are satisfied.

6 CONCLUSION

The paper presents a viable solution to the very relevant issue regarding air pollution, aiming to equip the customer with all the necessary tools for monitoring the environment, visualizing and storing data. It is structured in 3 main components: embedded system with sensing capabilities, being the central component of the implementation also, mobile application for data visualization and control, and backend services for data storage.

After the requirements of the project have been identified, a market study has been further conducted to evaluate the available options for development and the already implemented solutions addressing this problem. The results have showed that the ESP32 family is best suited as the main processing unit in this situation. Characteristics such as connectivity features, size, price, low-power consumption and high processing capabilities make the mentioned chip the perfect choice for implementing a performant embedded device.

The development process of the hardware component, mobile application and backend services, as well as the required technologies and protocols, and specific mechanisms for overcoming difficulties are implemented, based on the previously gathered information. With the help of constant feedback from the customers and testing processes, we have managed to create a working solution that fulfills all the technical, as well as user-oriented requirements.

Although a viable approach in combating the stated problem has been provided, future work can be pursued. For example, the data visualization is accessible on the Android platform exclusively, therefore, an iOS implementation for the specified component will expand the availability considerably. Improvements regarding the hardware of the embedded system, such as adding a rechargeable power source and its associated module, can also be performed.

The possibility for growth is considerably high and considering that the embedded related market is constantly developing, the methods of implementation and available solutions will continue to improve.

7 BIBLIOGRAPHY

[1] "Air Pollution," World Health Organization, [Online]. Available: https://www.who.int/health-topics/air-pollution. [Accessed 26 Jun. 2021].

[2] L. Francés-Morcillo , P. Morer-Camo, M. I. Rodríguez-Ferradas and A. Cazón-Martín, "Wearable Design Requirements Identification and Evaluation," *Sensors (Basel),* vol. 20, no. 9, p. 2599, 2020.

[3] V. G. Motti and . K. Caine, "Human Factors Considerations in the Design of Wearable Devices," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* vol. 58, no. 1, pp. 1820-1824, 2014.

[4] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications,* vol. 11, no. 6, pp. 54-61, 2004.

[5] S. Li, L. D. Xu and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers,* vol. 17, no. 2, pp. 243-259, 2015.

[6] A. Maier, A. Sharp and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," in *IEEE International Conference on Internet Technologies and Applications (ITA)*, Wrexham, United Kingdom, 2017.

[7] R. Piedrahita, Y. Xiang, N. Masson, J. Ortega, A. Collier, Y. Jiang, K. Li, R. P. Dick, Q. Lv, M. Hannigan and L. Shang, "The next generation of low-cost personal air quality sensors for quantitative exposure monitoring," *Atmospheric Measurement Techniques Discussions,* vol. 7, no. 1, p. 3325, 2014.

[8] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications,* vol. 12, no. 1, pp. 12-26, 2005.

[9] M. Babiuch, P. Foltýnek and P. Smutný, "Using the ESP32 Microcontroller for Data Processing," in *20th International Carpathian Control Conference (ICCC)*, Krakow-Wieliczka, Poland, 2020.

[10] D. Hasenfratz, O. Saukh, S. Sturzenegger and L. Thiele, "Participatory Air Pollution Monitoring Using Smartphones," in *Proceedins of the 2nd International Workshop on Mobile Sensing*, Beijing, China, 2012.

[11] K. Hu, Y. Wang, A. Rahman and V. Sivaraman, "Personalising Pollution Exposure Estimates Using Wearable Activity Sensors," in *IEEE Ninth International Conference on*

Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2014.

[12] G. Parmar, S. Lakhani and M. . K. Chattopadhyay, "An IoT Based Low Cost Air Pollution Monitoring System," in *International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, Bhopal, India, 2017.

[13] Espressif Systems, "ESP32 Series Datasheet," ESP32 datasheet, Oct. 2016 [Revised Mar. 2021].

[14] Espressif Systems, "ESP8266EX," ESP8266 datasheet, Dec. 2015 [Revised Oct. 2020].

[15] Texas Instruments, "CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a Single-Chip Wireless MCU," CC3200 datasheet, Jul. 2013 [Revised Feb. 2015].

[16] Atmel, "ATmega328P," ATmega328P datasheet, Nov. 2009 [Revised Jan. 2015].

[17] "Official IoT Development Framework," ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD, [Online]. Available: https://www.espressif.com/en/products/sdks/esp-idf. [Accessed 5 June 2021].

[18] "Arduino core for the ESP32," Github Inc., [Online]. Available: https://github.com/espressif/arduino-esp32. [Accessed 5 June 2021].

[19] Espressif Systems, "ESP32 Hardware Design Guidelines," ESP32 hardware design guidelines, Dec. 2016 [Revised Apr. 2021].

[20] Texas Instruments, "TPS6102x 96% Efficient Synchronous Boost Converter," TPS21025, Sep. 2003 [Revised Dec. 2014].

[21] B. Hauke, "Basic Calculation of a Boost Converter's Power Stage," Appl. Report. SLVA372C, Texas Instruments, Nov. 2009 [Revised Jan. 2014].

[22] Rohm Semiconductors, "Voltage Detector IC Series, Standard CMOS Voltage Detector IC," BD48E27G-TR datasheet, [Revised Dev. 2020].

[23] "ESP32 Boot Mode Selection," GitHub, Inc., [Online]. Available: https://github.com/tttapa/ESP8266/issues/11. [Accessed 5 June 2021].

[24] Future Technology Devices International Ltd, "FT231X (USB to FULL HANDSHAKE UART IC)," FT321XS-R datasheet, Dec. 2011 [Revised Feb. 2013].

[25] Bosch, "BME680 Low power gas, pressure, temperature & humidity sensor," BME680 datasheet, [Revised Jan. 2021].

[26] Vishay Semiconductors, "High Accuracy Ambient Light Sensor With I2C Interface," VEML6030 datasheet, [Revised Sep. 2019].

[27] K. Townsend, C. Cufí, Akiba and R. Davidson, Getting Started with Bluetooth Low Energy, Sebastopol, CA: O'Reilly Media, Inc., 2014.

[28] Espressif Systems, "ESP32 Bluetooth Architecture," ESP32 Bluetooth architecture, Jan. 2018 [Revised Nov. 2019].

[29] "The Ultimate Guide to Android Bluetooth Low Energy," PunchThrough, [Online]. Available: https://punchthrough.com/android-ble-guide/. [Accessed 12 Jun. 2021].

8 ANNEXES

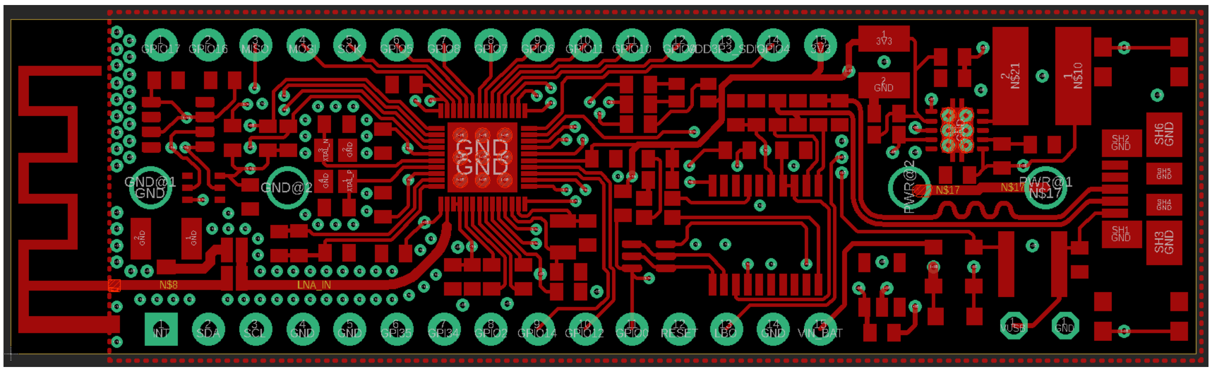


Figure 8.1 PCB Top Layer

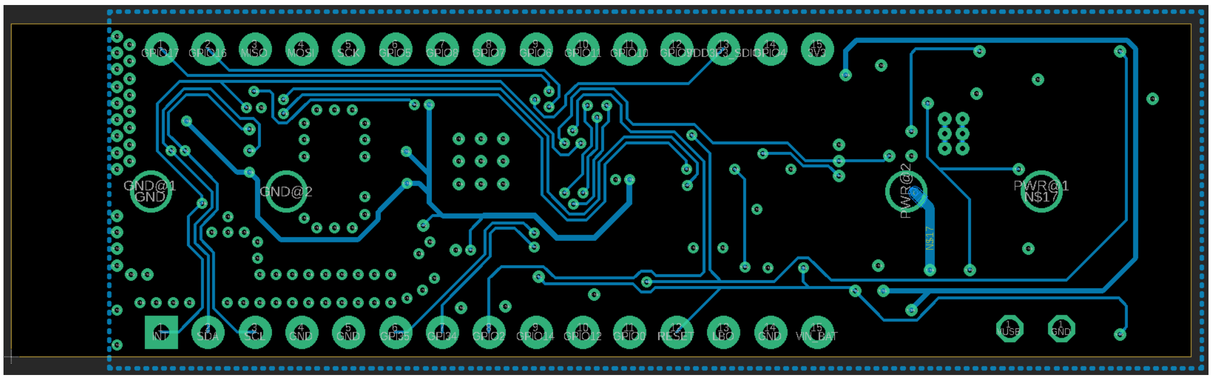
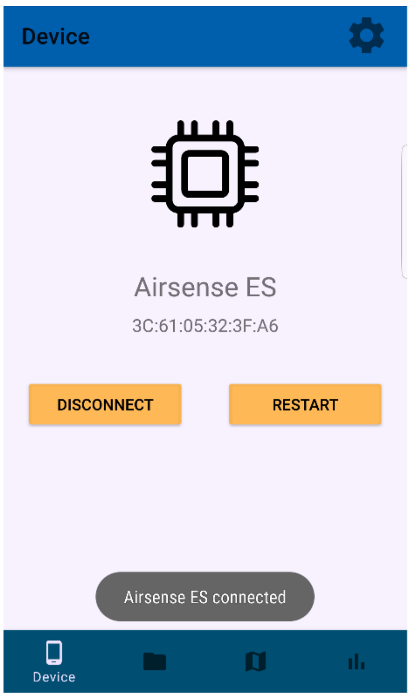
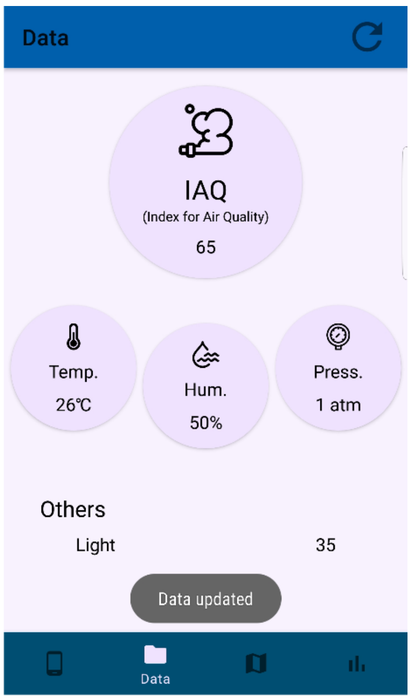


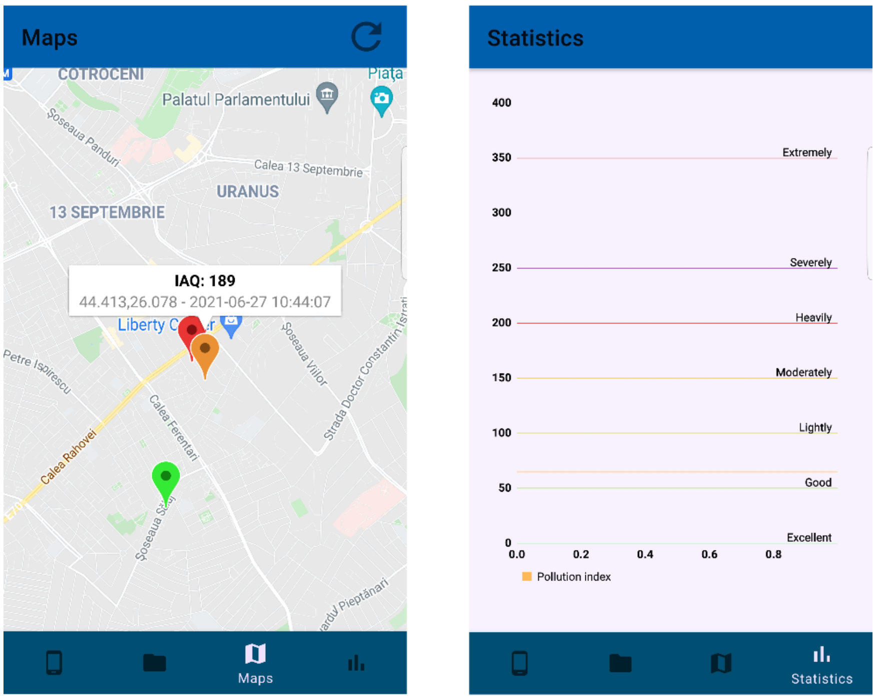
Figure 8.2 PCB Bottom Layer



(1)



(2)



(3)

Figure 8.3 Mobile application perspective