Development of an integrated respiratory rate monitoring system using STM32 micro-controllers.

Luca De Bartolo
Simone Pasquariello
Lorenzo Bartalucci
Marco Corrado
Francesco Ronchi

January 30, 2024

Contents

Chapter 1

Introduction

1.1 Summary

This project proposes the implementation of a system for noninvasive monitoring of respiratory rate through the use of an STM32 micro-controller. The hardware platform, designed in-house, incorporates a PCB on which C-language code has been implemented to manage and analyze signals from a wearable band placed around the user's chest.

The band, made of a stretch-sensitive material, acts as a resistance sensor, transmitting analog signals to the micro-controller in response to chest movements during breathing. Code implemented on the STM32 micro-controller processes these signals, identifying breathing cycles and calculating the user's breathing rate.

This approach not only enables continuous and accurate monitoring of respiration rate, but also offers a convenient and unobtrusive alternative to more invasive devices. The proposed system is intended to find practical applications in settings such as remote health monitoring and management of medical conditions that require regular monitoring of respiration. The integration of the STM32 micro-controller and the tailored design of the band make the system flexible and adaptable to the specific needs of users, paving the way for potential future developments in noninvasive bio-metric monitoring.

1.2 System specifications

The purpose of the project is to make a wearable device for measuring respiration rate using a silicone band applied to the chest. The respirometer consists of a wearable band to measure respiration rate that is composed by the strain gauge sensor, which is a silicon fabric that responds to strain with a change in resistance

$$\Delta R = R_0 \cdot S \cdot G \tag{1.1}$$

where R_0 is the resistance at rest, S is the strain and G is a constant called the gauge factor. First of all we need to describe the strain gauge sensor characteristics. It is called Muriel Sensor and was produced by a company in Reggio Emilia, Italy, called Le Mur. The major disadvantage is that it is a fabric composed of interwoven fibers, so it has some non-linearities. In fact, the step response of the sensor is a falling exponential that takes about 10 seconds to return to the resting value of about 5.2 kOhm. However, as can be seen from the following image, the calibration curve of the sensor was obtained by measuring its resistance each time causing a strain of a cm more than the previous one and waiting for the resistance value to return to steady state, and it can be observed that this curve follows the regression line quite well.

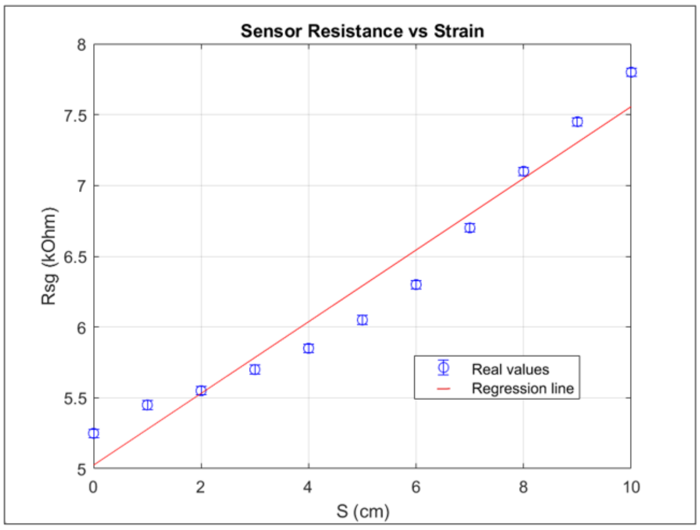


Figure 1.1: Calibration Curve

The project involves the design of a logic board, incorporating a microprocessor, supporting logic circuits, and two alternate power supply. This board is responsible for acquiring analog input data and transmitting the results through a communication line. Power is sourced from an external 5V supply as well as a local Li-Ion battery. The board must possess the following capabilities:

- Normal Operation on External Power: The board should function seamlessly when powered from an external source.
- Switch to Battery Power on External Power Failure: In the event of a failure or interruption in the external power source, the board should automatically switch to the Li-Ion battery as a backup power supply.
- Reversion to External Power: Once external power becomes available again, the system should seamlessly transition back to operating from this source.
- Li-Ion Battery Optimization: The design should prioritize minimizing the size of the Li-Ion battery to save space and weight in the application.
- Battery Recharging: When external power is accessible, the system should initiate the recharging of the Li-Ion battery to ensure it is ready for backup power use.

This multifaceted system must effectively manage power sources, power modes, and data storage while ensuring a compact and efficient Li-Ion battery solution.

According to project use case, the following electrical specifications have to be considered:

1. **Minimal Consumption**: The system is designed to minimize power consumption, ensuring efficient use of energy.
2. **Ambient Operating Temperature Range**: The system is capable of functioning reliably in an ambient temperature range from 0 to 85°C, making it suitable for a wide range of operating conditions.
3. **Communication Line**: The system utilizes a I2C in order to write on the embedded OLED display.
4. **Analog Input**: The system features one analog input with the following specifications: Single-ended input with a range of 0-3.3V and sampling rate of 50 Hz.

Chapter 2

Project Management

This chapter provides a comprehensive overview of the project management process. It delves into the intricate details of the research and development phases that culminated in the creation of the final product. The journey commenced with an initial exploration of the components required for the project, which was carried out in tandem with a series of group meetings employing a stream of consciousness approach.
Furthermore, the progression of the project was meticulously documented through the use of various tools and methods. These include the establishment and maintenance of a Gantt chart, which served as a visual timeline for project milestones and tasks. In addition, regular meeting reports and specific project reports were generated to capture the insights and progress made during the development process.
An essential aspect of this documentation effort involved tracking the allocation of time and resources. It included recording the duration and contributions of each team member to specific tasks, facilitating a clear understanding of the workload distribution and resource utilization throughout the project's lifecycle.

2.1 Task division

Regarding the task division, it closely followed the organizational chart. Specifically, Lorenzo Bartalucci oversaw the entire project's execution, Luca De Bartolo and Simone Pasquariello were specialized on the Software, while Francesco Ronchi and Marco Corrado worked through on the Hardware part.
A weekly meeting was scheduled, during which each team member presented and explained their activities for the week, discussed any challenges encountered, and outlined the next steps in the project. This approach ensured a continuous forward-looking perspective on the project's progress.
To manage task statuses and organize our work effectively we utilized Monday, a team management software.
This platform allowed us to delineate different project phases and their progression, providing a comprehensive overview of the entire project. The Gantt chart (Figure 2.1), in particular, proved highly beneficial in clarifying task relationships and dependencies.

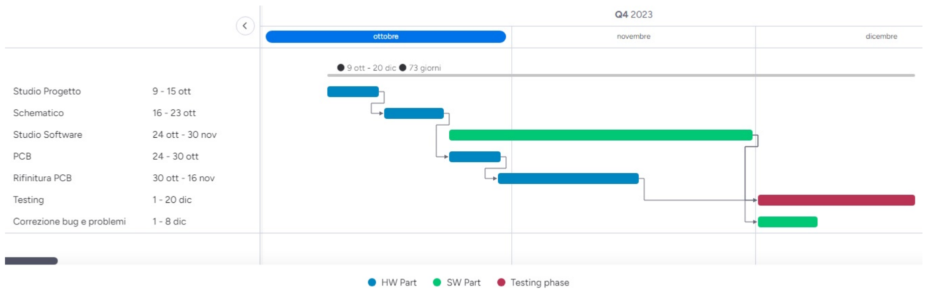


Figure 2.1: Gannt Chart

Our journey began with the research and development phase, where we initiated the system from a block diagram design, starting with the fundamental concepts of our project and gradually delving into the specifics of its design and implementation.
Subsequently, we compiled a Bill of Materials (BOM) encompassing all necessary components, considering various options that met our requirements and were best suited for our application, while also factoring in market availability. This phase concluded with the placement of an order with Mouser. Following this, our focus shifted to the PCB design, ultimately leading to order placement with JLCPCB.
In the realm of software analysis, we embarked on an exploration of MCU and IC datasheets to gain insights into developing software libraries and functions. We adhered to the product specifications, ensuring that our State Machine implementation paid special attention to actions necessitating specific power modes for computation. Our objective was to maximize efficiency while considering the imposed constraints.

2.2 Software Used

From the software point of view the following were used:

- The IDE used was CUBEMX accordingly to the micro-controller, using the NUCLEO development board in order to test the firmware.
- For realizing the Schematics and PCB layout it was used the Altium Designer software, from Altium.

Chapter 3

System Architecture

The system hardware and software co-design is crucial for achieving optimal performance, efficient resource utilization, and meeting the specific requirements and constraints of embedded applications. It results in more reliable, cost-effective, and adaptable systems that can better handle the challenges of the embedded environment. This approach is essential in embedded systems for several reasons:

- Optimized Performance: Hardware and software are tightly intertwined in embedded systems. Co-design allows for the optimization of both components to achieve better overall performance. By considering hardware and software together, designers can identify opportunities for hardware acceleration, parallel processing, and other optimizations that may not be apparent when designing each component in isolation.

- Resource Utilization: Embedded systems often operate under tight resource constraints, such as limited power, memory, and processing capability. Codesign enables the efficient utilization of available resources by tailoring both the hardware and software to work seamlessly together, avoiding unnecessary overhead and ensuring optimal use of resources.

- Real-Time Requirements: Many embedded systems have real-time constraints, where certain tasks must be completed within specific time limits. Co-design facilitates the identification and elimination of bottlenecks in both hardware and software that might affect the system's ability to meet real-time requirements.

- Reduced Development Time and Costs: Co-design can lead to a more streamlined development process. By considering both hardware and software requirements simultaneously, designers can make informed decisions early in the design cycle, reducing the likelihood of costly redesigns later in the development process.

- Improved Reliability and Robustness: Embedded systems often operate in challenging environments, and reliability is a critical factor. Co-design allows designers to address reliability issues by optimizing the interactions between hardware and software, identifying potential failure points, and implementing appropriate error-handling mechanisms.

- System Flexibility and Upgradability: As technology evolves, embedded systems may need to be upgraded or modified. Co-design ensures that the system architecture is flexible enough to accommodate future changes without significant disruptions. This is especially important in embedded systems with long life-cycles.
- Security Considerations: Security is a critical aspect of embedded system design. Co-design allows for the integration of security features at both the hardware and software levels, creating a more robust defense against potential vulnerabilities.

3.1 Hardware Architecture

The main idea for the hardware consist in a first protection circuit for ESD, all the pins exposed to the hand contact has been endowed of a TVS diodes.
Further protections can be achieved using reverse polarity protection, in order to avoid that an error in the connector positioning could cause a short circuit able to burn the board.
Then the sensor, which a variable resistance sensor, is passed through a conditioning circuit able to enlarge its characteristics. In particular, using a Wheatstone bridge circuit to convert the resistance change into a voltage change.
In order to be suitable for people with different chest sizes, the system uses a potentiometer, inside the Wheatstone bridge, that has to be adjusted manually, with the help of a calibration coefficient on the display. Since this is rather uncomfortable, we are thinking of a new solution where the resistance is changed directly by the micro-controller. Then, using op-amps, we used a differential amplifier to improve the accuracy when reading the data with the micro-controller's ADC. The gain is chosen in order to use all the input dynamics of the ADC.

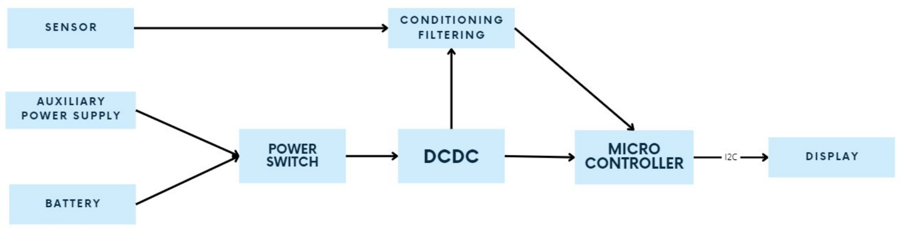


Figure 3.1: System Architecture

For noise filtering we opted for a second order active filter (Sallen Key filter) to remove frequency components above $1.1Hz$ (the maximum standard respiration rate of a human is below 1Hz).

$$f = \frac{1}{2\pi\sqrt{R_1R_2C_1C_2}} \tag{3.1}$$

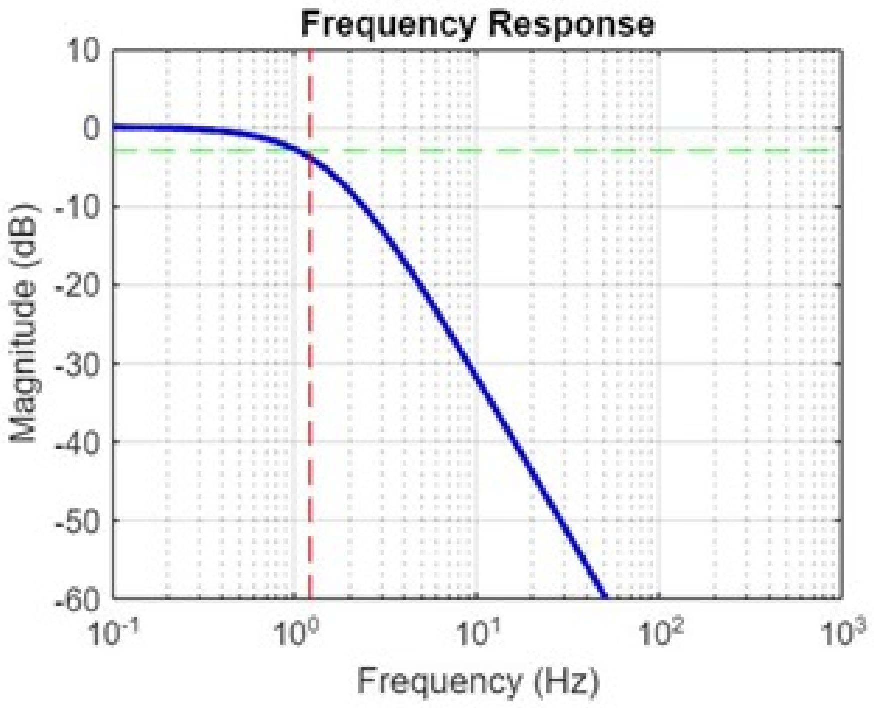


Figure 3.2: Filter Frequency Response

3.2 Power supply Architecture

For what concerning the power supply section we have three main blocks, a first integrated circuit that works as a battery charger, an analog switch circuit made of p-channel mosfets, and then we have a DCDC converter.
From the perspective of power management, we encountered three primary constraints:

- Charging the battery when the auxiliary power supply is connected while the system continues normal operations.
- Switching from the auxiliary power supply to the battery source when the power supply is disconnected, allowing the system to continue its operation, and then switching back when reconnected.

When we discussed about the best way to convert the external voltage into a one usable for the board and at the same time able to recharge our battery, we found three possible solutions:

1. The initial approach, which was ultimately dismissed due to its low efficiency, reliability, and elevated power consumption, comprised several sequential steps:
   (a) The preliminary design involved employing a first Battery charging circuit.

(b) Subsequently, the design incorporated the use of an IC functioning as a power switch;

(c) Due to the 3.7V battery, a buck converter was integrated into the system to ensure a consistent 3.3V output, regardless of whether the power was coming from the battery or an external power supply.

The overall framework is depicted below:

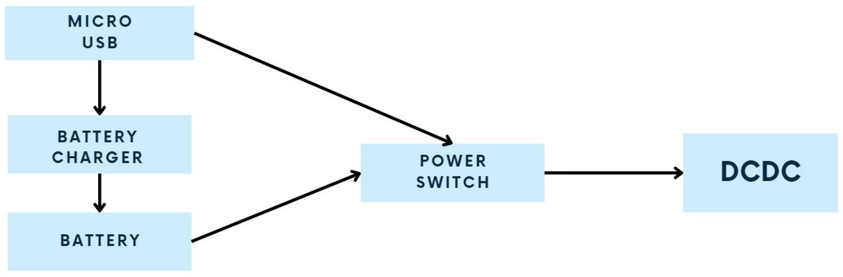


Figure 3.3: Initial Approach

2. The second approach involves an IC functioning as both a battery charger and a power backup manager.
This system charges the battery from a primary power source while providing power to the load. If primary power is lost, the load is seamlessly transitioned to the backup battery.
This solution was by far the most efficient but very high cost compared to the third option we choosen.

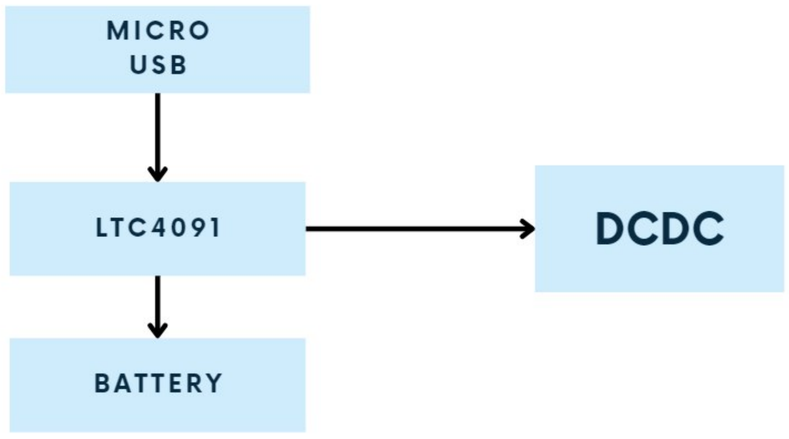


Figure 3.4: High Cost Architecture

3. The solution we ended up to use was chosen because of the low cost and

high integration possibilities, because of the use of a battery charging circuit that was very small. To avoid using multiple integrated circuits, the switching between the two power sources was done using a network of Schottky diodes and MOSFETs. In particular, two MOSFETs were connected in series but mirrored, with the same gate drive circuit, to avoid reverse current flowing through their internal body diode from the circuit back to the battery in an uncontrolled way, since from there the battery has only to deliver power when needed.

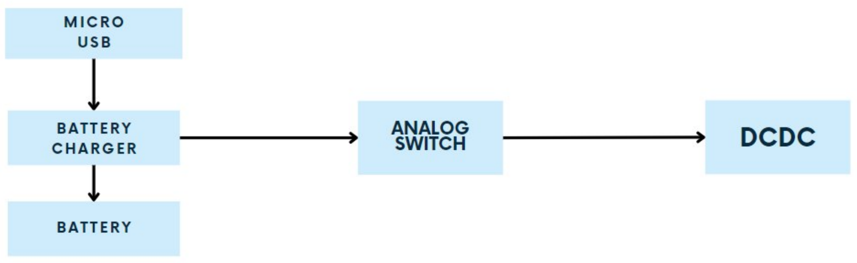


Figure 3.5: Low Cost Architecture

The overall reliability is enhanced as the system involves just one IC and a single DCDC.

3.3 Software Idea

The main Idea regarding the software included using a timer in order to sample the analog input through the ADC embedded in the micro-controller.
We wanted to use interrupt driven ADC, everytime the timer activates the main function the ADC convertion is started and then, an interrupt event is responsible of the insertion of the sampled data into the Value variable, then used in the main program.
In order to measure the frequency an FFT funcion has been created, the peak frequency is then collected and converted to frequency rate by the following relationship:

$$f_{resp} = Peak_{freq} \cdot 60 \tag{3.2}$$

This value is then sent to the OLED display by its specific funcion, that uses the related library.

Chapter 4

HW design

4.1 Simulations

4.1.1 Charging Circuit and Power Switch

Regarding the simulations we needed to test the working principle of our analog switch.

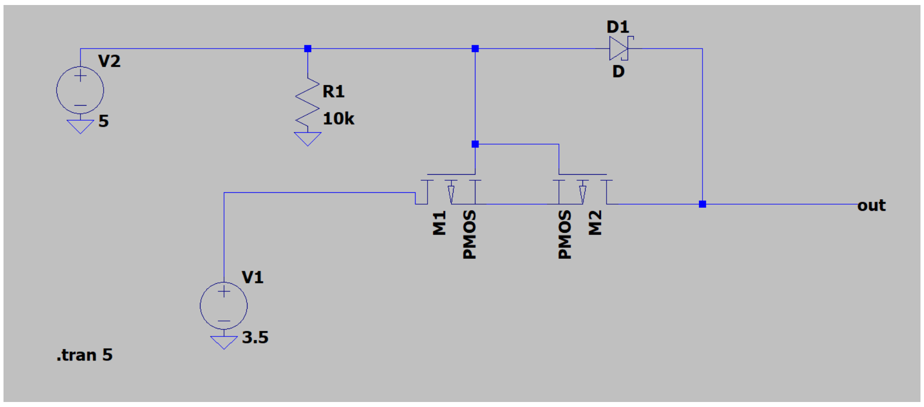


Figure 4.1: Power Switch

It is possible to see that, regardless of the presence of the input voltage Vin, the output voltage is constant above the threshold of the DCDC converter input. Moreover the switching between the two power sources has been tested, and allows the continuity of the service without losing data coming from the sensor.

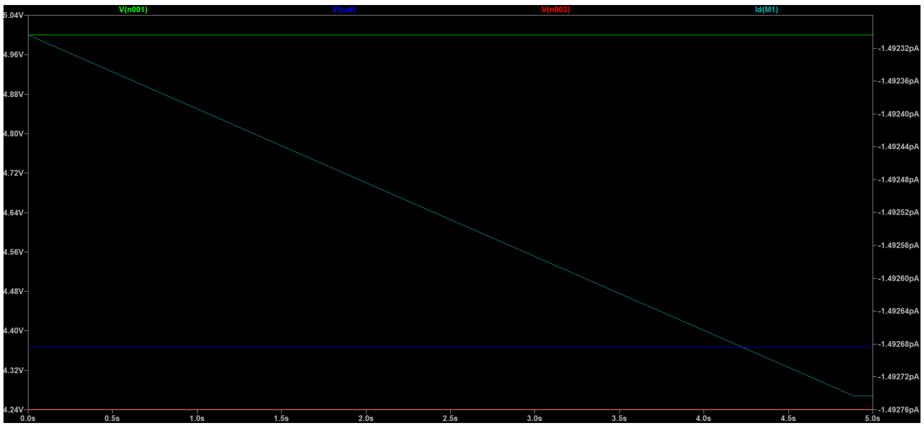


Figure 4.2: Power Switch OFF

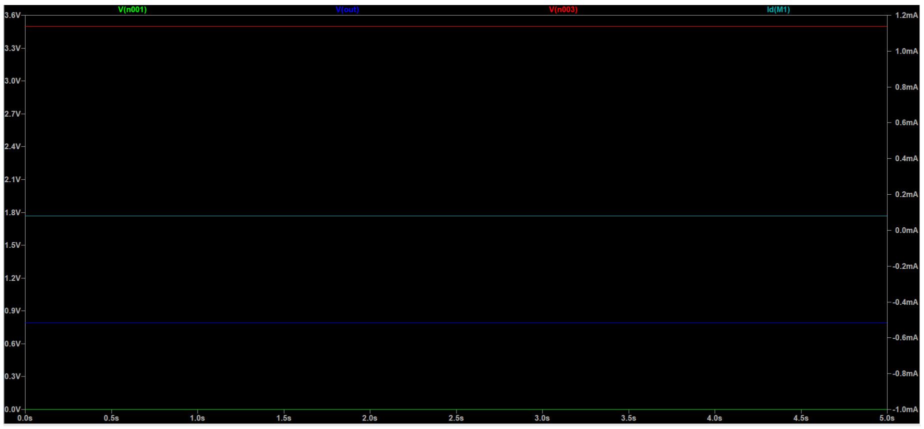


Figure 4.3: Power Switch ON

4.1.2 Sensor Conditioning and Filtering

All the data from the sensors underwent conditioning for proper functioning. Specifically, our circuitry had to meet the following requirements:

1. Protection against electrostatic discharge.

2. Filtering out high-order harmonics while accommodating variations in sensor speed.

3. External events impacting the micro-controller's pins from outside the board necessitated buffering for protection.

Regarding the specific implementation of the conditioning technique applied to the band sensor, we conducted calculations to determine the most suitable configuration for our requirements.
We evaluated two configurations:

1. Resistor Partition followed by a buffer.

2. Wheatstone Bridge followed by a differential amplifier.

The first configuration exhibited an accuracy of $\dfrac{0.1V \cdot 2^{12}}{3.3V} = 3\%$ which proved to be excessively low for our application.
The second configuration necessitated the use of two extra operational amplifiers for buffering the partition voltages. But it proved to be way more accurate, since all the 100%, of the possible values of the ADC could be used allowing a very high accuracy.

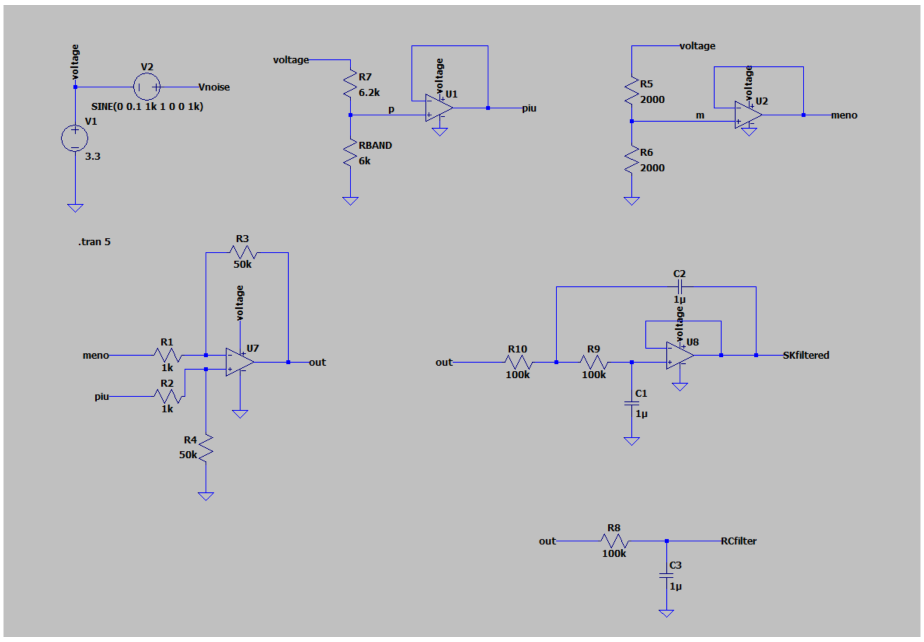


Figure 4.4: Signal Conditioning and Filtering

We simulated the conditioning network with variable resistance values, in the range of our resistive sensor:

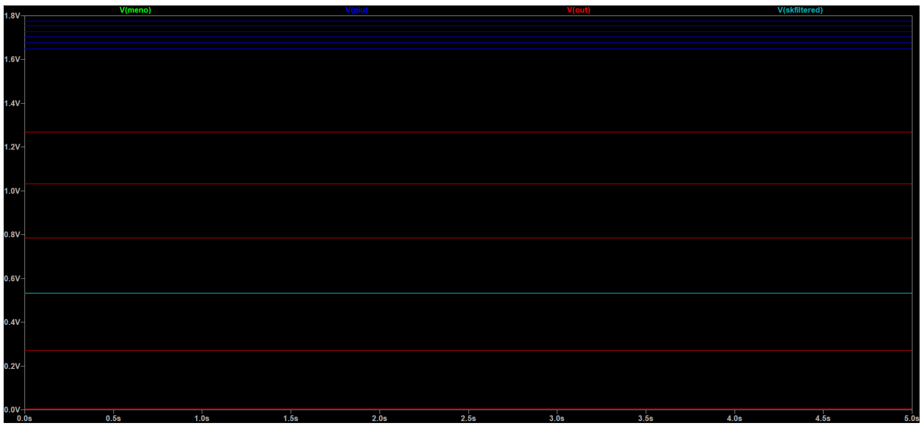


Figure 4.5: Simulation of Resistance sweep

Then, approximating a noisy signal as a sinusoid that is added to our dc voltage, we simulated the difference between an unfiltered signal and a filtered one, using at first a RC filter and then a SK filter.

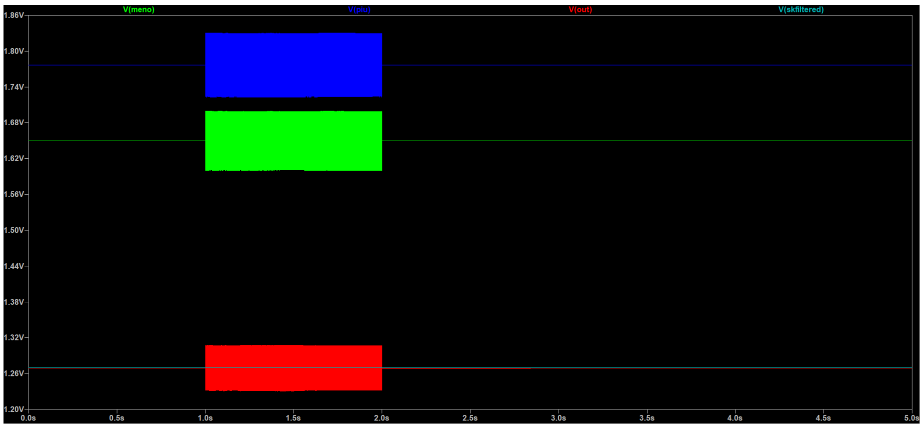


Figure 4.6: Simulation without a filter

In the simulation we can see the ripple is really small compared to the signal, hence low SNR.

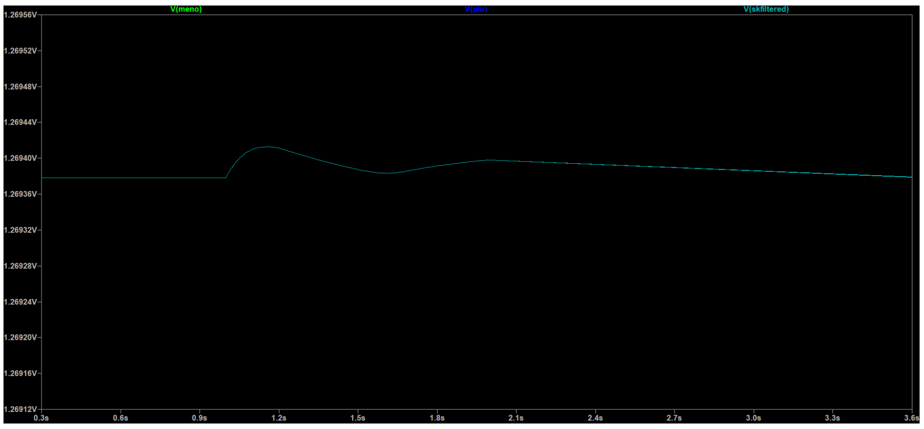


Figure 4.7: Simulation with SK filter

4.1.3 DCDC converter

We ended up our simulation analysis with the DCDC converter simulation, in our application case, when we needed 3.3V as output as the input voltage varies from 2.9V to 5V.

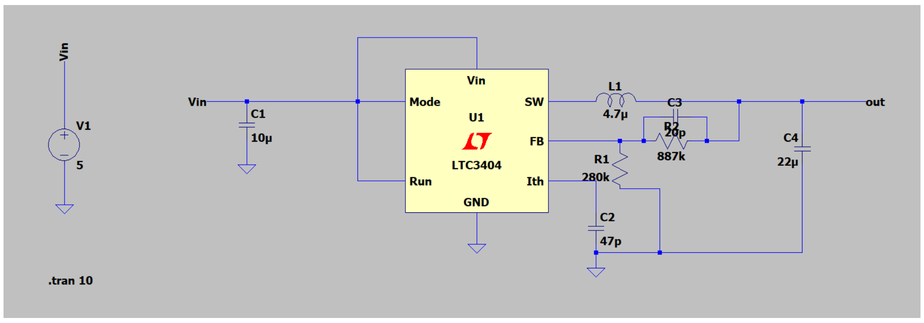


Figure 4.8: DCDC configuration

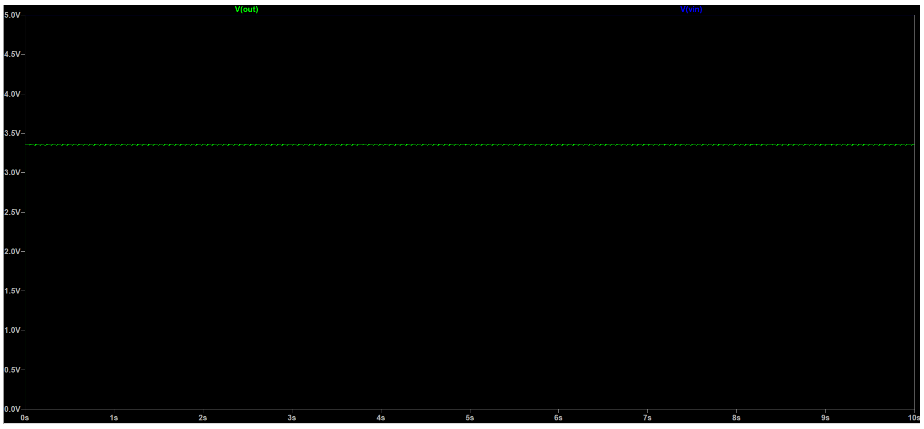


Figure 4.9: Simulation DCDC converter

4.2 Schematics

Taking a look at all the schematics, starting from the connectors and protections.

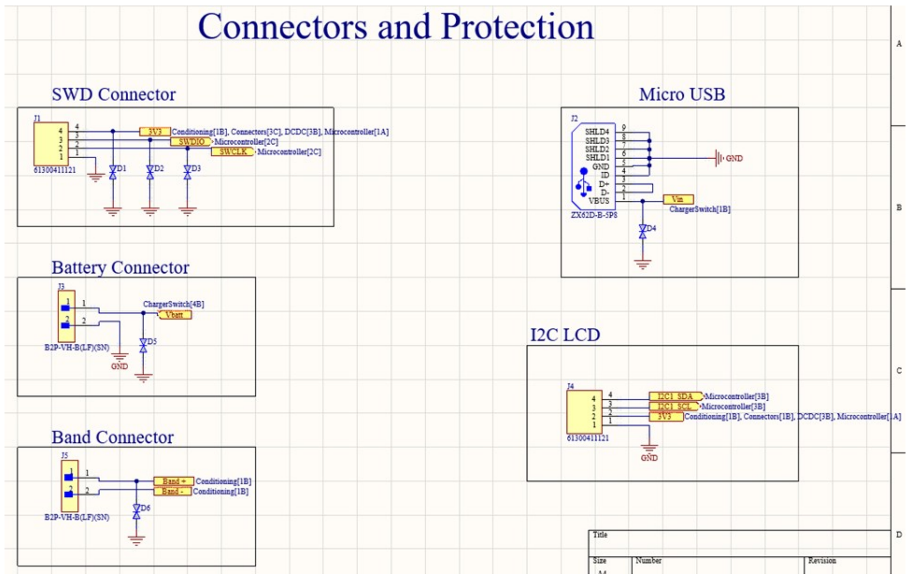


Figure 4.10: Connectors And Protections

The schematic includes several key components for connectivity and circuit protection.

- SWD connector (J1): the SWD (Serial Wire Debug) connector is a key component for programming and debugging the micro-controller. It is a 61300411121 connector with four pins, two of which are directly connected to the SWDIO and SWCLK pins of the STM32 micro-controller. This allows the micro-controller to be programmed and debugged in real time.
- Battery connector (J3): is u tilized to connect the battery to the circuit. It is a B2P-VH-B(LF)(SN) component, which enables secure connection of the battery and provides power to the system.
- Band connector (J5): used to measure the resistance change of the resistive band. It too is a B2P-VH-B(LF)(SN), and its pins are connected to the conditioning circuit to process the signal from the external device.
- Micro USB connector (J2): this is a ZX62D-B-5P8 connector used to power the circuit. SHLD pins are connected to ground to reduce electromagnetic interference. ID is connected to ground, indicating USB mode. D+ and D- pins are connected together for a two-wire USB configuration.
- LCD I2C connector (J4): this connector is used to connect an LCD display that uses the I2C protocol for communication. It is also a 61300411121 component and its pins are connected to the I2C1-SDA (Serial Data) and I2C1-SCL ((Serial Clock) pins of the micro-controller.
- Circuit protection: there are several diodes (D1, D2, D3, D4, D5, D6) that serve to protect the circuit from any overvoltages or misconnections. These diodes ensure that the circuit is protected from potential damage.

In general, this wiring diagrams page focuses on providing the connections needed to interface the micro-controller with the outside world, either for debugging and programming or for connectivity with other devices or sensors. Circuit protection is also a key aspect of ensuring the safety and durability of the system.

The circuit used for Battery Charger and Power Switch can be seen in the picture below:

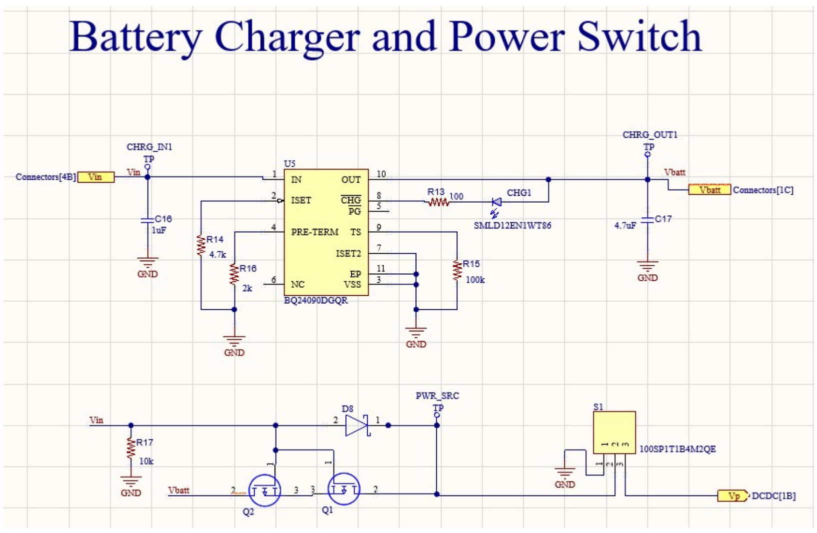


Figure 4.11: Battery Charging Circuit

The schematic details the battery charging management system and the power switch to ensure the safety and efficiency of the project's power system.

- Battery charging circuit (U5 - BQ24090DGQR): this circuit is the heart of the battery charging management system. The BQ24090DGQR is an integrated circuit that manages battery charging, ensuring safety and efficiency. The main pins include IN (pin 1) for power input, OUT (pin 10) connected to the battery, and various other pins for control and feedback of charging status, such as ISET (pin 2), CHG (pin 8), and PG (pin 5). The SMLD12EN1WT86 component used in the circuit is a TVS (Transient Voltage Suppressor) diode, which is used to protect against transient overvoltages.

- Power Switch (S1 - 100SP1T1B4M2QE): this switch is a key component for power control of the system. It has three pins: two for power control (pins 2 and 3) and one connected to ground (pin 1). The circuit uses two MOSFETs to select between the two power sources. One transistor controls the path from the battery (Vbatt) and the other from the USB connector (Vin). When the device is connected to USB power and the battery is not fully charged, the circuit activates the transistor connected to Vin, allowing this voltage to power the circuit. Conversely, if the battery is charged (or if there is no USB power available), the transistor connected to Vbatt is activated, allowing the battery to supply power to the circuit.

The transition between the two sources must be efficient and seamless to avoid malfunctions. In addition, it is essential to include protections against overvoltage and overcurrent, such as diode D8. The output is labeled Vp.

Looking at the schematic regarding the DC/DC converter:

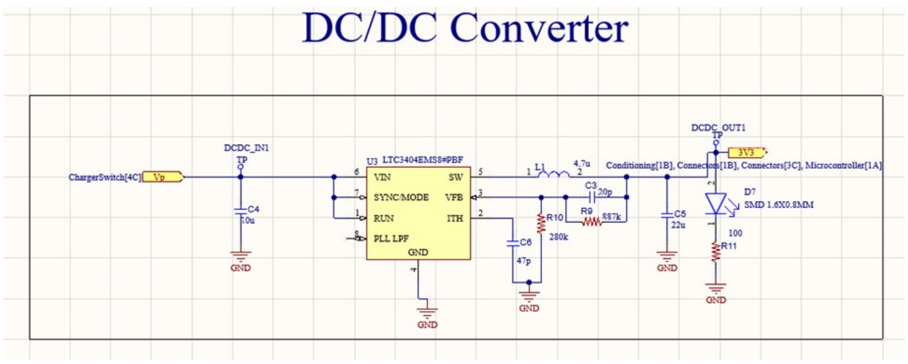


Figure 4.12: DCDC converter

This circuit is crucial for converting the supply voltage to an appropriate level for the micro-controller and other electronic components. But it has been also used for the undervoltage protection of the battery. Using a specific pouch battery, and analyzing its datasheet we discovered that 2.7V corresponds to the voltage limit, for this reason we choose a particular buck converter with an integrated comparator connected to the enable pin, in this way sending the voltage of the battery through a voltage divider to this pin the converter can be deactivated, preventing the battery from dropping below 2.9V (with a safety margin of 0.2V).

1. DC/DC converter (U3 - LTC3404EMS8PBF): this component is responsible for converting variable input voltages (3.3V to 6V) to a stable 3.3V voltage. The chip has several important pins:

    - VIN (pin 6): input of the voltage to be converted.
    - SW (pin 5): switching pin to control the conversion process.
    - SYNC/MODE (pin 7), VFB (pin 3): used for voltage modulation and feedback.
    - RUN (pin 1), ITH (pin 2), PLL LPF (pin 8), GND (pin 4): other pins for operational control and grounding.

2. Supporting Components:

    - Inductance (L1 - 4.7u): essential for the step-down conversion process, it stores and releases energy to maintain a stable voltage.
    - Capacitors: several capacitors are used to stabilize the output and input of the converter. C4 (10u) and C5 (22u) are used to stabilize the output, while C3 (20p) is used near the input pin.

- Resistors: R9 (887k), R11 (100k) and R10 (280k) are used to set specific converter operating parameters, such as feedback values and voltage limits.

3. Status LED (D7): a standard SMD LED is included to provide a visual indication of the status of the converter.

    - This circuit is critical to ensure that the micro-controller and other electronic components receive a stable and adequate power supply, which is essential for reliable and efficient operation of the device. The DC/DC converter was chosen for its efficiency, compactness, and ability to handle variations in the input supply while maintaining a stable and accurate output voltage.

In the following picture it is possible to denote the circuit used in order to: conditioning the input analog sensor, filtering unwanted noises and buffering, in order to protect the micro-controller's input ADC pins.

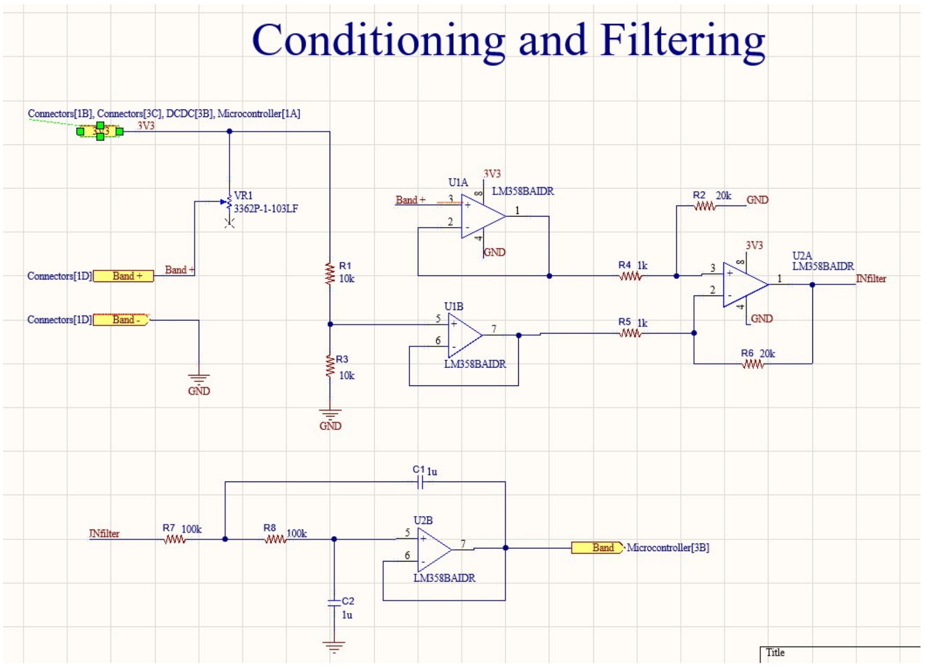


Figure 4.13: Conditioning and Filtering

The silicone band varies its resistance according to the deformations it undergoes during respiration. The output signal has a very low amplitude and a conditioning circuit is needed to make it processable by the micro-controller. This circuit amplifies and processes the output signal from the strain gauge:

- the first stage is a Wheatstone bridge to transform the resistance change into a voltage change. The 3362P-1-103LF trimmer has a maximum resistance of 10k Ohm and is a rotary trimmer, which means that its resistance can be adjusted by turning a pin or screw. It allows the bridge to be initially balanced and is necessary because the band has an initial extension that changes according to the person wearing it.

- The two LM358BAIDR operational amplifiers (U1A-B, U2A-B) amplify the input differential signal, bringing the amplitude to detectable levels with a gain of 20.
- The second-order low-pass filter formed by R7-R8 and C1-C2 with cut-off at 1.1Hz eliminates high frequencies that are not significant for the application, since the normal breathing range is 0.2Hz to 0.7Hz.
- The filtered voltage then feeds the ADC of the micro-controller.

The output voltage (Band), on the other hand, is related to the input voltage (3.3V) by the following formula:

$$Band = 3.3 \cdot \frac{R_2}{R_4} \cdot \left(\frac{R_{sg}}{R_{sg} + R_{trimmer}} - 0.5\right) \tag{4.1}$$

where the second factor represents the gain of 20. The following figure shows the output voltage as a function of the strain gauge sensor resistance, as described by the formula above. It is possible to observe that the curve is non-linear, but can be linearized around our working region (0, 3.3V), which is highlighted in green.

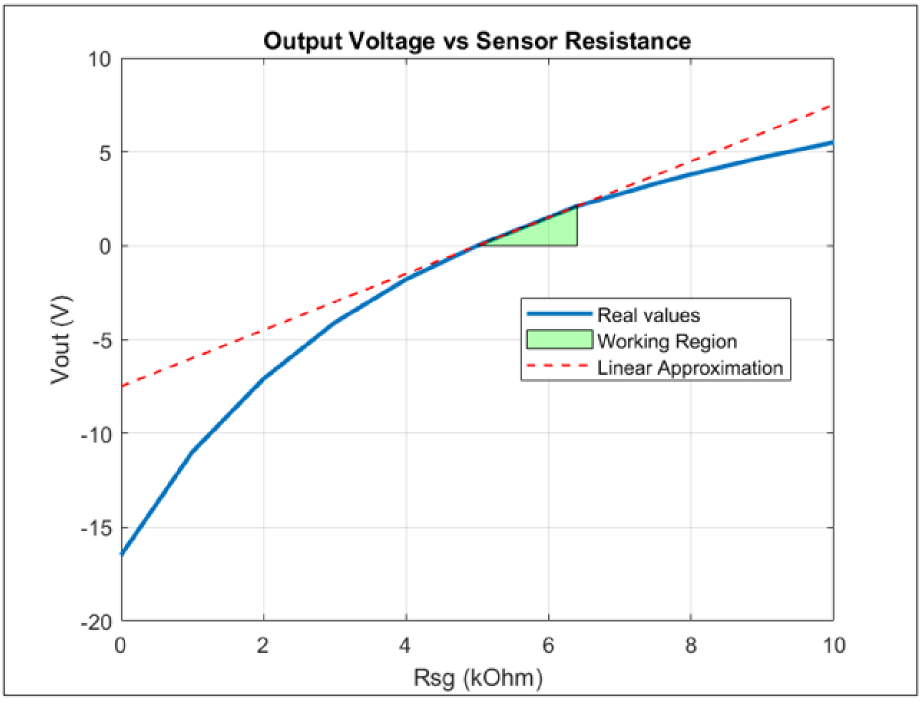


Figure 4.14: Output Voltage as a function of Strain Gauge Resistance

By combining the last plot with the calibration curve of the sensor, we get that our working region is also represented by a strain range of (0, 6cm), and for this reason we conclude that our band has a sensitivity of 0.55 V/cm. The micro-controller schematic can be seen in the picture below:

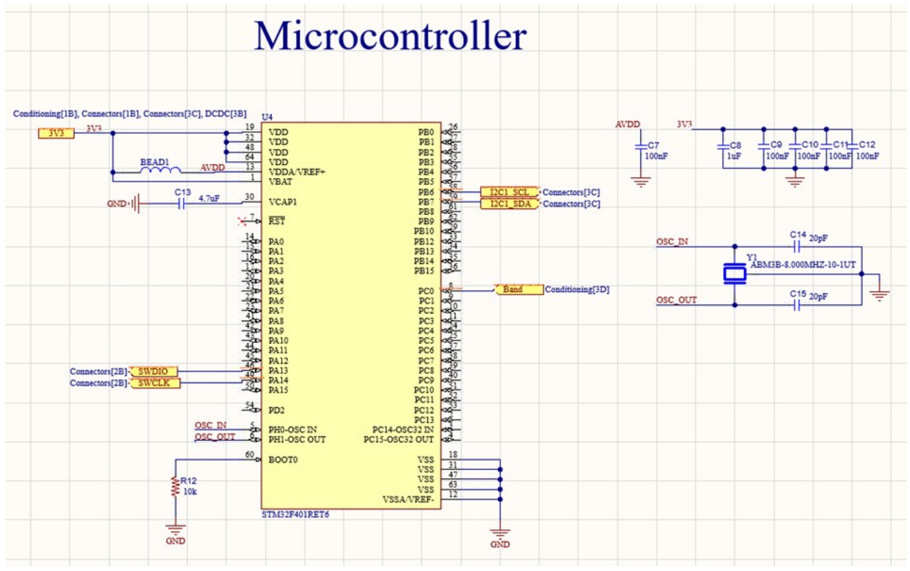


Figure 4.15: micro-controller Schematic

This section is crucial since the micro-controller is the brain of the device, handling all processing and control operations.

- micro-controller (U4 - STM32F401RET6): the STM32F401RET6 micro-controller is an advanced component with an ARM Cortex-M4 architecture, equipped with a wide range of features, including numerous I/Os, timers, and communication capabilities. Its pins are connected to various other circuit components for control and data management.
- I/O connections: the I/O pins of the micro-controller are used to interface with other circuit components, such as sensors, actuators, and communication modules. These pins include digital and analog inputs/outputs, as well as communication lines such as I2C and SPI.
- Power supply: 3V3 voltage is supplied to the VDD pins.
- Debug: this is done by connecting SWDIO and SWCLK to pins PA13 and PA14.
- External oscillator (Y1 - ABM3B-8,000MHZ-10-1UT): an external oscillator is connected to the OSC-IN and OSC-OUT pins of the micro-controller. This provides an accurate clock for micro-controller operation, which is essential for timing and control of operations.
- Stabilization capacitors: several capacitors (C7, C8, C9, C10, C11, C12, C13, C14, C15) are connected to the power and ground pins to stabilize the voltage supplied to the micro-controller, reducing noise and voltage fluctuations.

- Communication lines: the micro-controller is connected to communication lines such as I2C (I2C1-SCL, I2C1-SDA) to interface with the LCD display.
- Filtering inductor (BEAD1): an inductor (BEAD1) is connected to the VDDA pin to filter high-frequency noise from the analog power line, improving the quality of the micro-controller's analog signal.
- Signal processing from the band: the Band voltage output from the conditioning circuit is connected to the ADC of the micro-controller for further processing, thus obtaining the respiration frequency.

4.3 PCB Design

Regarding the PCB design, aside from the space optimization, which has been done only at the end of the design process, the circuit has been systematically organized, considering both functional and assembly aspects.
We have chosen a 2-layer PCB configuration, with the layer stackup consisting of TOP and BOTTOM layers, both with their own ground polygon pour.
The board is essentially divided into two distinct sections: the first part is allocated for the power components, encompassing the Charger, mosfets and DCDC. Meanwhile, the second part is designated for the signal components, housing the micro-controller, sensors, and the I2C interface. This structural division optimizes the organization and functionality of the board.
Additionally, in the corners, some LEDs have been incorporated to visually indicate important operational aspects of the board.
Certainly, during the PCB design process, meticulous consideration was given to the diverse datasheets of the various integrated circuits (ICs). Each datasheet was thoroughly examined, extracting essential guidelines for optimal placement to achieve peak performance and effective thermal management.
In addition to the careful attention given to the datasheets of individual integrated circuits, several other critical considerations were taken into account during the PCB layout process:

- Ensure that critical components, especially those sensitive to noise or interference, are adequately isolated.
- Employ proper routing techniques, such as avoiding 90-degree angles and using curved traces for better signal flow.
- Align components and traces to the grid to maintain a clean and organized layout.
- Perform DRC checks at various stages of the design to catch errors early.
- Clearly label critical components, connectors, and test points.
- Regularly save and backup your design files to prevent data loss

Referring to Figure below as a guide, the highlighted areas on the board denote the most crucial sections.

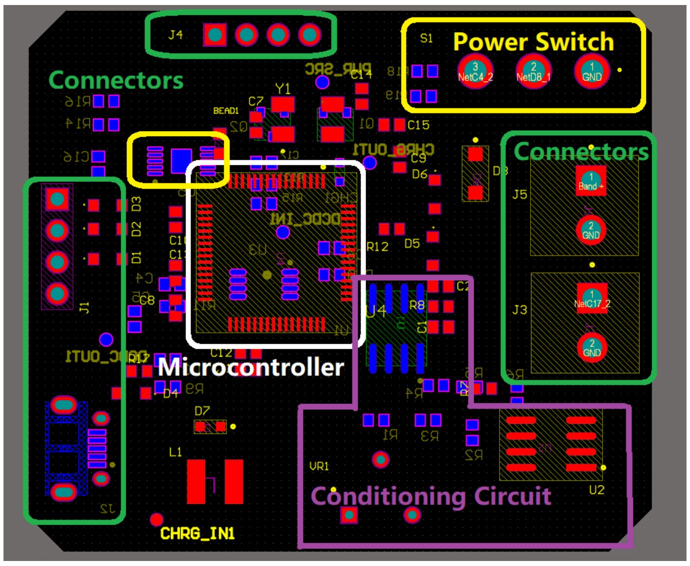


Figure 4.16: PCB Logic Division

The board measures 4.5x5 cm2. Figure shows both layers (3D).
The Top layer has the following components: micro-controller U4; operational amplifier U2; switch S1; connectors J1 (SWD connector), J3 (Battery connector), J4 (I2C LCD), J5 (Band connector); trimmer VR1; crystal Y1; test pad CHRG-IN1.
The Bottom layer houses the components: operational amplifier U1; voltage regulator U3; battery charger U5; connector J2 (micro USB); test pads PWR-SRC, CHRG-OUT1, DCDC-IN1, DCDC-OUT1.

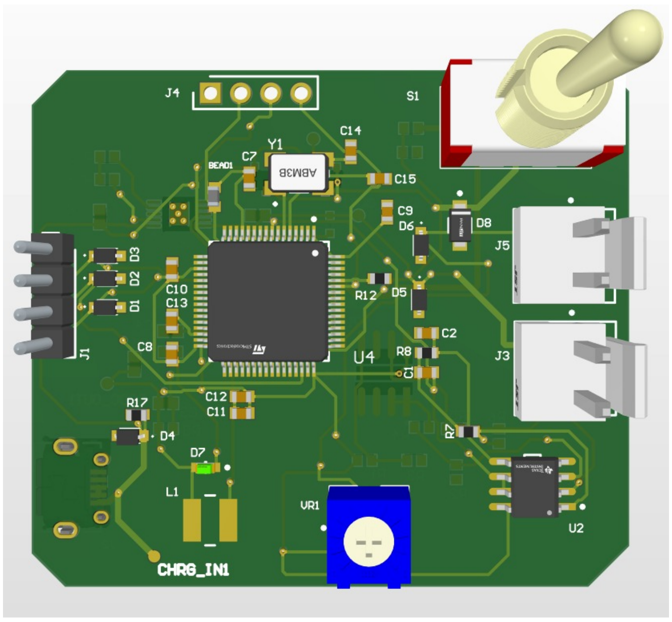


Figure 4.17: Top 3D

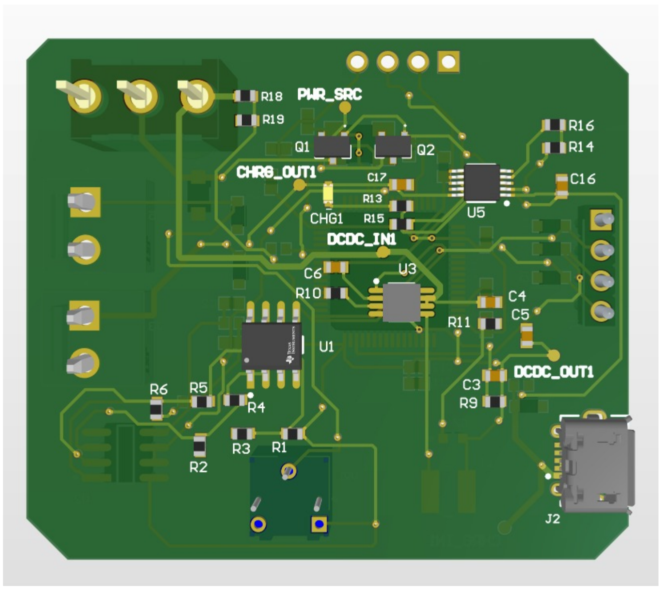


Figure 4.18: Bottom 3D

The next Figure again shows Top layer and Bottom layer in the 2D version with in addition two polygon pours, one per layer, associated with ground. This made it possible to remove several tracks and paths and thus make the board cleaner.

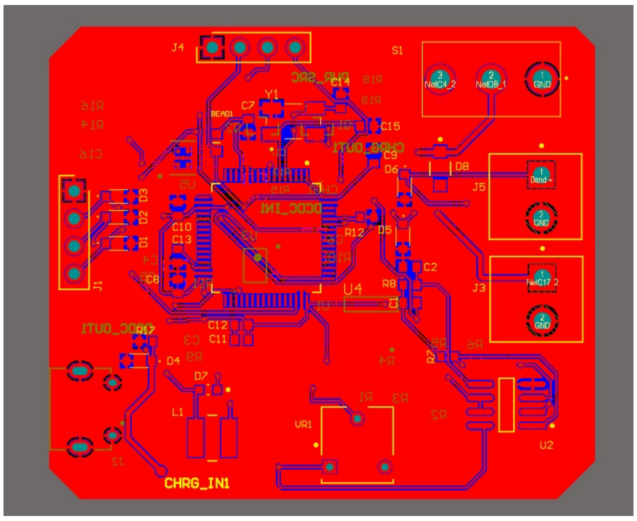


Figure 4.19: Top Design

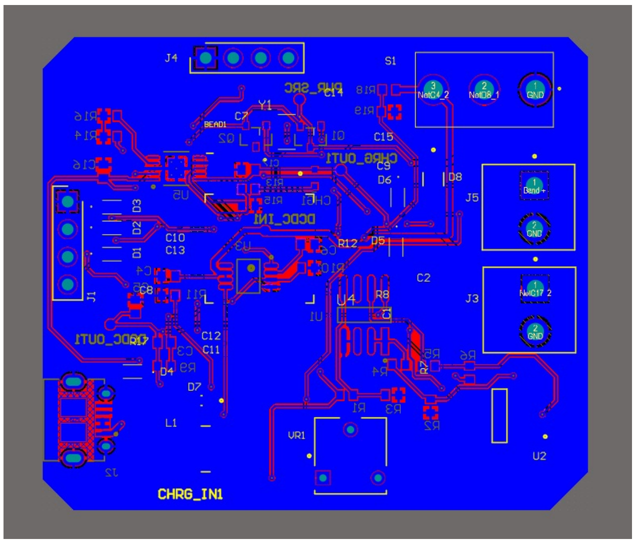


Figure 4.20: Bottom Design

The next Figure shows both layers in 2D without the polygon pours, and in particular, for each layer, the elements in the other layer have been hidden so as to have a clearer view, plus it shows them both on the same tab.

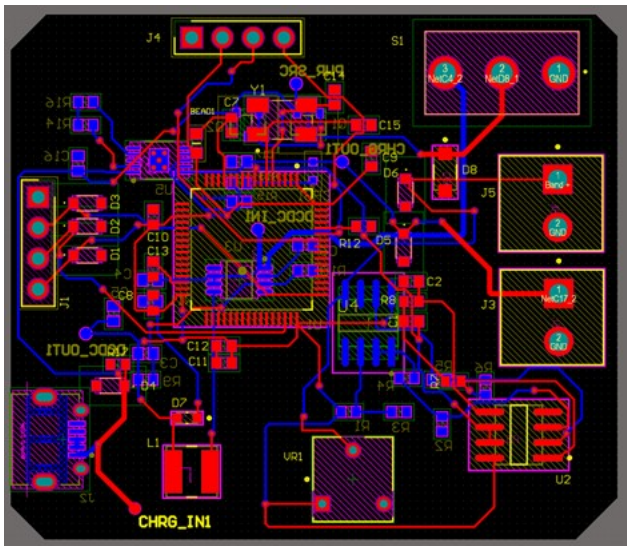
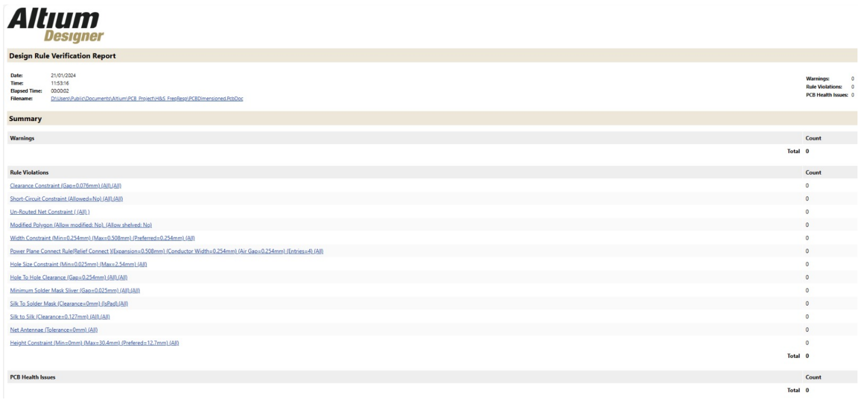


Figure 4.21: PCB design

The last figure shows that doing the Design Rules Check results in zero warnings, violations or problems, in accordance with the rules used: in particular, we complied with the SIMONE-RULES.RUL rules provided by the professor by slighly changing only two constraints, namely those related to the maximum allowed heigh for components and the maximum allowed size for tracks, due to the heigh of the switch and the fact that a common convention for the PCB is to make the tracks connected to the circuit power supply thicker, respectively.


Design Rule Verification Report

Date:	21/01/2024
Time:	11:53:16
Elapsed Time:	00:00:02
Filename:	D:\Users\Public\Documents\Altium\PCB_Project\H&S_FreqResp\PCBDimensioned.PcbDoc

Warnings:	0
Rule Violations:	0
PCB Health Issues:	0

Summary

Warnings	Count
Total	0

Rule Violations	Count
Clearance Constraint (Gap=0.076mm) (All),(All)	0
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Modified Polygon (Allow modified: No), (Allow shelved: No)	0
Width Constraint (Min=0.254mm) (Max=0.508mm) (Preferred=0.254mm) (All)	0
Power Plane Connect Rule(Relief Connect)(Expansion=0.508mm) (Conductor Width=0.254mm) (Air Gap=0.254mm) (Entries=4) (All)	0
Hole Size Constraint (Min=0.025mm) (Max=2.54mm) (All)	0
Hole To Hole Clearance (Gap=0.254mm) (All),(All)	0
Minimum Solder Mask Sliver (Gap=0.025mm) (All),(All)	0
Silk To Solder Mask (Clearance=0mm) (IsPad),(All)	0
Silk to Silk (Clearance=0.127mm) (All),(All)	0
Net Antennae (Tolerance=0mm) (All)	0
Height Constraint (Min=0mm) (Max=30.4mm) (Prefered=12.7mm) (All)	0
Total	0

PCB Health Issues	Count
Total	0

Figure 4.22: Rules

The main constraints provided by the rules used can be seen in Figure, and the reasons for each are explained below:

- Clearance constraint (Gap = 0.076mm): Adequate minimum spacing between tracks prevents short-circuits from occurring, helps ensure good isolation between adjacent tracks, reduces the risk of physical damage or wear and tear that could shorten PCB life, and facilitates component assembly by ensuring that the PCB layout is compatible with manufacturing processes and automated assembly techniques.

- Short-circuit constraint (Allowed=No): This constraint is critical to ensure the integrity and safety of the electronic circuit since short circuits can cause irreversible damage to electronic components, overheating, and in extreme cases, fires. In PCB design, it is an essential preventive measure to ensure the safety, functionality, and durability of the circuit. This parameter is especially important in high-density circuits, where the risk of accidental short circuits is greatest.

- Un-Routed net constraint: This constraint ensures that all nodes and paths provided in the circuit design are physically connected in the PCB. Without this verification, some components may remain unconnected, compromising the functionality of the entire circuit.

- Modified Polygon (Allowed modified:No): Prohibiting modifications to polygons after their initial placement ensures that the design respects the original intention. This is important to maintain the integrity of the electrical and physical layout of the PCB. In fact, accidentally or improperly modified polygons may approach or overlap traces or components, increasing the risk of short circuits or isolation problems.

- Width Constraint (Min=0.254mm) (Max=0.508mm): The Width Constraint in Altium, set between 0.254mm and 0.508mm is crucial to ensure PCB constructability, reliability and efficiency. It ensures proper current handling and signal integrity while balancing space requirements and manufacturing process limitations.

- Power Plane Connect Rule(Relief Connect)(Expansion=0.508mm) (Conductor Width=0.254mm) (Air Gap=0.254mm) (Entries=4): the Power Plane Connect Rule with Relief Connect in Altium is crucial to ensure robust and secure connections between power planes and components, balancing the needs for isolation, current management and signal integrity.

- Hole Size Constraint (Min=0.025mm) (Max=2.54mm): This range ensures that holes are large enough to accommodate component pins and small enough to maintain the structural integrity of the PCB.

- Hole To Hole Clearance (Gap=0.254mm): This value ensures that holes are adequately spaced to avoid physical and electrical interference between adjacent components. This rule is also essential to ensure adequate mechanical strength of the PCB, especially when there are multiple neighboring holes.

- Minimum Solder Mask Sliver (Gap=0.025mm): Minimum Solder Mask Sliver ensures that solder mask areas are wide enough to prevent unwanted solder bridges and ensure adequate coverage.

- Silk To Solder Mask (Clearance=0mm): This constraint ensures that the silkscreen (silkscreen) on the PCB can overlap the solder mask without clearance, improving the aesthetics and spatial efficiency of the design without compromising functionality. These rules are also important to ensure that the silkscreen is easily readable and does not confuse operators during assembly and maintenance.

- Silk to Silk (Clearance=0.127mm): Establishes a minimum distance between silkscreen lines to ensure readability and clarity, avoiding overlaps and confusion in component and track identification.
- Net Antennae (Tolerance=0mm): This constraint aims to eliminate any "antennae" (unconnected traces) in the PCB nets, which could cause electromagnetic interference or signal integrity problems.
- Heigh Constraint (Min=0mm) (Max=30.4mm) (Prefered=12.7mm): Imposes limits on the heigh of components mounted on the PCB, ensuring compatibility with the case and other mechanical components, with a preference for an average heigh that optimizes the trade-off between space and functionality.

Chapter 5

FW design

In the following section, we will comprehensively delineate the step-by-step process for developing the software, elucidating each phase of testing designed to rigorously assess and validate the accuracy and functionality of the system. The provided code is designed for an STM32 micro-controller and aims to count a user's respiration rate using an analogue sensor. The STM32 microntroller has a vaste and useful documentation aimed to the programmers and a lot of ready-to-use C-language libraries written by ST itself and many more thanks to the numerous number of users.
In order to test the SW not only on the final board itself, but also in parallel to the development of the HW, we used the NUCLEO development board, which relies on a micro from the same family of the one we chose. Together with a good documentation and precise datasheets, ST provides an enviroment for the coding design that consists of:

- CUBEMX, an IDE with built-in editor and debugger specific for the STM32 series micro-controllers
- SysConfig, a GUI that provides the PinMux and PinOut of the micro and the possibility to graphically set some configuration of ADCs, Timers, I2C communication ecc..

We then summed up the main requirements and functionalities of our micro, which are:

- Sample 50Hz the signals of the sensors through an ADC
- Transmit these data via I2C

After this first overview on the Software functionalities, we started writing some code to test separately the modules to be used, and, in parallel, to design a first sketch of the Block diagram of the firmware.

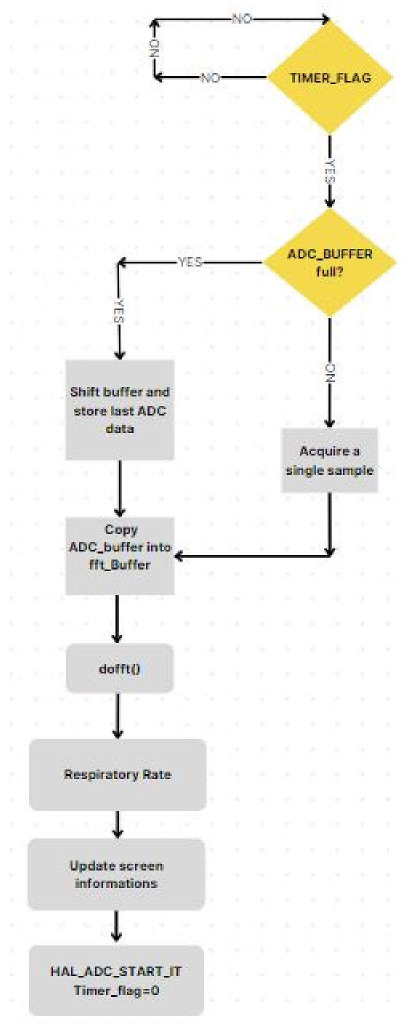


Figure 5.1: Code Block Diagram

The code revolves around the Interrupt Service Routine (ISR) of the timer, which invokes a subroutine every 20 ms. The primary purpose of this ISR controlling all micro operations by toggling a flag, granting permission to utilize the I2C interface, as well as perform necessary checks.
Initially, the transmission is disabled and the micro-controller begins by filling a buffer to execute all operations including the FFT analysis. Upon the first invocation the micro performs an ADC conversion and subsequently store the results in the buffer.
Then when the buffer is full it starts to being shifted, in order to store newer data, and sends I2C messages to the OLED display; remembering that the next conversions and transmission will occur when the watchdog triggers the ISR.

5.1 Code Description

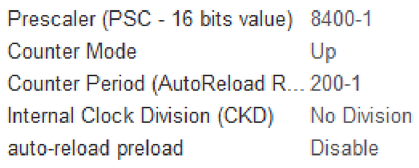
Below is a description of the main sections and functionality of the code:
Architecture of the code:

1. The architecture of the code is based on a timer, activated every millisecond, that in turn activates the ADC conversion, the result of which was inserted into a circular buffer of 1024 elements.

2. The buffer is then sent to the core function, the FFT function, which creates another buffer of the same length as the input one, inserting in it the amplitude and phase of each frequency component (512 pairs in this case).This was done according to the Nyquist-Shannon sampling theorem, so the sampling frequency had to be set to at least twice the signal frequency. For this reason the 512nd couple of values corresponded to half of the sampling frequency.

3. In the end, taking the modulus of each frequency component, the program stores the peak frequency and sends this value (converted in breaths per minute) to the OLED display.

The first important configuration to note is the counter settings:

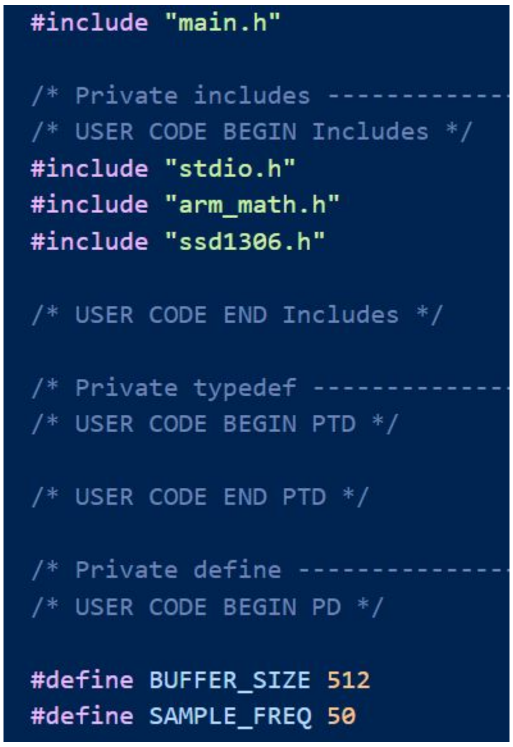


Prescaler (PSC - 16 bits value)	8400-1
Counter Mode	Up
Counter Period (AutoReload R...	200-1
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Figure 5.2: Timer

We set the counter with these values so that we have an interrupt that triggers our timer every 10 ms. This means a frequency of 50 Hz just as we wanted.Then our timer begins to count until it reaches the auto reload register (counter period) value we have chosen as 200-1. At that instant we are sure 10 ms has passed. At that point the timer resets and will start counting again until it reaches the value 200-1 again and so on.

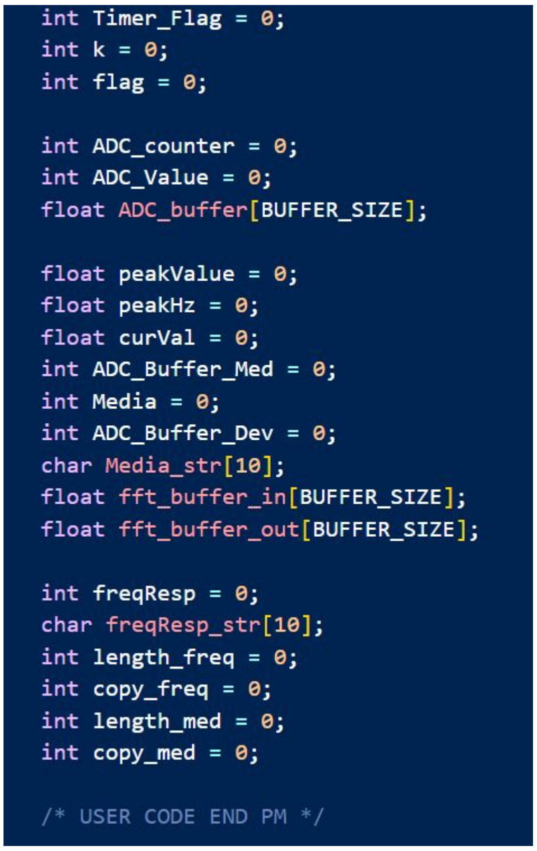
The code starts including various standard libraries for handling the microcontroller and performing DSP operations (FFT), as well as others specific to handling OLED displays via I2C.



```
#include "main.h"

/* Private includes ------------
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "arm_math.h"
#include "ssd1306.h"

/* USER CODE END Includes */

/* Private typedef -------------
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define --------------
/* USER CODE BEGIN PD */

#define BUFFER_SIZE 512
#define SAMPLE_FREQ 50
```

Figure 5.3: Included libraries

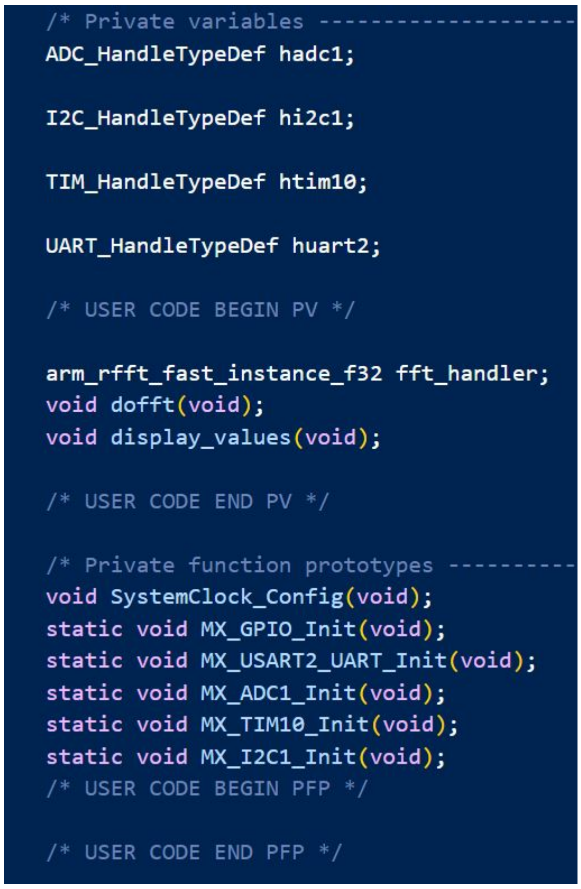
The ’arm-math.h’ library is fundamental for the implementation of the fast Fourier transform, which is a primary processing task for this project. Then some global variables are declared such as BUFFER-SIZE 512 (input buffer size to the fft), SAMPLE-FREQ 50 (sampling frequency), some variables for the LCD screen (width and height).
Looking at the image below:



```
int Timer_Flag = 0;
int k = 0;
int flag = 0;

int ADC_counter = 0;
int ADC_Value = 0;
float ADC_buffer[BUFFER_SIZE];

float peakValue = 0;
float peakHz = 0;
float curVal = 0;
int ADC_Buffer_Med = 0;
int Media = 0;
int ADC_Buffer_Dev = 0;
char Media_str[10];
float fft_buffer_in[BUFFER_SIZE];
float fft_buffer_out[BUFFER_SIZE];

int freqResp = 0;
char freqResp_str[10];
int length_freq = 0;
int copy_freq = 0;
int length_med = 0;
int copy_med = 0;

/* USER CODE END PM */
```

Figure 5.4: Variables

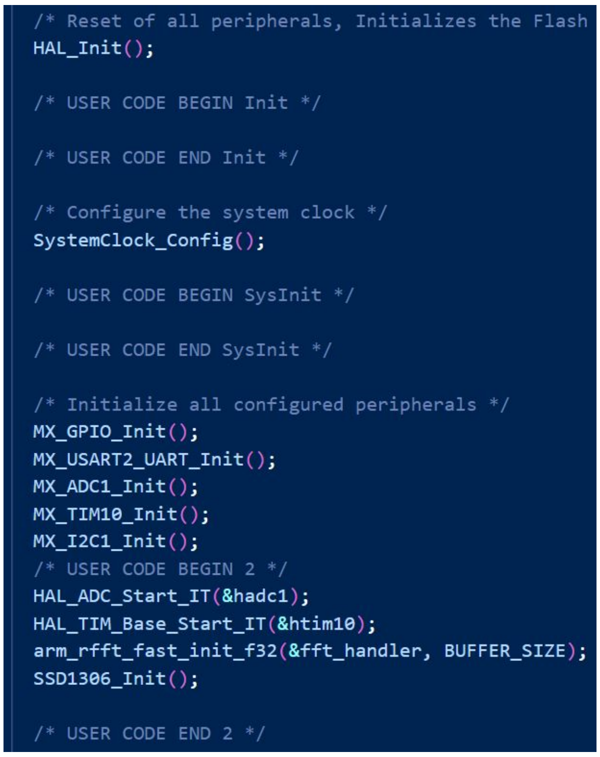
Other variables ar declared and used to store the values that the ADC returns and freqResp will contain the respiration frequency. PeakHz is the peak corresponding to the maximum frequency and it will be multiplied by 60 (see the definition of respiratory rate). Other variables useful for the operation of the program will be discussed in detail on the next pages.



```
/* Private variables --------------------
ADC_HandleTypeDef hadc1;

I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim10;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

arm_rfft_fast_instance_f32 fft_handler;
void dofft(void);
void display_values(void);

/* USER CODE END PV */

/* Private function prototypes ----------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM10_Init(void);
static void MX_I2C1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

Figure 5.5: Functions definitions

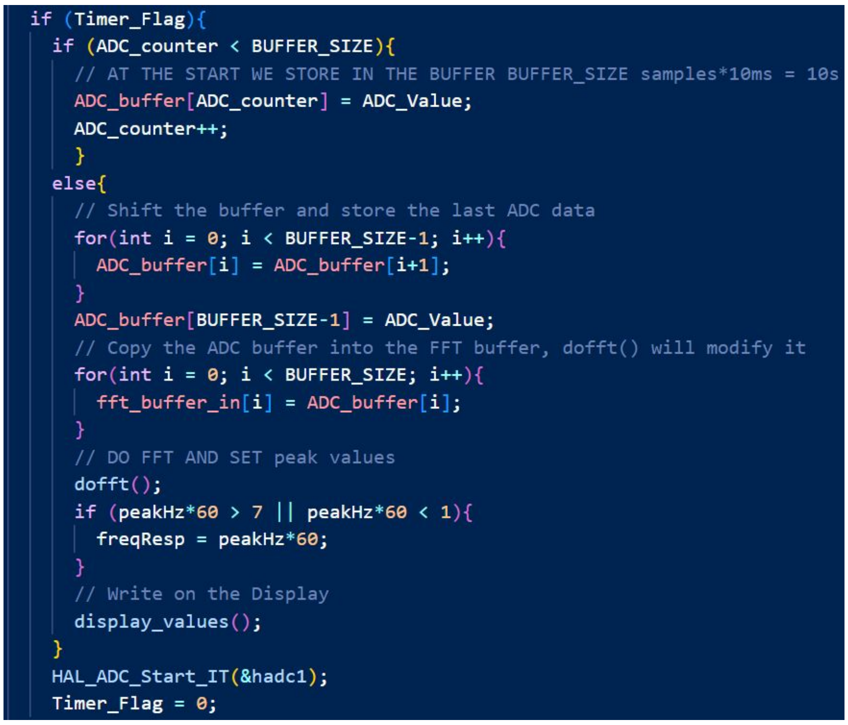
The first part of code is used to initialize the various peripherals used during the project and they are as follows: systick for interrupts, GPIO, ADC to get the data from the band, I2C to use the LCD screen, the TIM10 timer to sample at our desired sample rate.



```c
/* Reset of all peripherals, Initializes the Flash
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
MX_ADC1_Init();
MX_TIM10_Init();
MX_I2C1_Init();
/* USER CODE BEGIN 2 */
HAL_ADC_Start_IT(&hadc1);
HAL_TIM_Base_Start_IT(&htim10);
arm_rfft_fast_init_f32(&fft_handler, BUFFER_SIZE);
SSD1306_Init();

/* USER CODE END 2 */
```

Figure 5.6: Initialization

Then, the code continues with the main function, that is based on the if statement that checks if the timer flag is active:

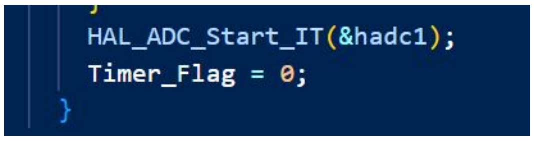


```
if (Timer_Flag){
  if (ADC_counter < BUFFER_SIZE){
    // AT THE START WE STORE IN THE BUFFER BUFFER_SIZE samples*10ms = 10s
    ADC_buffer[ADC_counter] = ADC_Value;
    ADC_counter++;
    }
  else{
    // Shift the buffer and store the last ADC data
    for(int i = 0; i < BUFFER_SIZE-1; i++){
      ADC_buffer[i] = ADC_buffer[i+1];
    }
    ADC_buffer[BUFFER_SIZE-1] = ADC_Value;
    // Copy the ADC buffer into the FFT buffer, dofft() will modify it
    for(int i = 0; i < BUFFER_SIZE; i++){
      fft_buffer_in[i] = ADC_buffer[i];
    }
    // DO FFT AND SET peak values
    dofft();
    if (peakHz*60 > 7 || peakHz*60 < 1){
      freqResp = peakHz*60;
    }
    // Write on the Display
    display_values();
  }
  HAL_ADC_Start_IT(&hadc1);
  Timer_Flag = 0;
```

Figure 5.7: Main

If it is true, it means it is time to sample some data for processing. We also check if the counter of the ADC is less than the BUFFER SIZE. If yes then we still have to fill the input data vector otherwise we shift the vector and store only the last element from the ADC. After doing this we copy the buffer of the ADC into the buffer of the FFT. The dofft() function is called (it is a function declared by us) and calculates the peakHz value. We multiply this variable by 60 and get freqResp which is exactly the number of breaths taken in one minute. After that we display freqResp on the display. Basically this is the entire elaboration needed to obtain our desired number of breaths.

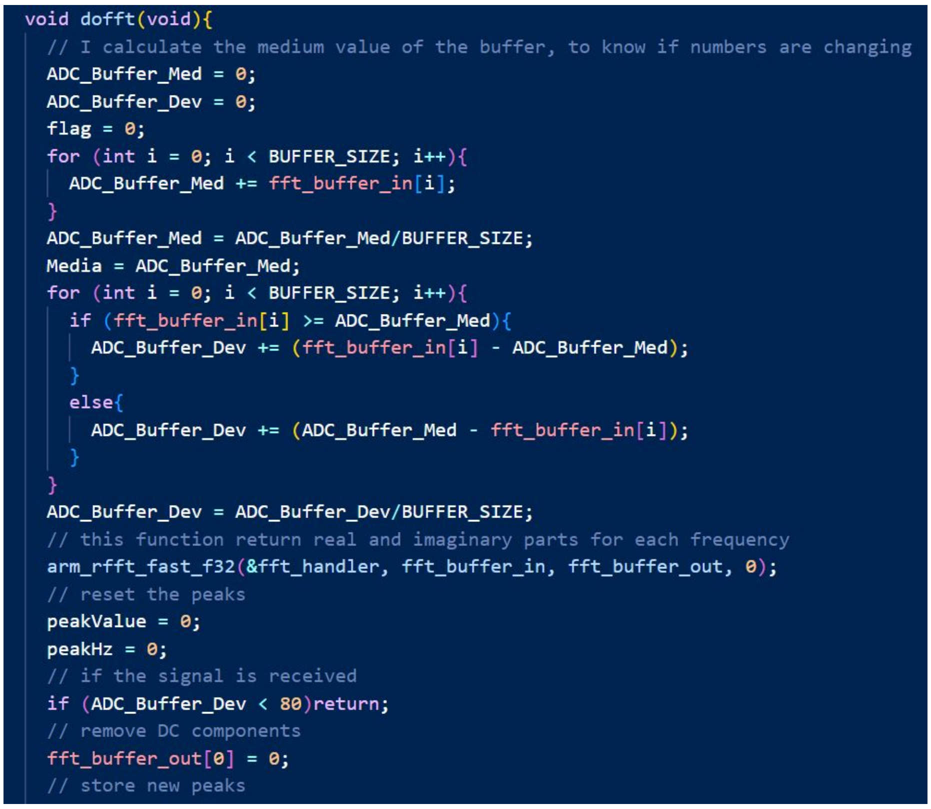
Then, restarting the ADC counter and set timer flag to zero.



```
    HAL_ADC_Start_IT(&hadc1);
    Timer_Flag = 0;
  }
```

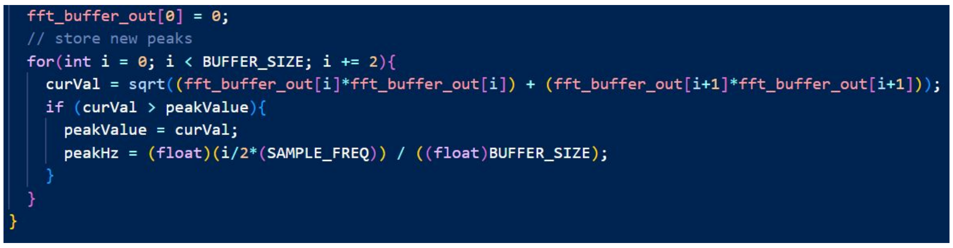
Figure 5.8: ADC Interrupt function

The main function of the code is showed below, the FFT function. We used this function to have a more compact code inside the main: in this way it is also easier to read the code.



```c
void dofft(void){
  // I calculate the medium value of the buffer, to know if numbers are changing
  ADC_Buffer_Med = 0;
  ADC_Buffer_Dev = 0;
  flag = 0;
  for (int i = 0; i < BUFFER_SIZE; i++){
    ADC_Buffer_Med += fft_buffer_in[i];
  }
  ADC_Buffer_Med = ADC_Buffer_Med/BUFFER_SIZE;
  Media = ADC_Buffer_Med;
  for (int i = 0; i < BUFFER_SIZE; i++){
    if (fft_buffer_in[i] >= ADC_Buffer_Med){
      ADC_Buffer_Dev += (fft_buffer_in[i] - ADC_Buffer_Med);
    }
    else{
      ADC_Buffer_Dev += (ADC_Buffer_Med - fft_buffer_in[i]);
    }
  }
  ADC_Buffer_Dev = ADC_Buffer_Dev/BUFFER_SIZE;
  // this function return real and imaginary parts for each frequency
  arm_rfft_fast_f32(&fft_handler, fft_buffer_in, fft_buffer_out, 0);
  // reset the peaks
  peakValue = 0;
  peakHz = 0;
  // if the signal is received
  if (ADC_Buffer_Dev < 80)return;
  // remove DC components
  fft_buffer_out[0] = 0;
  // store new peaks
```

Figure 5.9: FFT function



```c
  fft_buffer_out[0] = 0;
  // store new peaks
  for(int i = 0; i < BUFFER_SIZE; i += 2){
    curVal = sqrt((fft_buffer_out[i]*fft_buffer_out[i]) + (fft_buffer_out[i+1]*fft_buffer_out[i+1]));
    if (curVal > peakValue){
      peakValue = curVal;
      peakHz = (float)(i/2*(SAMPLE_FREQ)) / ((float)BUFFER_SIZE);
    }
  }
}
```

Figure 5.10: FFT function

First, an average is calculated both to better visualize on the screen the variation of the buffer elements and to be able to calculate the standard deviation, which is necessary to filter out unwanted variations in the signal (read by the ADC) caused by noise. Even if the resistive bandage is at rest, the noise causes variations that will be processed by the code, which will calculate the fft, amplitude and frequency. Thus, if the difference between the current sample and the average value of past samples is less than a threshold value we set, then the variations are noise and the code will not calculate the magnitude and frequency. The arm-rfft-fast-f32 function returns a vector which contains within it the real and imaginary values of the fft. So we will have in position zero the first real value of the fft, in position 1 the first imaginary value of the fft, in position 2 in the second real value of the fft etc.. The length of this vector will then be N/2 where N is the buffer size of the fft.

After resetting peakvalue and peakHz to zero, we are ready to compute the magnitude (for every value of i, so it is a magnitude calculated point by point) of the output vector of the fft. We called this valure curVal. Next we compare curVal with peakValue and if curVal is greater than peakValue, we update peakValue with the current curVal. Basically, since we consider one breath like a single sinusoidal signal, we are searching the highest value of curVal (so the highest magnitude component bin of the fft) because it is related to the highest frequency component in our signal and clearly that value is the breath rate we are trying to compute.
Last thing to do is calculate peakHz (breath frequency Hz) and we compute it in this way:

$$peakHz = \frac{i}{2}(\frac{SAMPLE_{FREQ}}{BUFFER_{SIZE}}) \tag{5.1}$$

This is the formula that allow us to derive wich is the frequency related to the chosen curVal. The meaning is the following: one bin width of the fft is calculated by dividing the sample rate by the FFT length and this is the second term of the formula. Then we have that the bandwidth of the FFT is from DC to 1/2 the sample rate and this explains why we have a division by 2.
The last thing to explain is the index i and the reason in there is because, naturally, we are calculating point by point our magnitude values.So if we find a curVal value and we want to compute peakHz we need also to multiply the formula by i in order to understand which bin we are referring to, otherwise when we could find another value of curval we would have no reference and peakHz would be the same as the previous one. The most important thing to understand is that we are trying to detect what frequency the curvalue corresponds to.

Chapter 6

Design consideration and conclusion

With the aim of producing a wearable smart band that allows the measurement of the respiratory rate, the dimension requirements for the board were very stringent, since the objective was to have the PCB with the same dimensions of the display. In order to meet the dimensional constraints, we choose to power the system with a small pouch battery, which can be recharged via a Micro USB connector, thanks to the integrated charging circuit designed in a way that serves also for the battery over-voltage protection.
All the design was also centered around the cost effectiveness, still guaranteeing small dimensions to comply the constraints. For this reason the decision of the integrated circuits was driven by the cost.
As an example the LTC4091 for charging the battery and power switching has been discarded due to its cost compared with our solution. Moreover we tried to use components we had in our laboratory, and for this reason we preferred using the LM358 operational amplifier instead of a instrumental amplifier that is the best in term of differential amplification. Since our solution was tested to be efficient using LTSpice and doesn't require additional costs, we opted for it.

6.1 Limitations and Further Improvements

The main limitation relegated to our project comprehend the fact that the band is connected to our board with two crocodiles that are very uncomfortable to use, morevore they add a noise to our measurement.
The improvement that we thought about is a little board embedded in the band and with a solid and stable connection with the band, that has only to send messages over wifi or bluetooth communication.
The frequency calculation using the FFT peak Hz to identify the frequency of respiration rely on the relationship, between the resolution (in respiration per minute), the sample frequency and the buffer size.

$$R = \frac{f_C}{2 \cdot Buffer_{size}} \cdot 60 \qquad (6.1)$$

But the FFT give rise to the peak frequency analyzing the last window of time, such window can be calculated:

$$T = \frac{Buffer_{size}}{f_C} \tag{6.2}$$

If we want to bring the resolution to 1 respiration per minute, we need to increase the buffer size or decrease the sample frequency. But, these two changes will linearly affect the window of time where the FFT is applied, creating a sort of inertia that opposes to each changes.
Using this technique we cannot do much about it, but we thought that a possible solution could be:
instead of relying only on the frequency peak we could improve our code flexibility by relying on the 10 (as an example) highest frequencies.
In this way, through an interpolation between these data we could find an approximate solution that describes better the changes and is way more reactive to frequency changes, because we could decrease the buffer size still achieving an enough good resolution, that is approximated only during the transitions between two far frequencies.
Another improvement can be done changing radically the approach to the problem: instead of collecting all the 1024 (in our case) values and then calculating the frequency peak we thought about a solution that involves only a tenth of the actual 1024 values and the other 90% of the data are predicted based on the first 10%.
The resolution can still be calculated using the same formula, but in this case the time needed to fill the buffer, and the time needed to reach the equilibrium when sudden frequency changes happens is reduced by 90%.
In the end we may think that in the future, and maybe already nowadays, approaches based on machine learning can be integrated even on a micro-controller allowing the prediction of a certain data based on a training dataset with an high accuracy regardless of the inaccuracy of the measurement.