UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

Design and application of a low-cost, easy-to-build, non-invasive, pressure-controlled ventilator for pediatric use in low-resource countries.

Barcelona, 23 January 2023

Author: Eduard Puig Bonjoch

Director & Tutor: Dr. Ramon Farré Ventura

Table of contents

Abstract

The aim of this project was to design and apply a low-cost, easy-to-build, non-invasive, pressure-controlled ventilator for pediatric use in low-resource countries. The ventilator was built using off-the-shelf components and an open-source design, with a total cost of less than €200. It is noteworthy that this ventilator is an adaptation of a previous project that was designed for adult use. In order to adapt the previous project to a pediatric use, a method to increase the respiratory rate was implemented, as children have a higher respiratory rate compared to adults, and this has been through the incorporation of a valve.

The prototype was evaluated in a bench test using an active patient simulator, which modeled the respiratory mechanics of patients with different levels of obstructive/restrictive diseases. Four respiratory systems were set for testing the ventilator, mimicking a patient with mild disease, a purely obstructive patient, a purely restrictive patient and a patient with both obstruction and restriction. The device was able to function effectively at high frequencies and was able to resolve the issue of inadequate time for breaths at high frequencies.

The results of this project demonstrate that it is possible to create a low-cost, easy-to-build, non-invasive, pressure-controlled ventilator for pediatric use in low-resource countries. The device is easy to construct, utilizes minimal complex components and can be replicated using the open-source design and materials. As a future improvement, the ventilator could be developed to also function as a support ventilator, detecting when the patient is attempting to inhale and initiating ventilation automatically when a predefined threshold pressure is exceeded.

Acknowledgments

I would like to express my sincere gratitude to Dr. Ramon Farré, the director and supervisor of this final degree project, as well as to Miguel Ángel Rodríguez Lázaro, for their invaluable support and trust throughout the duration of this project. Their guidance has been instrumental in the successful completion of this work. I would also like to extend my gratitude to the Department of Biophysics and Bioengineering at the Hospital Clínic de Barcelona for providing me with the necessary materials and facilities to conduct this research.

1. Introduction

1.1. Objectives

We currently live in a time where ventilators are an essential item for hospitals due to the Covid-19 pandemic. However, due to current commercial ventilator prices, there are countries that cannot afford these prices as they are part of low- and middle-income countries. This means that a considerable number of patients with acute and/or chronic respiratory failure cannot be adequately treated. In addition, we find an additional problem if any of these patients is a child, since the breathing rate is not the same as that of an adult, since it can reach 60 bpm, and therefore, they cannot use any respirator. If these countries already have few respirators, they will have even fewer for children adapted. Thus, the objective of this project is to design and test a bi-level, non-invasive, affordable, easy-to-build, pressure ventilator for children to enable a reduction of the severe shortage of ventilators in these types of countries.

The fact that it is a non-invasive ventilator is due to reasons of cost and ease of use. Non-invasive mechanical ventilation (NIV) is a widely used and accepted treatment for chronic respiratory diseases and apart from being an effective method, it is also a suitable approach to provide respiratory support to patients living in low-income developing economies [1].

It must be a low-cost device, since it is intended for low-income countries that cannot use the ventilators that are on the market today. The factor that it is easy to build also intervenes here, since it will be an open-source device, so that they can replicate the device in these countries autonomously.

In this way we can say that we have two main objectives, which would be:

- Carry out the design and construction of a controlled ventilator for children up to 14 years old.
- That this ventilator meets the characteristics necessary to end the shortage of ventilators in low-income countries.

To carry out these two objectives, it will be necessary to meet other objectives, which are the following:

- The device must be low cost. This refers to the material that makes it up.
- It must be easy to build, so that it can be recreated in these countries.
- It must be open source, so that these countries do not have to pay to recreate it.
- It must be non-invasive.

1.2. Project scope

What is expected of this project is that it can meet the main objectives mentioned above. As has been said before, this ventilator is intended for low-income countries, therefore we rule out its use in developed countries with large hospitals and significant investment in health. For this reason, in the countries to which this device is focused, they are mostly in Africa.

In these countries, the burden of critical illness is large and is expected to increase with increasing urbanization, emerging epidemics, and expanding access to hospitals [1]. Furthermore, as a consequence of the current pandemic caused by Covid-19, the demand for medical equipment such as mechanical ventilators has increased considerably in these countries. Mechanical ventilators are expensive, severely restricting their availability and, consequently, the ability to adequately treat a significant number of patients with acute and chronic respiratory failure [2].

Medical device donation can help provide mechanical ventilators to underserved regions, but these initiatives have been found to contain some limitations. Donating off-the-shelf equipment has been found to be partially effective, as up to 50% of given devices have been reported to become unusable due to lack of proper maintenance and inability to obtain spare parts [3].

It is due to this factor that we want to carry out this project, since in this way these countries would be the producers of the ventilators themselves (they would have access to all the detailed technical information) and would not depend on other companies that have the patent for the devices. Hence the impossibility of obtaining the spare parts.

The ventilator must have three essential components, which are: the high-pressure blower to generate the air flow, the need for pressure and flow transducers to know the pressure that is being exerted at that moment and the value of the airflow, and a controller with a digital display that will be in charge to control the magnitudes mentioned above.

Limitations and restrictions

In this project, the most important limitations would be at the cost level, because, as has been said before, it is aimed at low-income countries. The price should be below 200€ per unit. In terms of space limitations, there would not be much of a problem since the device should not be very large, more or less than 30x20x20 cm. What is necessary is that the patient is stretched out and comfortable, therefore the use of a bed could entail a spatial limitation. Finally, in terms of time, there would be a temporary limitation, since each day that passes is one more day that these countries do not have ventilators. In this project, speed is something that could help save a large number of lives and that is what interests us most.

2. Background

2.1. State of the art:

Mechanical ventilation could be defined as the treatment to help a person to breathe when they have difficulties or cannot breathe on their own. The way a mechanical ventilator works is to push airflow into the patient's lungs to help them breathe [4]. Currently there are different types of mechanical ventilators on the market. We can make a first classification between negative pressure and positive pressure respirators.

Negative pressure respirators were the first ventilators to be invented and currently their use is practically nil, as considerable progress has been made in recent years regarding this technology. Its operation is to generate a negative pressure on the outside of the chest and transmit it to the interior to expand the lungs and allow air to flow in [5]. The two types of negative pressure ventilators that exist are the iron lung and the chest cuirass ventilator. The iron lung was the first mechanical ventilator created in 1929 and was practically a metal cylinder that completely wrapped the patient up to the neck. The cuirass ventilator could be defined as a casing that was placed on the patient's chest to create a negative pressure [4].

Fig 1. Iron lung

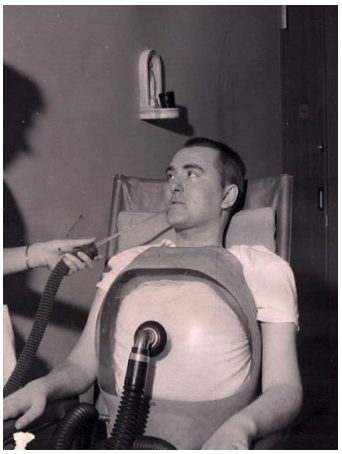


Fig 2. Chest cuirass

On the other hand, we could define positive pressure ventilators as respirators that send a flow of air to the patient's lungs through a tube. Within these we can make a classification between invasive or non-invasive. The invasive ones are characterized by the fact that the tube is inserted inside the patient through the airways. This procedure must be done in the intensive care unit of a hospital. As procedures to carry out invasive breathing we find endotracheal intubation, where the tube is inserted through the mouth or nose until it reaches the trachea, and tracheostomy, where the tube is inserted into the trachea through a direct incision in the neck.

As for non-invasive mechanical ventilators, we can try CPAP, APAP and BiPAP. These types of respirators include masks and can be used at home. CPAP provides a constant air pressure, APAP changes the air pressure according to the breathing pattern, and BiPAP provides air with different pressures for inspiration and expiration [6]. Our respirator, as mentioned before, is a two-level pressure respirator, therefore it will be a BiPAP respirator. To use this ventilator, the first step is to put on a mask that is connected to a tube attached to the ventilator. The device sends pressurized

air into the airways, and with this air pressure, the machine helps open the lungs. This is called positive pressure ventilation [7]. What differentiates it from other ventilators such as CPAP is that at the time of expiration this pressure decreases, thus allowing better breathing.

2.2. State of the situation

It should be noted that this project is an adapted project from a previous one. The only difference with the previous one is that it was designed for adults and this one is designed for children between 3 and 14 years old. In the bibliography of this work [2] you can find the article of this project where all the technical part and the tests that have been carried out so that it is accepted and can be used are detailed. The people who participated in this project were: Onintza Garmendia, Miguel A. Rodríguez-Lazaro, Jorge Otero, Phuong Phan, Alejandrina Stoyanova, Anh Tuan Dinh-Xuan, David Gozal, Daniel Navajas, Josep M. Montserrat and Ramon Farré; and the article was published last 2020.

The ventilator was built using commercially available materials through e-commerce and consisted of a high-pressure blower, two pressure transducers, and an Arduino Nano controller with a digital display. The total cost of the device was less than 75 €, so it met the low-cost goal. Details of its construction were also provided so that its replication could be carried out free of charge. The ventilator was evaluated and compared with a commercially available device (*Lumis 150 ventilator; Resmed, San Diego, CA, USA*) and the results were satisfactory. The problem arose when it was seen that due to the high respiratory rate of children, it could not be used in individuals of that age. This is the reason why it was decided to develop this project.

Fig 3. BiPAP respirator O. Garmendia et al.

3. Market analysis

Mechanical ventilators are aimed at the medical sector and there are currently several companies dedicated to manufacturing this type of device. Those that stand out are *Becton, Dickinson and Company; Phillips; Hamilton Medical; Medtronic; GE Healthcare* among others. Most of the leading companies in this field are located in Europe and the United States. It must be said that the market has evolved a lot in recent years, since the biomedical engineering sector has begun to stand out in a notorious way. As can be seen in the previous point of the state of the art, it has gone from a device where you had to put your entire body, into a respirator that you can have at home. That is why the fact that there has been an evolution in this type of technology has caused the market to evolve together. During the first wave of the Covid-19 pandemic, there were companies not oriented to the medical sector; such as *Seat, Tesla* or *Dyson*, which adapted to manufacture respirators, due to their scarcity. This tells us that this market can still evolve much further.

Regarding the standard price of a mechanical ventilator that we can find in the market, it is USD 25.000 [8], which reveals what was previously commented on the scarcity of these in low-income countries. If we focus on the low-cost ventilator market, apart from finding the aforementioned BiPAP ventilator, we find the *RESPIREM* project [9]. This project was launched thanks to the collaboration of various companies, the University of Barcelona and the Health and Public Administration. This project consists of the automation of a manual resuscitator and would also meet the objective of being economically affordable and easy to build.

Fig 4. RESPIREM project

4. Concept engineering

The main components of the respirator are: the blower that generates the pressure to provoke the air flow; the pressure transducers to measure the pressure that is being exerted and the controller with the display to, as the name indicates, control the pressure and the other indicators of the ventilator. It will also be necessary to find a method by which the frequency of breathing can be increased, since children have a greater number of breaths per unit of time.

4.1. Pressure generating blower

The blower must be able to generate an airflow in order for the patient to breathe properly. To begin with, we are interested in a blower that can be controlled by a controller device, be it Arduino, Raspberry, etc. For this we will need a blower that is controlled through a voltage, that is, more voltage, more pressure.

Apart from this feature we will also need the blower to meet the appropriate dimensions. It will have to be small, about 7x7x7 cm, since it should fit inside the structure of approximately 30x20x20 cm. Regarding the type of turbine that we will choose, in the market we find high, medium and low-pressure blowers. The pressure that we will need for the patient to breathe correctly will be maximum 30cmH_2O. It must be specified that due to the dimensions that our blower must have, when we speak of high, medium or low pressure, reference is made to blowers with these measurements, not of an industrial level, since the low-pressure industrial blowers have more power than the high pressure of the dimensions we need.

That said, since we can control the pressure of the blower through the voltage regulation, we will choose the high pressure one, since in this way we will reach 30cmH2O pressure without any problem.

4.2. Pressure transducers

The pressure transducers are essential in this device, since they are responsible for measuring the pressure that is being exerted in each moment, converting that measured pressure into an electrical signal and being able to transmit it to the controller. The transducers will also be useful to calibrate the device before starting its operation. There are two types of pressure transducers: mechanical and electromechanical.

The mechanical transducers are those that determine the pressure exerted by a liquid of known density and height. These elements are of direct or indirect measurement, but in either case they lack high sensitivity [10]. In contrast, electromechanical transducers use an elastic mechanical element associated with an electrical transducer that generates the electrical signal proportional to the supported pressure. These provide a much more accurate result than those discussed above. There are four main groups: strain gauges, piezoelectrics, resistives, and capacitives [11].

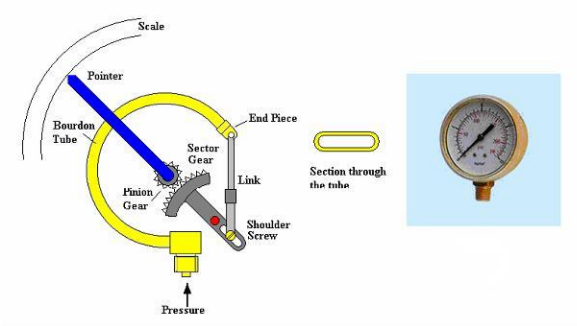


Fig 5. Mechanical pressure transducer

Fig 6. Electromechanical pressure transducer (piezoelectric)

Strain gauges are based on the change in length and diameter that a conductor undergoes when it withstands mechanical oscillations as a result of pressure. This causes the resistance of this conductor to also vary and produce what is called the piezoresistive effect. To calculate this increase in resistance, a Wheatstone bridge is used.

Piezoelectric transducers work with the accumulation of electrical charges in areas of a crystalline sheet that is formed by certain materials. This is due to withstand mechanical pressure. The glass is located between two sheets of identical metallic materials that collect electrical charges, allowing pressure changes to be measured.

Next, in the resistive ones, the pressure causes the displacement of a cursor on a resistance, acting as a potentiometer that modifies its value proportionally to the supported pressure. Finally, in the capacitive ones, pressure is exerted on a metallic diaphragm, which is a plate of a condenser, thus modifying the separation between the diaphragm and the other plate and thus causing variations in capacity proportional to the applied pressure.

Considering these options we see clearly that we will need an electromechanical transducer, since we will need the transducer to convert the measured pressure into an electrical signal. Regarding the type of electromechanical transducer that we are going to use, it could be any. The least probable would be the use of strain gauges, since the fact that a Wheatstone bridge must be designed would cause the use of more space and more time invested. As for the others, the ideal would be to carry out a series of tests to see which one is the most suitable, or since it is an adaptation of the low-cost respirator mentioned previously, we could use the same type of transducers as in this case, which were the piezoelectric.

4.3. Controller

As its name indicates, this device is responsible for controlling the pressure through the information it receives thanks to the transducers. Currently on the market there are several types of controllers, but the easiest to use and that also has a low cost is the Arduino microcontroller. Even so, there are different models of boards, each with its own characteristics. The main ones are the following [12]:

- Arduino UNO:

It is the standard board and the best known and documented. It came out in September 2010 replacing its predecessor *Duemilanove* with several hardware improvements that basically consisted of using its own USB HID instead of using an FTDI converter for the USB connection. It is a recommended board to start with electronics since it is quite intuitive. It has a voltage input of 6 to 20 V, although it is recommended that a maximum of 12 V be used. Its measurements are 69x53mm and it has 6 analog and 14 digital pins.

Fig 7. Arduino UNO board

- Arduino Mega:

This board is bigger than the Arduino UNO. It has 54 digital pins, 16 analog pins and 4 serial ports. The dimensions are 102x53mm and the one that is shared with the Arduino UNO is the input voltage that goes from 6 to 20V.

Fig 8. Arduino Mega board

- Arduino Nano:

It is the smallest Arduino board, since its dimensions are 18x45 mm. Despite being the smallest, it has 22 digital and 8 analog pins and the input voltage is the same as the others.

Fig 9. Arduino Nano board

- Arduino Ethernet:

This board has the same characteristics as the Arduino UNO, but it also incorporates an Ethernet port, which allows us to connect to a network or to the Internet through its network port.

To choose the type of Arduino microprocessor, we must also take into account the display that we will use, which is explained in detail in the next point of the report. We have already seen that each model has its own characteristics. Since the Arduino Mega has a greater number of pins and it is also the most powerful processor of the four, it will be the chosen option. In contrast to the adult respirator where an Arduino Nano was used, for this one we need a more powerful processor and with more pins, since we will have to incorporate a valve to modify the respiratory frequency and a TFT display, as outlined in the following points. The fact of using an Arduino board implies that the code to program this controller must be done with Arduino programming.

4.4. Display

In order to see the ventilator configuration values, it is necessary to incorporate a display to it. This display will need to be compatible with the Arduino microcontroller that we have mentioned above. There are two types of displays adaptable to the Arduino module on the market: LCD screens (Liquid Crystal Display) and OLED (organic light-emitting diode) screens.

LCD screens and OLED screens are both types of flat-panel displays that can be used with Arduino. However, they have some key differences [13]:

- LCD screens use a backlight to illuminate the crystals that make up the display, while OLED screens have individual pixels that emit their own light. This means that OLED screens can produce deeper blacks and a wider range of colors, but they also have a shorter lifespan and are more susceptible to burn-in.
- OLED screens are typically thinner and more flexible than LCDs, which makes them a better choice for certain applications, such as wearable devices.
- OLED screens also have a faster response time than LCDs, which means they can display fast-moving images more clearly.

- LCDs are typically cheaper than OLEDs and are more common in the market.
- OLEDs consume less power than LCDs, which makes them better suited for battery-powered applications.

Taking these characteristics into account, the chosen display will be an LCD screen, since it meets the low-cost objective. OLED screens have a greater range of colors and display images more clearly, but this is not the purpose of our screen, since this is to display the configuration parameters. Also, the fact that OLED screens are more susceptible to burn-in is quite an important factor for the choice of LCD screens.

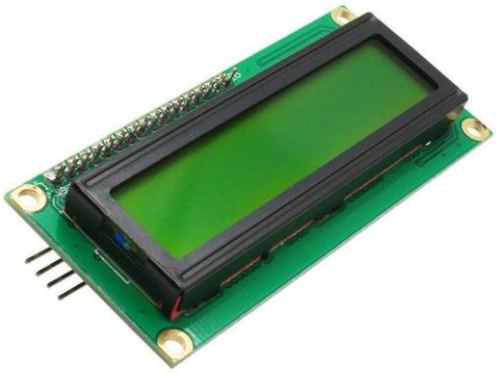


Fig 10. LCD screen

Fig 11. OLED screen

LCDs are made with either a passive matrix or an active matrix display grid. On the one hand, the passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. On the other hand, the active matrix LCD, also known as a thin film transistor (TFT) display, has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current in an active matrix display can be switched on and off more frequently, improving the screen refresh time [14]. This is why between LCDs with passive or active matrix the decision is opted for the screen with active matrix.

4.5. Adaptation of the respirator for children

Given that this respirator is specifically designed for children, certain modifications must be made to accommodate the higher breathing frequency of young individuals. One potential solution would be to adjust the parameters in the code to increase the rate of breaths per minute provided by the blower. However, this approach may prove problematic, as the blower may not have sufficient time to complete the exhalation process, resulting in a lack of adequate breaths at the frequency appropriate for children. Given that the breathing frequency of children can reach up to 60 bpm, it is deemed necessary to create a valve to regulate the airflow, in order to facilitate a faster exhalation and thus increase the overall frequency. In light of this, several types of valves that could potentially be implemented in this device are outlined below.

4.5.1. Ball valve

This type of valve is a stopcock mechanism that serves to regulate the airflow. It is characterized by a regulating mechanism located inside, which has the shape of a perforated sphere. The valve opens by turning the shaft attached to the sphere, allowing the passage of air when the hole is aligned with the inlet and outlet of the valve. When the valve is closed, the hole will be perpendicular to the inlet and outlet. This type of valve can be two-way or three-way. Two-way valves typically have a standard bore, while three-way valves allow for easy disassembly of the ball and centerpiece elements, making it easy to clean sediments and replace deteriorated parts without having to disassemble the elements that connect to the valve [15]. In addition, this valve is usually manual, but in our case, it would be connected to a servomotor that is in charge of opening and closing it at the right time.

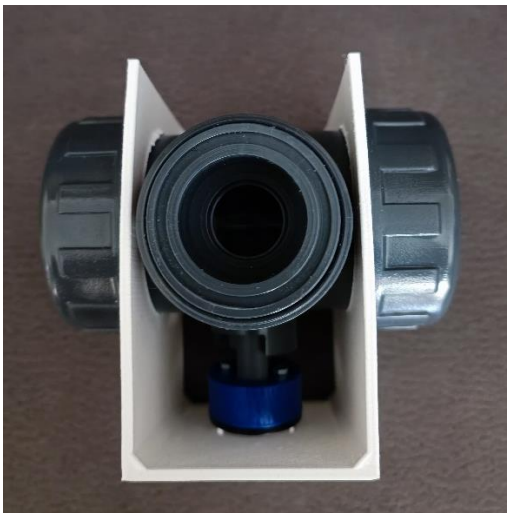


Fig 12. Ball valve

4.5.2. Leakage control valve

The function of this valve is to control the release of air during exhalation, resulting in faster expiration and resolving the issue at hand. The valve is composed of a tube with a hole, which is covered during inhalation by a servomotor connected to the mechanism responsible for covering the hole and therefore control the leak. The goal is to adjust the opening of the hole based on the frequency of breathing, in order to release the appropriate amount of air for the desired number of breaths per unit of time.

This type of valve is not currently available on the market and must be designed and 3D printed. During testing, various hole shapes were experimented with, as shown in figures 10 and 11. Initially, round and elongated round holes were tested (figure 13), but it was found that too much air was released at once. Therefore, the valve seen in figure 14 was created, featuring a triangular-shaped hole. This design allows for minimal air release at first and the servomotor can open the valve further if needed, releasing the air more regularly.

Fig 13. First version of valve with leak control

Fig 14. Final version of valve with leak control

4.5.3. Pinch solenoid pneumatic valve

A pinch solenoid pneumatic valve is a type of valve that uses a solenoid to control the flow of compressed air in a pneumatic system [16]. The valve typically consists of a solenoid-operated actuator, which is connected to a pinch mechanism. The pinch mechanism is responsible for opening and closing a tube through which the compressed air flows. When the solenoid is energized, the actuator moves the pinch mechanism to pinch the tube, effectively stopping the flow of air. When the solenoid is de-energized, the actuator releases the pinch mechanism, allowing air to flow through the tube.

Pinch solenoid pneumatic valves have several advantages over other types of valves. They are relatively simple and easy to control, as they only require an electrical signal to operate. They are also very reliable and have a long service life. Pinch solenoid valves are also very compact and can be easily integrated into existing pneumatic systems. They are also very easy to maintain and repair, as they have very few moving parts. Additionally, pinch valves are typically very versatile. Despite they are widely used in pneumatic systems for industrial automation, such as in assembly lines, packaging machines, and other industrial processes, we could adapt it to the ventilator [17].

Fig 15. Pinch solenoid pneumatic valve

4.5.4. Evaluation of the valves

After testing each of the valves, it has been decided that the best option is the pinch solenoid pneumatic valve. The ball valve has been ruled out because the servomotor had to exert too much force to open and close the valve and this caused it to break. These valves utilize a lubricant to minimize friction and promote smooth operation of their components. However, these valves are intended for infrequent opening and closing, not a frequency of 60 cycles per minute. As a result, excessive usage, as exhibited by the ventilator's characteristics, results in rapid consumption of the lubricant, leading to hardening of the valve mechanism and ultimately failure of the servomotor.

As an alternative, the leakage control valve was considered to be a viable option due to its ability to address the issue of hardened mechanism and its cost-effectiveness compared to the ball valve. However, upon conducting multiple evaluations, it was determined that the valve exhibited a slight leakage between the sealing component and the aperture when in the closed position. This resulted in issues during the inspiration phase and thus, it was ultimately discarded as a solution.

Finally, it was determined that the pinch solenoid pneumatic valve was the most suitable valve for use in a ventilator system. The pinch valve possesses several advantageous properties, such as a lack of leakage, a high level of durability, and ease of control. Additionally, the use of a servomotor, as employed in other valve systems, is not required as the pinch valve is connected directly to the electrical circuit of the device. However, the pinch valve does have one notable drawback, which is its relatively high cost, with prices ranging around 100€.

4.6. Enclosure

It is also necessary to create an enclosure to place all the elements mentioned above in an organized way. It is also necessary to create an enclosure to place all the elements mentioned above in an organized way. Since most of the parts used in this project are the same as in the adult ventilator, we can use the enclosure of the previous ventilator as a first version. Therefore, the support will be made up of two rectangular pieces printed in 3D printing, which fit together and in which the components are fixed and well placed. Despite this, because we must include the valve to increase the frequency, the dimensions will be larger and the placement of the components will have to be redistributed.

4.6.1. 3D printing programs

To begin with, the essential feature of the program that we will use to design and print the ventilator support is that it must be free, since it must meet the objective of low cost. The other characteristic is that it cannot have a high level of complexity due to the objective that the construction of the ventilator must be able to be carried out by anyone. After the research, some of the best 3D printing software found are: 3D Builder, SketchUp, OpenSCAD and 123Design.

The 3D Builder program is a free 3D modeling application that allows you to view, create, and customize 3D objects. The great feature of 3D Builder is that it can be used by any user, with or

without experience in 3D modeling. This program supports the most important 3D printing file formats: STL, OBJ, 3MF, etc. Although it fulfills the two main objectives, this program has a drawback. This is that it is only available for Windows, so someone with a device running MacOS or Linux software would not be able to use it [18].

Another possible program would be SketchUp. This program is available for other operating systems apart from Windows. It is completely free and, thanks to its easy use, it is also suitable for beginners. If required, the software also offers the option to access a library of free 3D models. Alternatively, you can also create your own model, by which you can use a variety of paint, measure and offset tools with surface modeling software. It is worth mentioning that considering these characteristics, this program would be a good option for the creation of support [18].

Another 3D printing software would be OpenSCAD. OpenSCAD free molding software is an open-source CAD program that creates 3D models from scripts. With this program, complex 3D models can be created from simple geometric bodies. The problem is that for this, the user must become familiar with the programming language, since at first it may seem complicated. Due to this inconvenience, this will not be the chosen program.

Finally, the last option would be the 123Design program. This is a program very similar to SketchUp and is based on Autodesk Inventor. In addition to the most basic drawing and modeling capabilities, it also has assembly and constraint support and STL export, so you can 3D print. That said, we see that the two best options are the SketchUp and 123Design programs. The decision factor that has led to choosing 123Design is that it was the same program used in the ventilator for adults and had very satisfactory results.

5. Detail engineering

In this section, the specific characteristics of the respirator designed in this project are thoroughly outlined. A comprehensive examination of each software component, including the blower, pressure transducers, controller, display, and valve, is provided. Additionally, the software created through Arduino programming and the design of the enclosure through the 123Design program are discussed. Furthermore, the integration process of all the various components of the ventilator is explained, along with the results obtained and a thorough analysis thereof.

5.1. Hardware

5.1.1. High pressure blower

As previously stated, this project represents an adaptation of a previous project in which the ventilator was designed for adult use. As part of this adaptation process, it was determined that the incorporation of a pinch valve was necessary in order to adjust the frequency of the respirator to the range specified for pediatric use. This was necessary as the blower alone was not capable of achieving the required cycles per minute, which can reach up to 60. As a result, the component responsible for generating pressure, the blower, may be the same model as the one utilized in the adult ventilator previously developed, WM7040-12/24V-65W blower. The detailed characteristics of this component, as provided by the manufacturer's website, can be seen in figures 16, 17, and in Table 1. The cost of this component is approximately 50€.

Fig 16. Blower dimensions

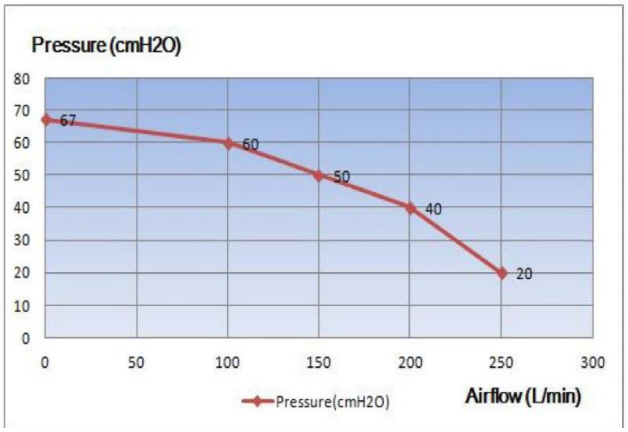


Fig 17. Graph of the pressure levels that can be reached

Voltage and Current	12V (4.5A±0.1A), 24V(2.7A±0.1A)
Max Air Flow (Air pressure=0)	240L/min, 14m³/h, 8.5 CFM
Max Pressure (Air flow=0)	7.5 Kpa, 75 cmH_2O
Size	700mm*40mm
Power	65 W
Speed	35000±5% rpm

Noise	45dB (1Kpa-1M), 73dB (7Kpa-1M)
Certification	CE

Tab 1. Blower characteristics

The data displayed illustrates the relationship between pressure and flow for this specific blower unit under different power supply conditions of 12V, 15V, and 24V.As can be observed, when powered at 15V, the blower unit exhibits a maximum pressure of 30 cmH_2O, indicating that this voltage is sufficient for the operation of the ventilator.

These pressure-flow figures give an understanding of the blower unit's performance as a pressure source. The greatest pressure that can be produced by the blower is when the output is completely blocked (zero flow). As the output is gradually opened, the flow increases and the pressure generated by the blower decreases. For instance, when the blower is powered at 15V, the highest pressure that can be generated is 30 cmH_2O. Notably, for flows of up to 100 ml/min, the pressure drops to 25 cmH2O. On the other hand, when powered at 12V and 24V, the pressures generated for the same flow rate are 17 cmH2O and 66 cmH2O, respectively. These results suggest that by increasing the power supply to 24V, the blower unit could potentially produce pressures appropriate for intubated, mechanically ventilated patients [19].

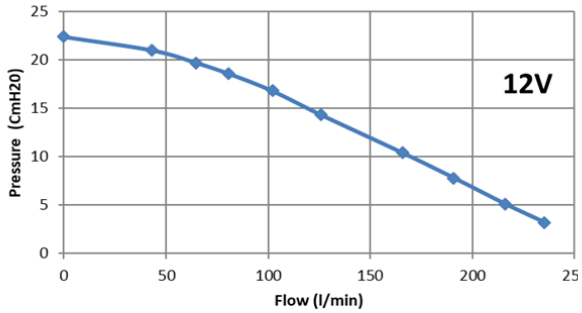


Fig 18. Pressure-flow graph of the blower at 12V

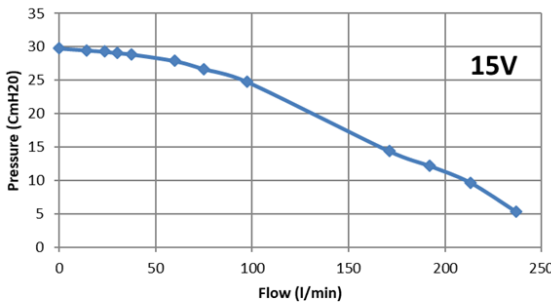


Fig 19. Pressure-flow graph of the blower at 15V

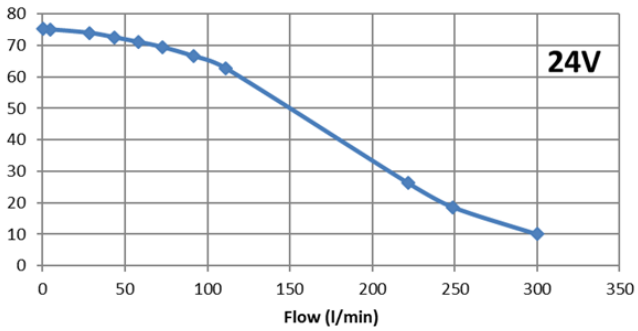


Fig 20. Pressure-flow graph of the blower at 24V

In order to ensure the proper operation of the blower, it is imperative to incorporate a driver controller. A driver controller is needed to use a blower with an Arduino because the blower motor typically requires more power than the Arduino can provide on its own. The driver controller regulates the power supplied to the motor, ensuring that it runs at the desired speed and does not overload the Arduino. Additionally, a driver controller also provides additional features such as PWM control and current sensing, which allow for precise control of the blower motor and protection

against overcurrent. The fact that it is used to control the blower motor is essential in our case, as it enables modifications to the desired pressure, frequency, and other related parameters. The driver chosen to control the blower is driver 7040, which technical parameters are shown in Annex 1.

Fig 21. Driver 7040

5.1.2. Pressure transducers

As previously outlined in the concept engineering phase, the pressure transducers selected for this project utilize piezoelectric technology. In contrast to the previous ventilator design, which utilized only two pressure transducers, the current design requires the incorporation of four transducers, with two of them being differential. This modification is necessitated by the incorporation of two pneumotachographs for measuring flow, as opposed to the single pneumotachograph utilized in the ventilator for adults.

In the previous ventilator, flow measurement was achieved through the subtraction of signals obtained from two non-differential pressure transducers, with one of them being utilized for pressure feedback. However, with the incorporation of the valve in the ventilator for children, the utilization of four non-differential transducers for measuring flow and two additional transducers for pressure feedback would have been required. By utilizing differential pressure sensors, the number of transducers required for measuring flow is reduced to two, thus reducing the total number of transducers to four.

One of the key benefits of differential pressure transducers is their ability to measure low pressure ranges with high sensitivity. This is due to their design which utilizes a diaphragm or a bellows to sense the pressure difference, allowing them to detect even small changes in pressure. Another advantage of differential pressure transducers is their ability to reject common-mode noise, which can occur when measuring pressure in systems with multiple pressure sources. This makes them more reliable and accurate compared to non-differential pressure sensors.

The pressure transducers utilized in this design are the XGZP6847 model, which is the same as the transducers used in the previous adult ventilator design. The XGZP6847 is a silicon pressure sensor module that provides a ratiometric analog interface for measuring pressure over a specified range. It has a built-in Application Specific Integrated Circuit (ASIC) which is used to compensate for offset, sensitivity, temperature and non-linearity, providing repeatability, linearity, stability and sensitivity. It is suitable for high-volume applications in various fields such as medical equipment,

fitness machines, home electronics, and other pneumatic devices [20]. The principal features of the XGZP6847 pressure sensor module are the following ones:

- The specified pressure range for this device is -100kPa to 1000kPa, with a full scale accuracy of ±1.0%.
- It is suitable for measuring non-corrosive gases or dry air.
- The output is a calibrated and amplified analog signal.
- The sensor is temperature compensated for use within the range of 0ºC to +85ºC.
- The device is intended for direct application in various systems and is low-cost.

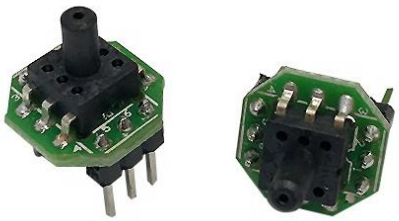


Fig 22. XGZP6847 pressure transducer

It is important to note that various models of this pressure transducer exist, each with distinct pressure measurement ranges. In this particular application, the XGZP684705KPG model will be employed, which has a measurement range of 0 to 5 kPa. This model was selected as it is suitable for measuring a maximum pressure of 30 cmH_2O while providing an acceptable level of sensitivity. The conversion of 5 kPa to cmH_2O is approximately 50 cmH_2O, thus the selected model's range of 0-5 kPa is sufficient for the measurement of 30 cmH_2O pressure. The cost of this component is approximately 4€ per unit.

In regards to the differential pressure transducer selection, the XGZP6897A transducer has been chosen for its capabilities. Unlike non-differential pressure transducers, this model is capable of measuring a pressure range of -100kPa to 200kPa and is equipped with temperature compensation within a range of 0°C to 60°C. The XGZP6897A differential pressure transducer is available in multiple models with varying pressure ranges. For the specific application of the ventilator, the XGZP6897A005HPDPN model has been selected due to its pressure measurement range of -500 Pa to 500 Pa, corresponding to an approximate range of -5 cmH2O to 5 cmH2O, which is sufficient to meet the flow measurement requirements of the system [21]. The cost of this transducer is about 5€ per unit.

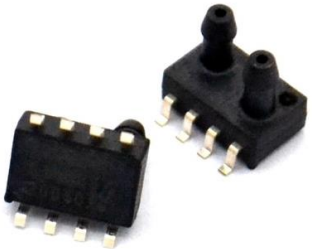


Fig 23. XGZP6897A pressure transducer

In annex 2 and 3 are detailed the performance parameters of XGZP6847 and XGZP6897A pressure transducers, respectively.

5.1.3. Arduino MEGA Controller

As mentioned in the concept engineering section, the controller chosen for this device is Arduino Mega, specifically the model ArduinoMEGA2560. Below are enumerated the key characteristics of this microcontroller with a schematic of the layout of the board components [22][23]. This board can be purchased online for around 5€.

- Microcontroller: ATmega2560
- Digital I/O Pins: 54 (15 PWM capable)
- Analog Input Pins: 16
- UART (Hardware Serial Ports): 4
- Operating voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limit): 6-20V
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Clock Speed: 16 MHz
- USB Connection
- Power Jack
- ICSP Header
- Reset Button
- Length: 101.52 mm
- Width: 53.3 mm
- Weight: 37 g

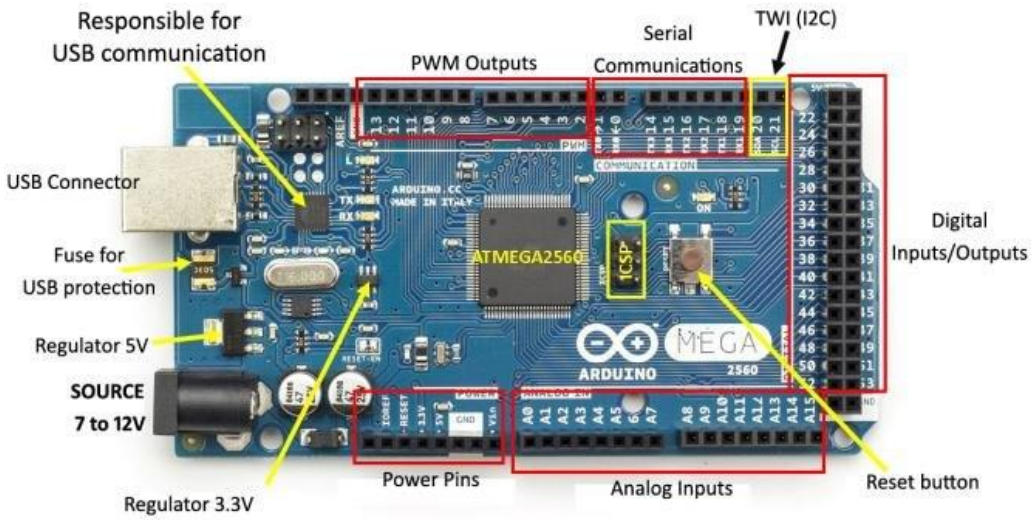


Fig 24. Layout of ArduinoMEGA2560

5.1.4. TFT Display

TFT displays are a popular choice for use with Arduino boards due to their high resolution and color capabilities. These displays use a TFT (thin-film transistor) screen to display images and graphics, and are controlled by an integrated circuit (IC) that communicates with the Arduino via a parallel or serial interface. This is another reason why this display has been chosen. The fact that the ventilator can show the pressure and flow vs time graph is a good improvement respect to the other ventilator.

TFT displays come in a variety of sizes and resolutions, with some even supporting touch input. The specific TFT display chosen for the ventilator application in this case is a 3.5 inch display with a resolution of 480x320 pixels and is compatible with the Arduino Mega2560 microcontroller. It is worth noting that in this particular application, the display does not require touch functionality as the ventilator parameters are adjusted through potentiometers. However, it is worth mentioning that the price difference between non-touch and touch modules is minimal and is around 2€. Below are the component technical details [24]:

- Display Color: RGB 65K color
- SKU: MAR3513
- Screen Size: 3.5 inch
- Driver IC: ILI9486
- Resolution: 480*320 Pixel
- Module Interface: 16-bit parallel interface
- Active Area: 48.96x73.44 mm
- Module PCB Size: 60.30x96.60 mm
- Back light: 6 chip HighLight white LEDs
- Operating Temperature: -20ºC~60 ºC
- Storage Temperature: -30 ºC~70 ºC
- Operating Voltage: 5V/3.3V
- Power Consumption: TBD
- Product Weight: about 49g

Fig 25. TFT display 3.5 inch compatible with ArduinoMEGA2560

5.1.5. Pinch solenoid pneumatic valve

The pinch valve of choice to adapt the adult ventilator to children up to 14 years of age is the ASCO™ Series S307 Pinch solenoid valve. ASCO is a leading manufacturer of valves and fluid control products for a wide range of industries, including power generation, oil and gas, chemical and petrochemical, pharmaceutical, biotechnology, and more. This valve is part of ASCO's line of pinch valves, which are known for their ability to provide accurate and consistent flow control in a wide range of industries.

The Series S307 Pinch Solenoid Valve is designed for use with a variety of fluids, including water, air, and gases. It is available in a wide range of sizes, materials, and configurations to meet the specific requirements of different applications [25]. The valve is also equipped with a variety of options and accessories, including manual overrides, position indicators, and explosion-proof enclosures. One of the key features of the Series S307 Pinch Solenoid Valve is its ability to provide precise flow control. The valve is equipped with a pinch mechanism that can be adjusted to provide a wide range of flow rates, from a complete shut-off to full flow. This allows for accurate and consistent flow control, ensuring that the correct amount of fluid is delivered at the right time.

The Series S307 Pinch Solenoid Valve is also designed for durability and reliability. It is constructed with high-quality materials that are resistant to corrosion and wear, and it is designed to have a long service life with minimal maintenance requirements. In addition, the Series S307 Pinch Solenoid Valve is available with a range of actuators and control systems, including air-piloted, solenoid-piloted, and mechanically-operated options. This allows for easy integration with existing systems and control networks, providing customers with a complete solution for their flow control needs [26].

The decision to utilize the valve, despite its relatively high cost (estimated at approximately 100€), was made based on a comprehensive analysis of the factors mentioned above. These factors included the valve's ability to provide precise flow control, its durability and reliability, and its compatibility with the specific requirements of the project. Ultimately, it was determined that the benefits offered by the valve, such as its precision and longevity, outweighed the additional expense.

Figure 15, shown at the engineering point of design, shows the ASCO™ Series S307 Pinch solenoid valve and in Annex 4 there is a summary of the technical aspects of the valve.

5.2. Software

The ventilator system has been implemented using the Arduino development platform, version 1.8.57.0. The Arduino platform utilizes its own proprietary programming language, based on the high-level programming language Processing, which exhibits similarities to the C++ programming language. The code for the implementation of the ventilator system, including all necessary libraries, can be found in Annex 5 of the technical documentation. Even so, the algorithm is described in the points below.

5.2.1. Import of libraries and definition of variables

At the beginning of the code, it is crucial to import the libraries that will be utilized in the development of the program. These libraries include TFT_HX8357.h for configuring the display, and PID_v1.h for the implementation of the PID control algorithm. The inclusion of these libraries is necessary for the proper functioning of the system, as they provide the necessary functions for the display and control of the ventilator. The PID algorithm uses three parameters: Proportional, Integral, and Derivative. The Proportional term calculates an error between the desired value and the current process value, the Integral term sums up the error over time and the Derivative term calculates the rate of change of the error. By adjusting the gain values of these parameters, the controller can achieve the desired response in the system being controlled [27]. Después de importar las librerías es necesario definir los pines de cada una de las señales, indicando si se tratará de una señal digital o analógica, y todas las demás variables.

5.2.2. Initial offset adjustment

Once the initial library imports have been completed, the next step in the development process is to configure the calibration routine for the sensors utilized in the ventilator system. It is important to note that the blower should not be in operation during the calibration process. The code is designed to read the signals from the pressure transducer at zero pressure and digitally correct any offset that may be present. In case of error during the calibration process, the routine is repeated until the calibration is successfully completed. Additionally, during this process, a conversion from the sensor's native units to cmH_2O is also executed to ensure accurate measurements. The calibration process is crucial to the proper functioning of the ventilator system and should be executed with care and precision to ensure accurate measurements and control.

5.2.3. Configuration of the display

After configuring the calibration of the ventilator, the next step is to create the graphical representation of the pressure and flow on the display. To accomplish this, the TFT_HX8357.h library, imported earlier, is utilized. The first step is to determine the interval in microseconds that is used to plot the pressure and flow pixels.. This interval is defined using the following expression:

*Tx = -347.21*bpm + 25000;*

The bpm value represents the frequency, which can be adjusted by a potentiometer and varies between 12 and 60, as said before. The minimum value of Tx (bpm = 60) is 4167, and the maximum value (bpm = 12) is 20833. Therefore, if a bpm of 12 is selected, the display will plot one pixel every 20833 microseconds.

This interval allows the display to expand or contract in order to display a couple of cycles on the screen, regardless of the bpm setting. The complete code for the creation of the graph and its explanation can be found in Annex 5 of the technical documentation.

5.2.4. Configuration of the main parameters

The main parameters of this ventilator which can be modified according to the needs of the patient are the following:

- Frequency: Refers to the number of breaths that the ventilator delivers to the patient per minute. It is also known as the respiratory rate. In our case, the ventilator has a range of 12 to 60 bpm.
- Inspiratory pressure: Refers to the amount of pressure that the ventilator generates to deliver air into the patient's lungs. The range of this parameter is of 4 cmH_2O to 25 cmH_2O.
- Inspiration/Expiration ratio: Refers to the proportion of time spent delivering inspiration to the patient, compared to the time spent delivering expiration to the patient. The I/E ratio is usually expressed in the form of a fraction, such as 1/2, which means that for every 1 second of inspiration, the ventilator delivers 2 seconds of expiration.
- Flow cycling: Refers to the way in which the ventilator delivers breaths to the patient. A flow-cycled ventilator delivers a set flow of air to the patient, regardless of whether the patient is actively inhaling or not. The flow rate is determined by the setting on the ventilator, which is usually expressed as a percentage of the maximum flow rate. For example, a ventilator set to deliver a flow rate of 50% would deliver half of the maximum flow rate of air to the patient during each breath. The % flow cycling is an important parameter to set in the ventilator, as it affects the tidal volume delivered to the patient, that is the amount of air that enters the patient's lungs with each breath. The range is between 30 and 50%.

At this point the valve variable is also configured, 0 implies that it is in expiration mode and 1 in inspiration mode.

5.2.5. Pressure and flow measurement

The measurement of pressure and flow is obtained continuously via the utilization of pressure transducers. Unlike adult respirators, in this case, it is not necessary to apply a filter to eliminate noise when calculating the flow, as the noise generated by non-differential pressure transducers is minimal. To calculate the flow, both inspiratory and expiratory, in units of L/s, it is necessary to perform a conversion by applying a constant k1 and another constant k2, as specified in the sensor data sheet. With the calculation of the flow in L/s, the inspiratory and expiratory volumes can be derived and displayed on the ventilator screen.

Once the sensors have been calibrated, the flow and pressure measurements begin to be accumulated and while being saved, they are also plotted on the screen, using the TFT_HX8357.h library, previously imported. The pressure transducers provide a continuous measurement of the pressure and flow, allowing for real-time monitoring and control of the patient's ventilation.

5.2.6. Ventilation start

Once all the parameters have been set, including the frequency, inspiratory pressure, inspiration/expiration ratio and flow cycling, the ventilation process can commence. The turbine generates a positive pressure according to the previously defined parameters. During the inspiration phase, the valve is programmed to be open, allowing air to enter the lungs. Conversely, during expiration, the valve is programmed to be closed, resulting in a faster pressure drop. The mechanical ventilator operates utilizing a PID controller, allowing for real-time modification of the parameters, with the PID control algorithm responsible for implementing these changes. During ventilation, the following parameters are displayed on the ventilator's screen:

- Pressure vs. time graph
- Flow vs. time graph
- Respiratory rate (breaths per minute)
- Inspiration/Expiration ratio
- Inspiratory pressure (cmH2O)
- Inspiratory volume (L)
- Expiratory volume (L)

5.3. Enclosure 3D printed

The dimensions of this enclosure exceed those of the adult ventilator, as the valve measures 100mm x 40mm x 50mm. As a result, the fan designed for pediatric use has dimensions of 275mm in width, 100mm in height, and 203mm in depth. The enclosure consists of two parts: a base where all components are housed and a top lid that serves to cover them, as depicted in Figure 26.

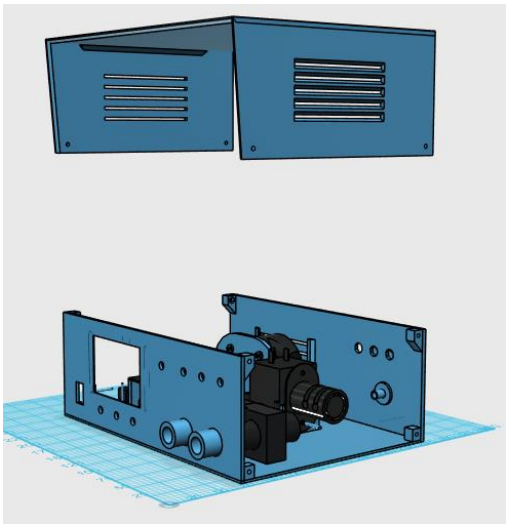


Fig 26. Enclosure of the ventilator

Three views of the enclosure are depicted below: front and back elevations, and the profile view. These views indicate the components located within each hole or their function. As can be observed in the front elevation view, there is a button labeled "Control/Support." The objective of this project is to create a controlled fan, however, a future improvement for the fan would be to adapt it to also

function as a support. This is why the button is included, so that in the future the same fan can have two modes. In this view, we can also see the location of the potentiometers that will be used to regulate various parameters (frequency, pressure, I/E ratio, and flow cycling). On the other hand, in the rear elevation view, we can see the output of the channels that provide the corresponding voltage for pressure, inspiratory flow, and expiratory flow. The profile view and the rear elevation view also show a grid. This grid serves to prevent overheating of the components, thereby improving their operation and safety.

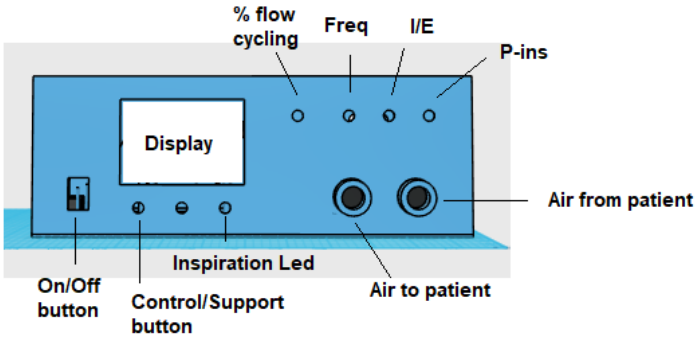


Fig 27. Front elevation view

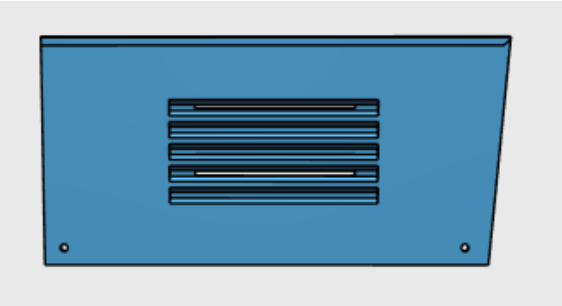


Fig 28. Profile view

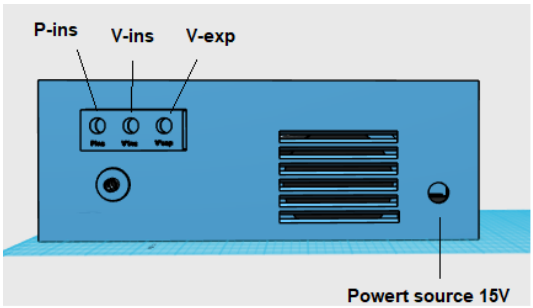


Fig 29. Back elevation view

5.4. Integration and results

Upon integration of all components, it became apparent that utilizing a PCB (printed circuit board) would be highly beneficial, as it eliminated the need for multiple wiring connections. After the creation of the PCB and the subsequent electronic connections of the components, functional testing was conducted using software. Once all issues were resolved, the results were collected for analysis and to draw conclusions for the project

5.4.1. Integration of Hardware components

A PCB (printed circuit board) is a board made of insulating material, such as fiberglass or plastic, with conductive pathways etched onto its surface. These pathways connect different electronic components together, allowing them to communicate and function as a cohesive unit [28]. There are several advantages to using a PCB, including [29]:

- Reduced Size and Weight: PCBs allow for the compact and lightweight design of electronic devices. By eliminating the need for bulky wiring, PCBs can make electronic devices smaller and more portable.
- Improved Reliability: PCBs eliminate the risk of loose connections or short circuits that can occur with traditional wiring methods. The conductive pathways on a PCB are etched into the board, ensuring a stable and consistent connection between components.
- Increased Efficiency: PCBs allow for a higher degree of integration between electronic components, resulting in a more efficient and streamlined design. This can lead to faster processing speeds and improved performance in electronic devices.
- Cost-effectiveness: PCBs can be mass-produced at a relatively low cost, making them an affordable solution for both small-scale and large-scale electronic projects.
- Easy Assembly: PCBs make electronic assembly much easier, as components are soldered directly to the board, reducing the need for complicated wiring. This makes it easy to upgrade, modify or debug the circuit.
- Repeatability: PCBs can be designed with the same layout, which allows for the production of multiple identical devices, ensuring consistency in performance and functionality.
- Safety: PCBs are designed to meet safety standards, ensuring that the electronic devices are not harmful to the users, also the components are protected from environmental factors.
- Flexibility: PCBs are versatile and can be used in a wide range of applications, from simple electronic devices to complex systems, including embedded systems, automation, and robotics.

Overall, PCBs offer many advantages over traditional wiring methods, this is why it has been chosen for this project. A PCB board has to be designed as it is unique to the device used. There are several steps involved in creating a PCB:

- Design the PCB layout: This step involves creating a layout of the PCB using PCB design software. The layout includes the placement of the electronic components, the routing of the conductive pathways, and the location of any necessary connectors or power connectors.
- Generate the Gerber files: Gerber files are used to communicate the PCB layout to the manufacturer. These files include information on the PCB's dimensions, the location of the components, and the routing of the conductive pathways.
- Order the PCB: Once the Gerber files are ready, the PCB can be ordered from a manufacturer. The manufacturer will use the Gerber files to create the PCB according to the specified design.
- Solder the components: After the PCB is received, the electronic components are soldered onto the PCB according to the layout.
- Testing: Once the components are soldered, the PCB is tested to ensure that all connections are correct and the components are functioning properly.

Figure 30 shows the layout of the fan PCB board and below, in the other figures, the connection diagrams of the different components.

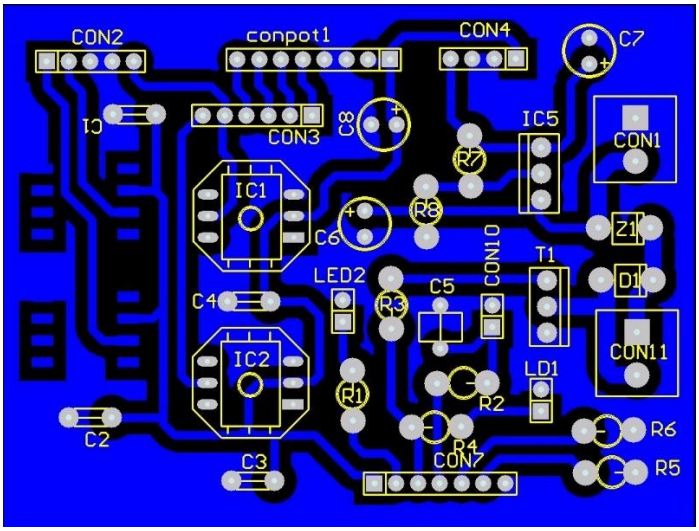


Fig 30. PCB board layout

In figure 31 we can see the connections of the analog signals with the ArduinoMEGA2560 board. Each of the pins corresponds to the following components:

- A2: IPap potentiometer
- A3: Flow cycling potentiometer
- A4: Frequency potentiometer
- A5: I/E ratio ponentiometer
- A6: Pressure sensor 1
- A7: Pressure sensor 2
- A12: Flow sensor 1
- A13: Flow sensor 2

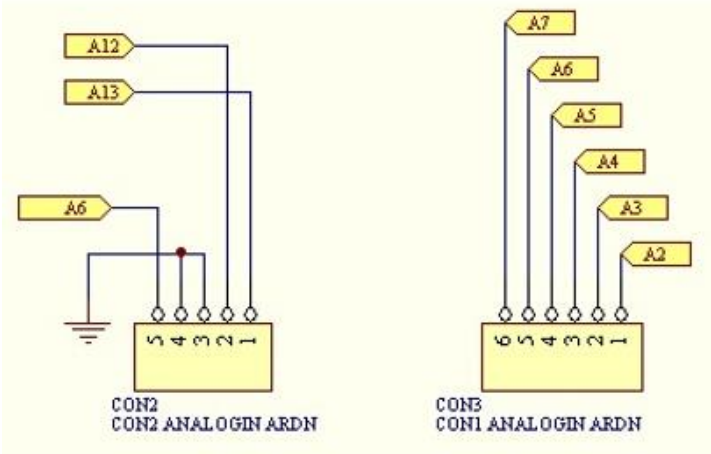


Fig 31. Diagram of the analog pin connections

Next, in figures 32 and 33 we see the connections of the flow and pressure transducers, respectively. In the diagram of the pressure transducers we can also see the connections with the valve that is controlled by digital pin 10.

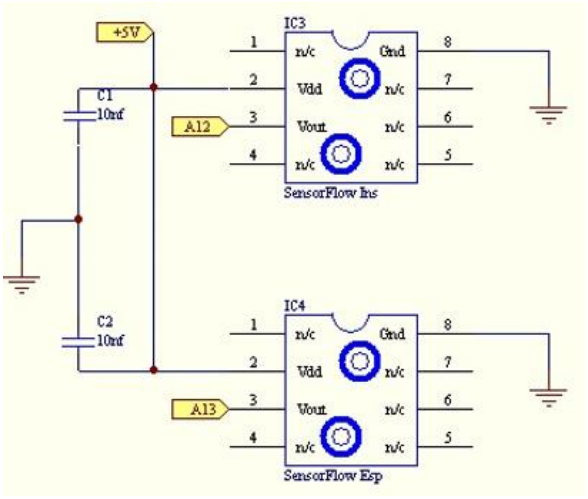


Fig 31. Diagram of the flow sensor connections

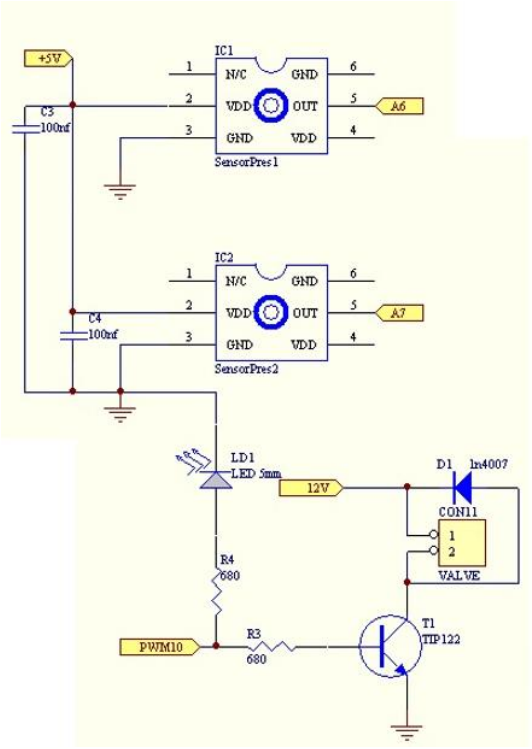


Fig 32. Diagram of the pressure sensor connections

The figures shown below, 33 and 34, refer to the connections of the digital pins with the Arduino board and the connection for the blower control. Each of the pins corresponds to the following components:

- PWM8: Inspiration led
- PWM10: Valve
- PWM11: Blower
- PWM12: SwitchPin switch
- PWM13: Control/Support switch

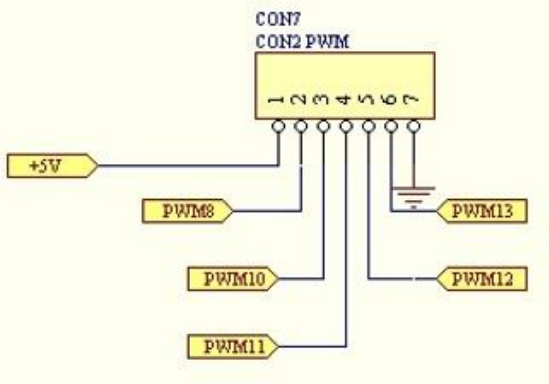


Fig 33. Diagram of the digital pin connections

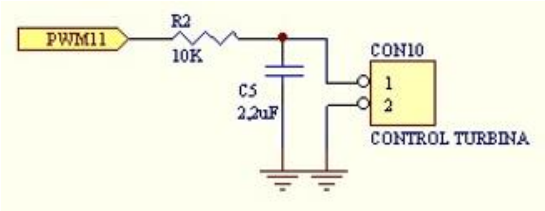


Fig 34. Diagram of the blower connections

Finally, the last two figures shown, 35 and 36, show the connections for powering the Arduino from the 7040 driver and the connection to turn on the led at the moment of inspiration.

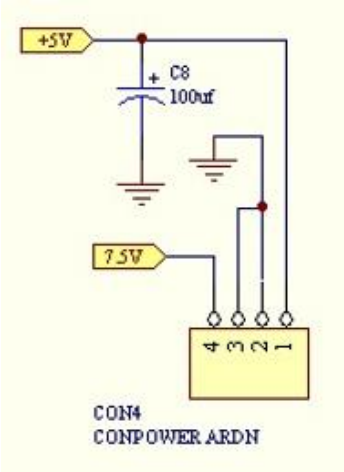


Fig 35. Diagram of the Arduino power supply connection

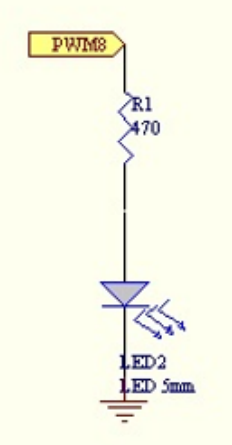


Fig 34. Diagram of the blower inspiration led connection

5.4.2. Results

To evaluate the performance of the controlled pressure ventilator for pediatric use under controlled conditions, a prototype was subjected to bench testing using an active patient simulator that simulated the respiratory mechanics of patients with varying levels of obstructive/restrictive diseases. The passive component of the respiratory system model utilized was a variable resistance-compliance (R-C) lung model (Adult SmartLung; IMT Analytics, Buchs, Switzerland), similar to the adult ventilator [2]. Four R-C systems were configured for testing the ventilator, simulating a patient with mild disease, a purely obstructive patient (increased resistance), a purely restrictive patient (reduced compliance) and a patient with both obstruction and restriction (Table 2). Three breathing frequencies were employed and different inspiratory efforts were set according to the level of disease. Despite typical respiratory rates in children ranging from 0-14 years being 12-60 bpm, in this case, testing was conducted at higher frequencies (30, 40, and 50 bpm) as it is these frequencies that can potentially cause problems to the ventilator operating. The parameters of flow cycling and the inspiratory-to-expiratory (I/E) ratio were maintained constant during the testing process. The flow cycling was set to 50% and the I/E ratio was also set to 50%. To measure

the results, two pressure transducers were used, one to measure inspiratory pressure and the other to measure pleural pressure and a flow transducer.

Simulated Patient	Resistance $cmH2O \cdot s \cdot L^{-1}$	Compliance $mL \cdot cmH2O^{-1}$	Breathing rate $breaths \cdot min^{-1}$	Inspiratory pressure cmH_2O
Mild				
1	5	30	30	9
2	5	30	40	9
3	5	30	50	9
Obstructive				
4	20	30	30	10
5	20	30	40	10
6	20	30	50	10
Restrictive				
7	5	15	30	14
8	5	15	40	14
9	5	15	50	14
Obstructive and restrictive				
10	20	15	30	16
11	20	15	40	16
12	20	15	50	16

Tab2. Respiratory resistance–compliance systems of 12 different conditions simulated for the bench test

After conducting tests with simulations of various patient types, it was determined that the low-cost, easy-to-construct, non-invasive, pressure-controlled ventilator for pediatric use functions effectively at high frequencies. The pressure vs. time and flow vs. time graphs for conditions 4 and 12 are depicted below. These graphs demonstrate that the issue of inadequate time for breaths at high frequencies has been resolved through the incorporation of the valve. As a result, the ventilator can be utilized at high frequencies, such as those observed in children. In Annex 6 and 7 the graphs of pressure-time and flow-time of all the simulated conditions are shown.

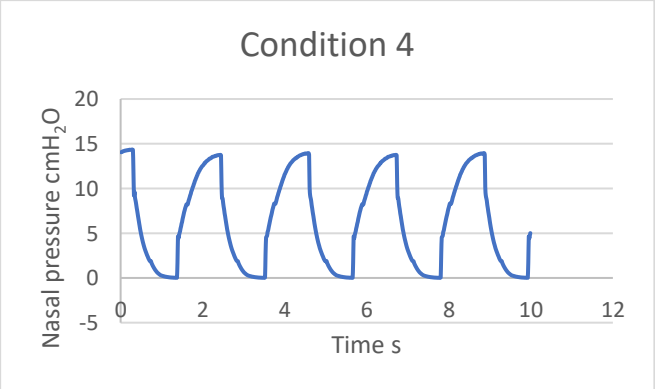


Fig 35. Pressure-time graph of the condition 4

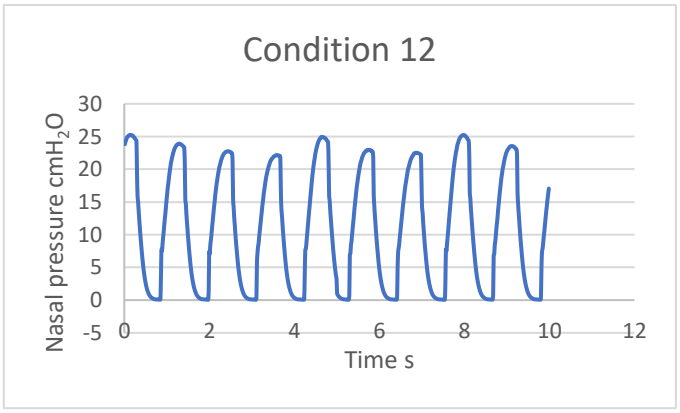


Fig 36. Pressure-time graph of the condition 12

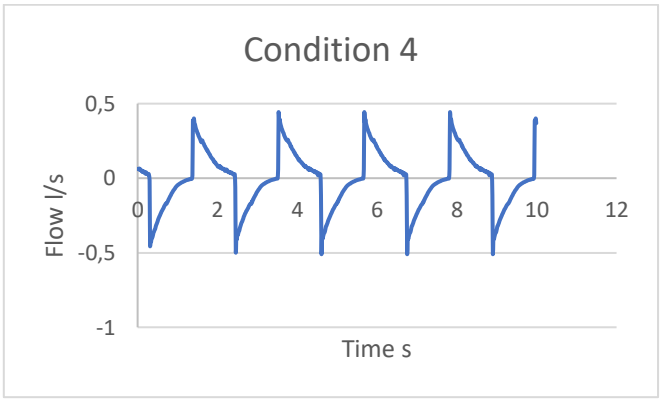


Fig 37. Flow-time graph of the condition 4

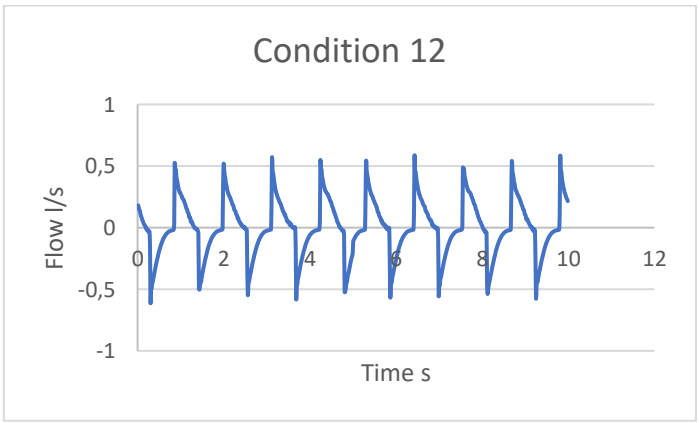


Fig 37. Flow-time graph of the condition 12

6. Execution Schedule

6.1. Work-breakdown Structure

Design and application of a low-cost, easy-to-build, non-invasive pressure support ventilator for pediatric use in low-resource countries.	
1. Preparation and advance planning	1.1. Bibliographic research
	1.2. Definition of objectives and scope of the project
	1.3. Preparation of the work plan
	1.4. Acquisition of materials
2. Hardware	2.1. PCB board design
	2.2. Arduino-Transducers Implementation
	2.3. Blower Implementation
	2.4. Display Implementation
	2.5. Functionality test
3. Software	3.1. Code creation
	3.2. Checking the increase in respiratory rate through the code
	3.3. Implementation with the hardware
	3.4. Functionality test
4. Adaptation of the respirator for children	4.1. Search for methods to adapt the ventilator
	4.2. Assessment of different valves
	4.3. Valve testing
5. Enclosure	5.1. Definition of the sizes and the program with which it will be designed
	5.2. Creation of the enclosure
	5.3. Enclosure impression
6. Final integration	6.1. Integration of all parts
	6.2 Modification of the code
	6.3. Functionality test of the ventilator
	6.4. Obtaining and analyzing test results
7. Completion of work	7.1. Report writing
	7.2. Preparation of the oral presentation

Tab 3. Work-Breakdown Structure of the project

6.2. WBS dictionary

In the WBS dictionary, the cost of each of the tasks is usually stated, but in this case, the price of each of these tasks can be found later in point 8, the economic pre-feasibility section.

Task: Preparation and advance planning
Description: Process in which the project is planned.
Deliveries: Task 1.1, Task 1.2, Task 1.3, Task 1.4.
Assigned resources: Computer.
Duration: 8 weeks
Key points: • Bibliographic research: Previous step to know the essential basis of the project. (2 weeks) • Definition of objectives and scope of the project. (2 weeks) • Preparation of the work plan: It refers to the establishment of all the tasks prior to carrying out the project. Includes the creation of the WBS, the PERT and the GANNT in order to carry out a better organization. (2 weeks) • Acquisition of materials: They must be ordered in order to start the assembly part of the project. (2 weeks)

Tab 4. Work-Breakdown Structure dictionary of Task 1

Task: Hardware
Description: It refers to the most mechanical part and where more use is made of the electronics of the project.
Deliveries: Task 2.1, Task 2.2, Task 2.3, Task 2.4, Task 2.5.
Assigned resources: Computer, laboratory, Arduino Nano board, high pressure blower, two pressure transducers, two differential flow transducers, PCB board, display and blower-controller.
Duration: 8 weeks
Key points: • PCB board design: The design of this component is important to make the connections of the other components. (3 weeks) • Arduino-Transducers Implementation: Connection of the Arduino Nano controller with the pressure and flow transducers. (1 week) • Blower Implementation: Arduino connection with the transducers and the high-pressure blower. (1 week) • Display Implementation: Implementation of the display with the other components already connected. (2 weeks) • Functionality tests: Verify that the connections are well made and that no component fails. (1 week)

Tab 5. Work-Breakdown Structure dictionary of Task 2

Task: Software
Description: It refers to the creation of the code to be able to configure the ventilator and for it to carry out the necessary commands.
Deliveries: Task 3.1, Task 3.2, Task 3.3, Task 3.4.
Assigned resources: Computer with the Arduino program, laboratory.
Duration: 6 weeks
Key points: • Creation of the code. (3 weeks) • Verification of the increase in the respiratory rate through the code: We must see if really changing the frequency of the blower it is possible or not to reach the level of respiration of a child. (1 week) • Hardware implementation: Upload the code to the Arduino board. (1 week) • Functionality tests: Tests to verify that the integration between the Hardware and the Software works correctly. (1 week)

Tab 6. Work-Breakdown Structure dictionary of Task 3

Task: Adaptation of the respirator for children
Description: Find the best method to adapt the respirator to children so that they can use it.
Deliverables: Task 4.1, Task 4.2, Task 4.3.
Assigned resources: Computer with the design program, laboratory, 3D printer, plastic for 3D printing.
Duration: 8 weeks
Key points: • Search for methods to adapt the ventilator. (4 weeks) • Assessment of different valves: Once decided that the best method to adapt the ventilator is through a valve, make an assessment of them. (2 weeks) • Valve testing: After the assessment, test the valves to choose the best one. (2 weeks)

Tab 7. Work-Breakdown Structure dictionary of Task 4

Task: Enclosure
Description: Creation of a structure to be able to place all its components in an orderly manner. It also has the function of protecting them, since it is a casing that covers them.
Deliverables: Task 5.1, Task 5.2, Task 5.3.
Assigned resources: Computer with the design program, laboratory, 3D printer, plastic for 3D printing.
Duration: 4 weeks
Key points: • Definition of the sizes and the program with which it will be designed. (1 week) • Creation of the enclosure: Design of the enclosure with the 3D program previously chosen. (2 weeks) • Printing of the enclosure. (1 week)

Tab 8. Work-Breakdown Structure dictionary of Task 5

Task: Final Integration
Description: It consists of attaching all the parts that we have separately and making the first version of the BiPAP respirator for children.
Deliverables: Task 6.1, Task 6.2, Task 6.3.
Assigned resources: Computer with Arduino and Labview, laboratory, 3D printer, plastic for 3D printing, Hardware, Software, valve or some other method.
Duration: 12 weeks
Key points: • Integration of all parts: Integrate all parts of the ventilator (Hardware, Software and Enclosure) to start using the device. (3 weeks) • Modification of the code: When introducing the valve as a new component, modifications must be made to the previously created code. (4 weeks) • Functionality test of the ventilator: Test to verify that the ventilator works correctly. (1 week) • Obtaining and analyzing test results: Collect the data from the tests carried out with the ventilator to analyze its operation and draw the corresponding conclusions. (4 weeks)

Tab 9. Work-Breakdown Structure dictionary of Task 6

Task: Completion of work
Description: Compilation of all the information collected during the work and completion of the report.
Deliveries: Task 7.1, Task 7.2 and final report.
Assigned Resources: Computer
Duration: 3 weeks
Key points: • Report writing: This task does not refer to the writing of the entire project report, since each part of it is included in the previous tasks, but to the last modifications before delivery. (1 week) • Preparation of the oral presentation. (2 weeks)

Tab 10. Work-Breakdown Structure dictionary of Task 7

6.3. PERT

To determine the time required to do the project, a PERT diagram has been done. As it can be seen below, some of the tasks overlap. This is because there is a part of the adult ventilator project that can be taken advantage of. The reason and which are the affected tasks is explained in more detail in point 6.4, with the GANTT diagram.

Task	Preceding task	Consequent task	Duration (weeks)
1.1	Start	1.2	2
1.2	1.1	1.3/1.4	2

1.3	1.2	2.1/3.1/5.1	2
1.4	1.2	2.1/3.1/5.1	2
2.1	1.3/1.4	2.2	3
2.2	2.1	2.3	1
2.3	2.2	2.4	1
2.4	2.3	2.5	2
2.5	2.4	3.3	1
3.1	1.3/1.4	3.2	3
3.2	3.1	3.3/4.1	1
3.3	2.5/3.2	3.4	1
3.4	3.3	6.1	1
4.1	3.2	4.2	4
4.2	4.1	4.3	2
4.3	4.2	5.2	2
5.1	1.3/1.4	5.2	1
5.2	4.3/5.1	5.3	2
5.3	5.2	6.1	1
6.1	3.4/5.3	6.2	3
6.2	6.1	6.3	4
6.3	6.2	6.4	1
6.4	6.3	7.1	4
7.1	6.4	7.2	1
7.2	7.1	End	2

Tab 11. PERT tasks

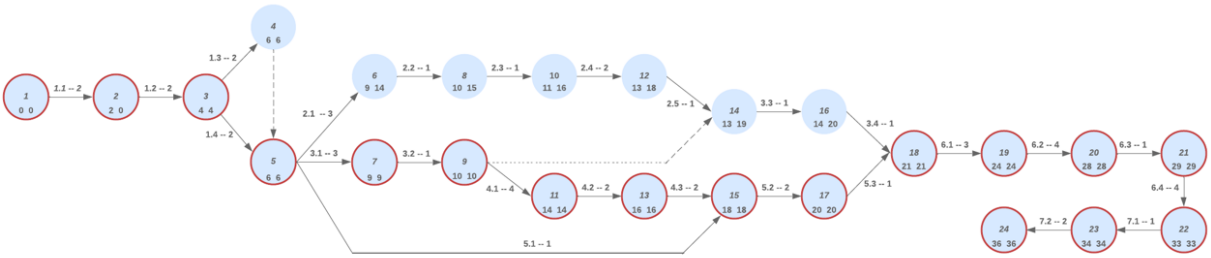


Fig 38. PERT diagram

The critical path is the one marked with the red circles. It is as follows: 1-2-3-5-7-9-11-13-15-17-18-19-20-21-22-23-24. To see it with higher resolution, go to Annex 8.

6.4. GANTT

Tom's planner program has been used to make the Gantt chart. It starts in May 2021 and ends in January 2023.

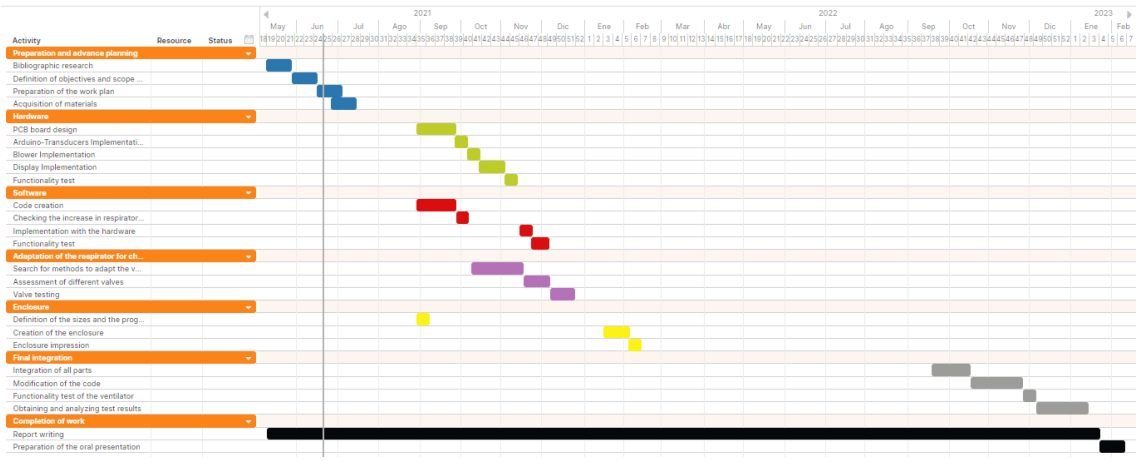


Fig 39. GANTT diagram

The activities that I have decided to overlap and therefore carry out more than one task on the same week have been the hardware tasks with the code creation and the first task of the enclosure part. This is because we can use the code and the enclosure of the ventilator for adults as a first version for the new ventilator. Even so, the Hardware-Software implementation task must be done after the hardware functionality tests. In this period in which no Software tasks are carried out, the task of finding a method to adapt the ventilator for children is completed. Once the method to be used has been decided, the enclosure creation tasks are done. It should also be noted that the task of writing the report is done throughout the project. To visualize the GANTT diagram better, click on the following link: https://plan.tomsplanner.es/public/tfgrespirador

7. Technical viability

Strengths	**Opportunities**
-There is a previous product created that is similar with which we can fix. -We have selected small components that fit in the desired measurements. -Selected components that meet the objectives of low cost and ease of use. -The acquisition of these components is easy, since they can be found on the Internet.	- Implementation in low-income countries, since the objectives are met. - Help combat the Covid-19 pandemic in countries where they have very few respirators.
Weaknesses	**Threats**
-At the moment, it is a controlled ventilator and still lacks the triggering. Even so, a future goal is to incorporate it. -The ventilator has not been tested on patients yet.	-It is very likely that the place where the respirator is used does not have a 3D printer, however, the support is not one of the parts that needs replacement, only in case it breaks. -The valve involves an extra cost.

Tab 12. Project DAFO

8. Economic viability

The next point refers to the study of costs and budget. Regarding the first task of *Preparation and advance planning of the project*, we can say that it would have a cost of 15€/h. The tasks of this project have been planned for weeks, but it could be said that the weekly hours invested in this project would be an average of 10 hours. As can be seen in the GANTT diagram in figure 11, the *Preparation of the work plan* task and the *Acquisition of materials* have an overlapping week, therefore, even if the total of the tasks add up to 8 weeks, economically they would only be considered 7. This is why the *Preparation and planning of the project* task would have a cost of 1.125€, 1.050€ referring to personnel costs and 75€ to the materials purchased. Being a student myself and working without getting paid, it would not be necessary to assume the cost of employee, but if anyone else on the team did, it would count as a cost.

Regarding the second task of *Hardware*, there would be no expense for the material, since the materials are acquired in the previous task, but there would be for the devoted time. If we maintain the price of 15€/h, this work would cost a total of 565€. This is because during the 8 weeks dedicated to *Hardware* tasks, time is also dedicated to other tasks, as it is indicated in figure 11. However, it happens the same as before, in the case that I was the one doing this task, these costs would not be considered. The third task, the *Software* one, would have a total cost of 415€ and the *Adaptation of the respirator for children* task, around 685€. Within this task, the price of the valve should also be added, which would be around 100€ [25]. Next, the price of the *Enclosure* task would be 400€, considering that the laboratory has a 3D printer as in this case. Its printing, due to the material used and the hours of operation of the printer, would have a total extra cost of between 10€ and 15€. All these calculations have been made with the allocation of 15€ per hour of work.

The cost of devoting the necessary time for the *Final integration* is 1.800€ for the time invested and 5€ for the material purchased, the PCB Board. In the case of the last task of *Writing the report* and *Preparing the presentation*, it would not represent any additional cost, since this is something that I would do myself and as I said before, as a student, I would not be financially rewarded for the time invested. In this way, the project would have a total cost of approximately 5.300€, but we have already seen that the price of the materials to build the respirator would be less than 200€, therefore we would continue to meet the low-cost objective. Below, in Table 13 the classification of costs can be looked more clearly.

	Personnel	**Material**	
Preparation and advance planning	1.050€	75€	
Hardware	565€	-	
Software	415€	-	
Adaptation of the respirator for children	750€	100€	
Enclosure	500€	15€	
Final integration	1.800€	5€	
Total	5.080€	**195€**	5.275€

Tab 13. Project costs

9. Regulations and legal aspects

At the international level, electrical medical equipment, mechanical ventilators among them, must meet the general safety requirements according to the IEC 60601-1 Standard and the particular standards of the IEC 60601 Series. The most technically appropriate standards for the national manufacture of these medical devices are the following [30]:

- ISO 13485 quality management system.
- International standard IEC 60601-1-1: 1996. Electromedical equipment - General safety requirements. This standard establishes the necessary safety requirements to provide protection to the patient, the operator and the environment.
- UNE-EN 60601-1: 2006/A1: 2013: Part 1: General requirements for basic safety and essential operation. (Ratified by AENOR in November 2013).
- UNE-EN 60601-1-2: 2015: General requirements for basic safety and essential operating characteristics. Collateral standard: Electromagnetic disturbances. Requirements and tests.
- UNE-EN 60601-1-11: 2015: General requirements for basic safety and essential operation. Collateral Standard: Requirements for medical electrical equipment and the medical electrical system used for care in the home medical environment.
- UNE-EN 62304: 2007/A1: 2016: Medical device software. Software life cycle processes.
- ISO 10993: Fifth edition 2018-08: Biological evaluation of medical devices - Part 1: Evaluation and testing within a risk management process.
- ISO 18.562-1 First edition 2017-03: Assessment of biocompatibility of respiratory gas pathways in healthcare applications - Part 1: Assessment and testing within a risk management process.
- ISO 18.562-2 First edition 2017-03: Evaluation of the biocompatibility of respiratory gas pathways in healthcare applications. - Part 2: tests for particulate matter emissions.
- ISO 18.562-3 First edition 2017: evaluation of the biocompatibility of respiratory gas pathways.
- ISO 18.562-3 First edition 2017: Evaluation of biocompatibility of respiratory gas pathways in healthcare applications - Part 3: Volatile organic compound emission tests.
- ISO 80601-2-12: 2011: Particular requirements for the basic safety and essential performance of critical care ventilators.

10. Conclusions and future improvements

With the results obtained, it can be affirmed that this project has fulfilled all its objectives. In the first place, it has been possible to adapt the previously created ventilator to be used in adults at high frequencies so that it can also be used in children from 1 to 14 years of age. We can consider that compared to the other respirators on the market it is a low-cost device, since materials do not exceed €200. We can also affirm that it is an easy device to build, since it does not have complex components to use. The only thing that could be complicated to make for someone without knowledge of the subject would be the PCB board, but even so, as it is an open-source project, the idea is to provide both the layout of the PCB board design and the Arduino code so that can be recreated in low-income countries.

With regards to potential future enhancements, one would be to integrate support ventilation capabilities. This would enable the ventilator to detect when the patient is attempting to inhale and initiate ventilation automatically when a predefined threshold pressure is exceeded (trigger). This feature was attempted in this project, however, due to the integration of the valve, it proved to be difficult to implement.

11. Bibliography

11.1. References

[1] Mandelzweig, K. et al. (2018) “Non-invasive ventilation in children and adults in low- and low-middle income countries: A systematic review and meta-analysis,” Journal of Critical Care, 47, pp. 310–319. Available at: https://doi.org/10.1016/j.jcrc.2018.01.007.

[2] Garmendia, O. et al. (2020) “Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions: Open Source Hardware Description, performance and feasibility testing,” European Respiratory Journal, 55(6), p. 2000846. Available at: https://doi.org/10.1183/13993003.00846-2020.

[3] Howie, S. (2008) “Beyond good intentions: Lessons on equipment donation from an african hospital,” Bulletin of the World Health Organization, 86(1), pp. 52–56. Available at: https://doi.org/10.2471/blt.07.042994.

[4] What are the different types of mechanical ventilation? (2020) MedicineNet. MedicineNet. Available at: https://www.medicinenet.com/different_types_of_mechanical_ventilation/article.htm

[5] Cheifetz, I.M. et al. (2006) “Respiratory support for the child with critical heart disease,” Critical Heart Disease in Infants and Children, pp. 307–332. Available at: https://doi.org/10.1016/b978-032301281-2.50014-x.

[6] UCSanDiegoHealth (2021) Ventilador BiPAP, UCSanDiegoHealth - BiPAP ventilator. Available at: https://myhealth.ucsd.edu/Spanish/RelatedItems/3,90237es

[7] El Hospital (2021) Ventiladores Para Cuidado Intensivo, El Hospital. Available at: https://www.elhospital.com/temas/Ventiladores-para-cuidado-intensivo+8062027?pagina=7

[8] Author Heather Glass et al. (2022) High-acuity ventilator cost guide, Medtronic. Available at: https://hcpresources.medtronic.com/blog/high-acuity-ventilator-cost-guide

[9] Respirem - respirem.org (2021) Respirem. Available at: https://respirem.org.cutestat.com/

[10] Transductores Mecánicos (2017) TRANSDUCTORES. Available at: https://tranductoresjado.wordpress.com/transductores-mecanicos/

[11] Transductores de presión mecánicos (2021) UDG - MX. Available at: https://biblioteca.udgvirtual.udg.mx/jspui/handle/123456789/3274?mode=full

[12] Jecrespom, P. and Jecrespom (2017) Placas Arduino, Aprendiendo Arduino. Available at: https://aprendiendoarduino.wordpress.com/2017/06/19/placas-arduino-2/

[13] Ugai, Y. et al. (1997) “Development of wide-viewing-angle TFT-lcds using halftone gray-scale method,” Electronics and Communications in Japan (Part II: Electronics), 80(5), pp. 89–98. Available at: https://doi.org/10.1002/(sici)1520-6432(199705)80:5<89::aid-ecjb12>3.0.co;2-2.

[14] Contributor, T.T. (2019) What is LCD (liquid crystal display)?, WhatIs.com. TechTarget. Available at: https://www.techtarget.com/whatis/definition/LCD-liquid-crystal-display

[15] Brandt, M.J. et al. (2017) “Valves and meters,” Twort's Water Supply, pp. 743–775. Available at: https://doi.org/10.1016/b978-0-08-100025-0.00018-1.

[16] Electronic pinch valves (no date) Clippard Electronic Pinch Valves. Available at: https://www.clippard.com/products/isolation-valves-npv

[17] Fang, Z. et al. (2020) “AmbuBox: A fast-deployable low-cost ventilator for covid-19 emergent care,” SLAS Technology, 25(6), pp. 573–584. Available at: https://doi.org/10.1177/2472630320953801.

[18] M., A. (2022) Softwares de modelado 3D gratis, 3Dnatives. Available at: https://www.3dnatives.com/es/softwares-modelado-3d-gratis-210720202/

[19] Garmendia, O. et al. (2020) "Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions: Open Source Hardware Description, performance and feasibility testing – Supplementary Technical Description," European Respiratory Journal, 55(6), p. 2000846. Available at: https://doi.org/10.1183/13993003.00846-2020.

[20] XGZP6847 Datasheet. Available at: https://www.sgbotic.com/products/datasheets/sensors/02976-datasheet.pdf

[21] XGZP6897A Datasheet. Available at: https://cfsensor.com/wp-content/uploads/2022/11/XGZP6897A-Pressure-Sensor-V2.5.pdf

[22] Arduino Mega 2560 REV3 (2022) Arduino Official Store. Available at: https://store.arduino.cc/products/arduino-mega-2560-rev3

[23] Block diagram 1. arduinomega2560 (2015). Available at: https://www.researchgate.net/figure/Block-Diagram-1-ArduinoMEGA2560-The-Arduino-Mega-2560-is-a-type-of-microcontroller_fig5_281538436

[24] 3.5inch Arduino display-mega2560 (no date) 3.5inch Arduino Display-Mega2560 - LCD wiki. Available at: http://www.lcdwiki.com/3.5inch_Arduino_Display-Mega2560

[25] ASCO (2022) Emerson. Available at: https://www.emerson.com/en-es/catalog/automation/fluid-control-pneumatics/angle-seat-pinch-diaphragm-valves/asco-s307-en-gb

[26] Valves (2022) Asco™ Series S307 pinch solenoid valves, İntegral Elektronik. Available at: https://integralelektronik.com/product/asco-series-s307-pinch-solenoid-valves/#tab-specification

[27] Kadu, C.B. and Patil, C.Y. (2016) "Design and implementation of stable PID controller for Interacting Level Control System," Procedia Computer Science, 79, pp. 737–746. Available at: https://doi.org/10.1016/j.procs.2016.03.097.

[28] Bhunia, S. and Tehranipoor, M. (2019) "Printed Circuit Board (PCB): Design and test," Hardware Security, pp. 81–105. Available at: https://doi.org/10.1016/b978-0-12-812477-2.00009-5.

[29] Zhang, H., Krooswyk, S. and Ou, J. (2015) "PCB design for Signal integrity," High Speed Digital Design, pp. 27–115. Available at: https://doi.org/10.1016/b978-0-12-418663-7.00002-2.

[30] Normativa y especificaciones técnicas Ventiladores Mecánicos (2021). Available at: https://www.uchile.cl/documentos/normativa-y-especificaciones-tecnicas-ventiladores-mecanicos_162480_1_2932.pdf

11.2. Figures

Fig 1. Iron lung - Pulmón de Acero (2022) Wikipedia. Wikimedia Foundation. Available at: https://es.wikipedia.org/wiki/Pulm%C3%B3n_de_acero

Fig 2. Chest cuirass - Negative pressure ventilation (2021) Virtual Museum. Available at: https://museum.aarc.org/galleries/negative-pressure-ventilation/

Fig 3. BiPAP respirator O. Garmendia et al. - Garmendia, O. et al. (2020) "Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions: Open Source Hardware Description, performance and feasibility testing – Supplementary Technical Description," European Respiratory Journal, 55(6), p. 2000846. Available at: https://doi.org/10.1183/13993003.00846-2020.

11.3. Tables

12. Annexes

Annex 1 – Technical parameters of driver 7040

Garmendia, O. et al. (2020) "Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions: Open Source Hardware Description, performance and feasibility testing – Supplementary Technical Description," European Respiratory Journal, 55(6), p. 2000846. Available at: https://doi.org/10.1183/13993003.00846-2020.

+5V	Voltage supply to Hall		+5V	5V voltage supply
HU	Hall U		VSR	0-5 voltage or PWM to adjust speed
HV	Hall V		GND	GND
HW	Hall W		EN	Running Or Stop
GND	GND		FR	CCW or CW
U	Phase U		FG	Speed Pulse Output
V	Phase V		GND	24V Power -
W	Phase W		+	24VPower +

Product model	Unit	Driver Of 7040-24V
Rated voltage	V	24
Allowable operating voltage	V	7-26
Maximum continuous current	A	3
Holzer electrical angle	Degree	60
Weight	Gram	18

Annex 2 – Technical parameters of XGZP6847 pressure transducer

XGZP6847 Datasheet. Available at: https://www.sgbotic.com/products/datasheets/sensors/02976-datasheet.pdf

Item	Data	Unit
Output Signal	0.5-4.5	V
Accuracy	±1.0	%Span
TSO(Temp. Coefficient of Offset)	±0.03	%FS/℃
TCS(Temp. Coefficient of Span)	±0.03	%FS/℃
Long Term Stability(1year)	±2	%Span
Over Pressure	2X (≤500kPa)	Rated
	1.5X(≥500kPa)	
Compensation Temp.	0 ～ 85/32 ～ 176	℃/℉
Ambient Temp.	-20 ～ 100/-4 ～ 212	℃/℉
Storage Temp.	-40 ～ 125/-40 ～ 257	℃/℉

Annex 3 – Technical parameters of XGZP6897A pressure transducer

XGZP6897A Datasheet. Available at: https://cfsensor.com/wp-content/uploads/2022/11/XGZP6897A-Pressure-Sensor-V2.5.pdf

Item		Data	Unit	Remark
Available Pressure Range[1]		-100···-0.5 ~ 0 ~ 0.5···200	kPa	Customization acceptable
Power Supply[2]		5.0/3.3	Vdc	
Max. Excitation Current		3	mA	
Output Range[3]		0.5 ~ 4.5/0.2 ~ 2.7	Vdc	Customization acceptable
Total Accuracy[4]	10kPa < Pressure ≤200kPa	±2	%Span	Customization acceptable
	Pressure ≤10kPa or > 200kPa	±2.5		
Long Term Stability[5]		±0.5	%Span	
Over Pressure[6]	Pressure ≤5kPa	5X	Rated Pressure	Base on P1 > P2 and differ on specific pressure range
	5kPa < Pressure ≤200kPa	2.5X	Rated Pressure	
Burst Pressure[7]	Pressure ≤5kPa	10X	Rated Pressure	
	5kPa < Pressure ≤200kPa	3X	Rated Pressure	
Max. Pressure on P2 Port		250	kPa	
Compensation Temp.[8]		0 ~ 60/32 ~ 140	℃/℉	Customization acceptable
Operating Temp.[9]		-20 ~ 100/-4 ~ 212	℃/℉	
Storage Temp.		-30 ~ 125/-22 ~ 257	℃/℉	
Response Time[10]		2.5	mS	

Annex 4 – Technical aspects of ASCO™ Series S307 Pinch solenoid valve

ASCO™ pinch solenoid valve series - fluidconcept.de (2022). Available at: https://www.fluidconcept.de/media/downloads/dbl/ASCO-S307.pdf

Materials	
Body	Anodized aluminum
Pinching device	POM (reinforced acetal copolymer)
Internal components	Stainless steel
Core tube	Stainless steel

Installation

- Solenoid valve can be mounted in any position.

Coil		
Continuous Duty		ED 100%
Encapsulation material		PET (polybutylene terephftalate) fiberglass reinforced
Insulation class		F (140°C)
Ambient temperature		-10°C +60°C
Electric connections		DIN 46340 - 3 poles plug-connectors (EN175301-803)
Protection degree		IP 65 (DIN 40050) with plug-connectors
Voltages	DC	12-24V (+10% -5%)
		(Other voltages on request)

TUBINGS		Pinching strength (kg)	Series and type		Power absorption (W)	Notes	Weight (kg)
I.D. (mm)	O.D. (mm)		Valve	Coil			
4,8	7,9	0,850	**S307-05**	**Z130A**	13	-	0,420
6,4	9,5	1,100	**S307-06**				

Annex 5 – Arduino code for ventilator operation

The last part of the code is for the future improvement of support mode.

```
#define CENTRE 240

#define BLUE    0x001F //parameters of the colors for the display

#define RED     0xF800

#define GREEN   0x07E0

#define CYAN    0x07FF

#define MAGENTA 0xF81F

#define YELLOW  0xFFE0

#define WHITE   0xFFFF

#define BLACK    0x0000


#include <TFT_HX8357.h> // Hardware-specific library LCD

TFT_HX8357 tft = TFT_HX8357();

#include <PID_v1.h>    //PID library


#define analog  11                //define digital 11 as analog output PWM.

#define analog_flow  9           //define digital 9 as analog_flow.

#define switchPin  12            //define digital 12 as SwitchPin.

#define mode  13           //define digital 13 modo automatico/paciente.

#define led 8                //define digital 8 as led.

#define valve 10              //define digital 10 control valvula ins/esp

#define sensorpress1 A6             //define sensorpress1 as analog input A6 (pressure
sensor 1)

#define sensorpress2 A7           //define sensorpress1 as analog input A7 (pressure
sensor 2)

#define analog_ipap A2         //define analog_ipap as analog ipunt A2 (ipap
potentiometer)

#define analog_porcentaje A3   //define analog_porcentaje as analog input A3 (cycling
potentiometer)

#define analog_freq A4           //frequency potentiometer adjustment is connected to
the analog input 4

#define analog_dutty A5         //I/E potentiometer adjustment is connected to the
analog input 5

#define flow_ins A12            //define sensorpress Flow ins as analog input A12.

#define flow_esp A13            //define sensorpress Flow esp as analog input A13.
```

```
int freq, periodo, dutty, duttycien;

double Setpoint, Input, Output,flow_hp,flow_bp;   //control variables

double Kp=0.12, Ki=0.75, Kd=0;    // PID values variables
KP(proportional),Ki(integral),Kd(derivative)

byte autoipap = 0,lcdmodecpap=1,lcdmodebipap=1,switchState =
0,selectmode=0,mode1=0,restard,ctrolvalve=0;  //state switch variables

unsigned long t0,tiempoahora ,t1,t2;

int Tm =10000;    //Sampling freq in us

int Tx=20900;//time x pixel

int  bpm_1=-1, dutty100_1=-1,timediv_1=-1,P_ipap_1=-1,volumenesp=0,volumenins=0;

float tiempo =0;

float fc_hp=0.125;              //cuttof freq for HP filter

float fc_lp=1.125;              //cuttof freq for LP filter

float RC_hp =1/(2*PI*fc_hp);

float RC_lp =1/(2*PI*fc_lp);

double data_filt_hp[] = {0, 0};

double data_hp[] = {0, 0};

double data_filt_lp[] = {0, 0};

double data_lp[] = {0, 0};

double maximo=0,vmax=0,flow=0;

int ipap,zero1,zero2,zeroins,zeroesp,consigna=0;

float
porcentaje,P_sensor1,P_sensor1_g,P_ipap,P_sensor2,flowins,flowesp,flowins_g,flowesp_g;

int ctroltime;

int gain_flow =100;   // variable integer gain_flow

char P_sensor1string[4];

char P_sensor2string[4];

char P_IPAPstring[4];

char porcentajestring[2];

char timestring[2];

int i,x=0,yp,yesp,yins,h=0,y=0,z=0,g=0;

int samplepressure1[25];   //array for sampling pressure sensor 1 (samplepressure0)

int samplepressure2[25];  //array for sampling pressure sensor 2 (samplepressure1)

int sampleflowins[25];

int sampleflowesp[25];

int samplepressure1_g[25];

int sampleflowins_g[25];

int sampleflowesp_g[25];
```

```
double totalpress2 = 0, totalpress1 = 0,
totalpress1_g=0,totalflowins=0,totalflowesp=0,totalflowins_g=0,totalflowesp_g=0;

double
pressinzero1,pressinzero1_g,pressinzero2,samplezero1,samplezero2,totalzero1=0,totalzero2
=0,flowsinzeroins,flowsinzeroesp,samplezeroins,samplezeroesp,totalzeroins,totalzeroesp;

float flowesp_ls,flowins_ls;

uint16_t bufpres[480];

uint16_t bufflow[480];

char vesp[4],vins[4],Pset[4],Preal[2];

unsigned long color;

float K1esp=0.233,K2esp=2.33,K1ins=0.233,K2ins=2.33;

float calflowcmH2o =2.55; //pressure sensors to measure flow (2,55Cmh20=1V)

float calprescmH2o =12.7475;//pressure sensors to mesure pressure (12.7475Cmh20=1V)


PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT); //PID function declaration



void setup() {


Serial.begin(115200);


pinMode(switchPin,INPUT);       //define switchPin as input

pinMode(mode,INPUT);     //mode automatic =0 mode paciente =1

pinMode(valve,OUTPUT);

pinMode(led,OUTPUT);            //define led as output

tft.init();

tft.setRotation(1);

//tft.invertDisplay(1);

tft.fillScreen(BLACK);


myPID.SetMode(AUTOMATIC);     //PID in automatic mode



consigna=ipap;


//---------------------------------------- calibration process ------------------------
-------------------------------------------------
```

```
for(i=0; i< 24; i++)          //array initialization
   {
samplepressure1[i] = 0;
samplepressure2[i] = 0;
sampleflowins[i]=0;
sampleflowesp[i]=0;
   }
i=0;




tft.setTextColor(WHITE,BLACK);
tft.drawString("Calibrating zero",150, 80,4);



for(i=0; i< 91; i++)   //record every 50ms pressure value for a total time of 4.5s and
accumulates the sum in totalzero1,totalzero2,totalzeroins,totalzeroesp variable


   {


if (i==0 || i==6 || i==12 || i==18 || i==24 || i==30 || i==36 || i==42 || i==48 || i==54
|| i==60 ||  i==66 || i==72 || i==78 || i==84 || i==90)


      {
         tft.drawString("*",50+4*i,130,4);
      }


samplezero1 = analogRead(sensorpress1);   //Readings of pressure sensor 1
totalzero1 = totalzero1 + samplezero1;
samplezero2 = analogRead(sensorpress2);   //Readings of pressure sensor 2
totalzero2 = totalzero2 + samplezero2;
samplezeroins=analogRead(flow_ins);        //Readings of pressure sensor flowins
totalzeroins = totalzeroins + samplezeroins;
samplezeroesp=analogRead(flow_esp);         //Readings of pressure sensor flowesp
totalzeroesp = totalzeroesp + samplezeroesp;


delay(50);
```

```
   }

//------------------------real zero calculated (P1,P2,flow ins,Flow esp) ---------------
-----------------------------------


zero1=totalzero1/i;

zero2=totalzero2/i;

zeroins=totalzeroins/i;

zeroesp=totalzeroesp/i;

//-------------------------------- write text in the display----------------------------
------------------

tft.drawString("P1:",100,180,4);

tft.drawString("P2:",285,180,4);

tft.drawString("flow Insp:",40,220,4);

tft.drawString("flow esp:",270,220,4);


//---------------  verification that zero pressure correction is OK (<0.3cmH20). If not,
gives an error and restarts  ----------------------


for(i=0; i< 50; i++) {


// zero desviations correction

pressinzero1 =analogRead(sensorpress1)-zero1;

pressinzero2 =analogRead(sensorpress2)-zero2;

flowsinzeroins=analogRead(flow_ins)-zeroins;

flowsinzeroesp=analogRead(flow_esp)-zeroesp;


//conversion of pressure transducer to cmH2o


P_sensor1=abs(pressinzero1*4.8*calprescmH2o/1023);

P_sensor2=abs(pressinzero2*4.8*calprescmH2o/1023);

flowins=abs(flowsinzeroins*4.8*calflowcmH2o/1023);

flowesp=abs(flowsinzeroesp*4.8*calflowcmH2o/1023);


//write tha values of pressure transducer to cmH2o in the display

tft.drawFloat(P_sensor1,2,175, 180,4);

tft.drawFloat(P_sensor2,2,360, 180,4);
```

```
tft.drawFloat(flowins,2,175, 220,4);

tft.drawFloat(flowesp,2,395, 220,4);


delay(300);



  if (P_sensor1 >0.3 || P_sensor2>0.3 || flowins>0.3 || flowesp>0.3)     {    //
calibration error


        tft.drawString("err calibrating ",150, 80,4);

        tft.setTextColor(WHITE,RED);


              if (P_sensor1 >0.3)   {


                      tft.drawFloat(P_sensor1,2,175, 180,4);    // if the pressure error
is in Psensor1 write the value in red on the LCD


                                    }


              if (P_sensor2 >0.3)   {


                      tft.drawFloat(P_sensor2,2,360, 180,4);   // if the pressure error
is in Psensor2 write the value in red on the LCD


                                    }


              if (flowins >0.3){


                      tft.drawFloat(flowins,2,175, 220,4);   // if the pressure error is
in flowins write the value in red on the LCD


                                  }


              if (flowesp >0.3){


                       tft.drawFloat(flowesp,2,395, 220,4);    // if the pressure error
is in flowesp write the value in red on the LCD
```

```
                              }

tft.setTextColor(WHITE,BLACK);
tft.drawString("Restard sofware.",150, 80,4);
delay(3000);
restard =1;         //restart
i=50;
                                                                    }


                    }

//------------------------------------------------------- if error,restart variables
and restart setup calibration routine-----------------
if (restard ==1)      {

            i=0;
            totalzero1=0;
            totalzero2=0;
            totalzeroins=0;
            totalzeroesp=0;
            restard=0;
            setup();

                      }


inicio();

//t0 ,t1 and t2 control the time intervals
t0=micros();
t1=millis();
t2=micros();
```

```
tiempoahora =millis();



                    } //end setup


void grafica1(int pixel){   // loop for creating the graph


 color=BLACK;


   if( bufpres[x]==300 || bufpres[x]==50 ||bufpres[x]==175 ||x==0 ||x==478 )
{color=RED;}


    if( bufpres[x]==300 || bufpres[x]==50 ||bufpres[x]==175 ||x==0 ||x==478 )
{color=RED;}


if(x==48 || x==96 || x==144 || x==192 || x==240 || x==288 || x==336 || x==384 || x==432
|| x==480)


      {


              if ((bufpres[x]>=170 && bufpres[x]<=180) || (bufpres[x]>=290 &&
bufpres[x]<=300)) { color=RED;}


      }


if(bufpres[x]==50 || bufpres[x]==75 || bufpres[x]==100 || bufpres[x]==125 ||
bufpres[x]==5175 || bufpres[x]==200 || bufpres[x]==225 || bufpres[x]==250 ||
bufpres[x]==275 || bufpres[x]==300)


      {


            if ((x>=469 && x<=479) || (x>=0 && x<=10)) { color=RED;}


      }



if ( x <=478) {


  tft.drawPixel(x,bufpres[x],color);
```

```
  tft.drawPixel(x,pixel,WHITE);
  bufpres[x]=pixel;
  x++;
              }
else{x=0;}



   color=BLACK;


   if( bufflow[x]==300 || bufflow[x]==50 ||bufflow[x]==175 ||x==0 ||x==478 )
{color=RED;}


    if( bufflow[x]==300 || bufflow[x]==50 ||bufflow[x]==175 ||x==0 ||x==478 )
{color=RED;}


if(x==48 || x==96 || x==144 || x==192 || x==240 || x==288 || x==336 || x==384 || x==432
|| x==480)


        {


              if ((bufflow[x]>=170 && bufflow[x]<=180) || (bufflow[x]>=290 &&
bufflow[x]<=300)) { color=RED;}


        }


if(bufflow[x]==50 || bufflow[x]==75 || bufflow[x]==100 || bufflow[x]==125 ||
bufflow[x]==5175 || bufflow[x]==200 || bufflow[x]==225 || bufflow[x]==250 ||
bufflow[x]==275 || bufflow[x]==300)


        {


            if ((x>=469 && x<=479) || (x>=0 && x<=10)) { color=RED;}


        }



  }


 void inicio(){
```

```
    tft.fillScreen(BLACK);
    tft.drawLine(0,50,0,300,RED); //vertical axis

                    for (int i=50; i <=300; i=i+25){
                              tft.drawLine(0,i,10,i,RED);  //ticks y axis
                                                  }

    tft.drawLine(479,50,479,300,RED); //vertical y2 axis

                    for (int i=50; i <=300; i=i+25){tft.drawLine(469,i,479,i,RED);}
//ticks y2 axis

    tft.drawLine(0,50,479,50,RED);
    tft.drawLine(0,300,479,300,RED); //horizontal axis

                   for (int x=0; x <=479; x=x+48){tft.drawLine(x,290,x,300,RED); }
//ticks x axis

    tft.drawLine(0,175,479,175,RED); //horizontal x2 axis

        for (int x=0; x <=479; x=x+48){tft.drawLine(x,170,x,180,RED);}  //ticks x2 axis

    tft.drawLine(345,313,360,313,YELLOW) ;
    tft.drawLine(95,313,110,313,WHITE) ;
    tft.setTextColor(YELLOW);
    tft.drawString("Flow",370,310,1);
    tft.drawString("-1 L/s",445,310,1);
    tft.drawString("1 L/s",450,40,1);
    tft.setTextColor(WHITE);
    tft.drawString("Press",120,310,1);
    tft.drawString("0CmH2o",1,310,1);
    tft.drawString("40CmH2o",1,40,1);
    tft.drawString("ms/div",245,310,1);
```

```
              }



void loop() {


freq = analogRead(analog_freq); //respiratory frequency

periodo=map(freq,0,1023,5000,1000);

dutty= analogRead(analog_dutty); // Inspiration/Expiration relation


float duttycien=map(dutty,0,1023,20,50);

float duttyuno=(duttycien/100);

int bpm =map(freq,0,1023,12,60);

//String duttycien_string;

int dutty100;

dutty100=duttycien;


int valor=digitalRead(valve); //0=exp and 1=ins


sampleflowins_g[g]=analogRead(flow_ins);

totalflowins_g=totalflowins_g+sampleflowins_g[g];


sampleflowesp_g[g]=analogRead(flow_esp);

totalflowesp_g=totalflowesp_g+sampleflowesp_g[g];


samplepressure1_g[g] = analogRead(sensorpress1);   //Pressure sensor 1 measurements are
accumulated

totalpress1_g =  samplepressure1_g[g] + totalpress1_g ;




g++;


Tx=-347.21*bpm+25000; // interval in microseconds used to plot pressure and flow pixels


int timediv =(Tx/1000)*48; //indicates the duration of each division of the TFT
```

```
if ( timediv_1 != timediv)      {    //only plot the value of timediv if it changes
value


          tft.drawNumber(timediv,225,310,1); // timediv graph in the TF

          timediv_1=timediv;      //this reassignment of the variable is what makes it
only enter once in the

                                  //if and only enter again if there is a change in the
timediv variable


                                 }


// ------------------------------ data graph every 20900 microsec (tx) -----------------
-----------------


if (micros()-(t2)>=Tx)  {


      tiempo=(micros()-t2);

      t2=micros(); //restart the time variable

      P_sensor1_g=100*abs((((totalpress1_g/g)-zero2)*4.8*calprescmH2o/1023));
//pressure in cmH20

      yp=map(P_sensor1_g,0,4000,300,55);




grafica1(yp);



switchState = digitalRead(switchPin);             //setup values switch

int ndec =P_ipap*10;

int  n =P_ipap;

int nint=n*10;

int dec =ndec-nint;

sprintf(Pset, "%02d.%01d", n, dec);









if ( switchState == LOW) {                              // Auto mode
```

```
            if (lcdmodebipap ==1)  {       //write text BPM I/E P Vi Ve one time only


                           tft.setTextColor(WHITE,BLACK);

                           tft.drawString("BPM:        I/E:       %  P:             Vi:
Ve:     ",0,10,4);

                           lcdmodebipap=0;


                                  }




            if (valor==0) {        //if valve is in exp mode


                    if (volumenins >-1) {


                             sprintf(vins, "%04d", volumenins);
// convert inspiratory volume to text with 4 digits

                             tft.drawString(vins,321,10,4);
// shows the inspiratory volume in ml on the screen

                             volumenins=-2;


                                       }




             flowesp_g=(((totalflowesp_g/g)-zeroesp)*4.8*calflowcmH2o/1023);
//exp flow sensor in cmH20

             flowesp_ls=100*(-K1esp+sqrt(K1esp*K1esp+(4*K2esp*flowesp_g)))/(2*K2esp);
//exp flow sensor in l/s applying k1 k2

             volumenesp =volumenesp+(flowesp_ls*tiempo/100000);
//calculate the exp volume

             yesp=map(flowesp_ls,0,100,175,300);
//mapping exp flow to graph values

             tft.drawPixel(x,bufflow[x],color);
//remove previous yesp saved in bufflow from the graph

             tft.drawPixel(x,yesp,YELLOW);
//plot current yesp

             bufflow[x]=yesp;
//saves the yesp value of the current pixel in bufflow so that it can be deleted from
the graph later before plotting the new one
```

```
                        }

          if (valor==1)  {               // if valve is in ins mode

                    if (volumenesp >-1) {

                        sprintf(vesp, "%04d", volumenesp);                  //
convert expiratory volume to 4-digit text
                        tft.drawString(vesp,420,10,4);                      // shows
the expirtory volume in ml on the screen
                        volumenesp=-2;
                                    }

          flowins_g=(((totalflowins_g/g)-zeroins)*4.8*calflowcmH2o/1023);
//ins flow sensor in cmH20
          flowins_ls=100*(-K1ins+sqrt((K1ins*K1ins)+(4*K2ins*flowins_g)))/(2*K2ins);
//ins flow sensor in l/s applying k1 k2
          volumenins =volumenins+(flowins_ls*tiempo/100000);
//calculate the ins volume
          yins=map(flowins_ls,0,100,175,50);
//mapping ins flow to graph values
          tft.drawPixel(x,bufflow[x],color);
//remove previous yins saved in bufflow from the graph
          tft.drawPixel(x,yins,YELLOW);
//plot current yins
          bufflow[x]=yins;
//saves the yins value of the current pixel in bufflow so that it can be deleted from
the graph later before plotting the new one

                       }

        dtostrf(porcentaje,2,0,porcentajestring);


        if ( bpm_1 != bpm)  {               //writes the selected BPM value to the
screen only if there are changes in its value

                bpm_1=bpm;
                tft.drawNumber(bpm,65,10,4);

                        }
```

```
        if ( dutty100_1 != dutty100) {      //writes the selected dutty100 value (I/E)
to the screen only if there are changes in its value


                dutty100_1=dutty100;

                tft.drawNumber(dutty100,145,10,4);


                                    }


        if ( P_ipap_1 != ipap)  {           //writes the selected pressure (ipap) value
to the screen only if there are changes in its value


                P_ipap_1=ipap;

                tft.drawString(Pset,235,10,4);


                                }




                            }



  if (switchState == HIGH)  {                       //  setup mode


          if (lcdmodebipap ==0)  {

                    tft.drawString("set Press:              CmH2o   Pres:         CmH2o
",0,10,4);

                    dutty100_1=-1;

                    bpm_1=-1;

                    P_ipap_1=-1;


                                    }


    lcdmodebipap=1;

    digitalWrite(valve , HIGH);

     dtostrf(P_ipap,2,1, P_IPAPstring);


          if ( P_ipap_1 != ipap)    {
```

```
                    P_ipap_1=ipap;
                    tft.drawString(Pset,115,10,4);

                                }
    int Psensor1 =P_sensor1;
    sprintf(Preal,"%02d",Psensor1);
    tft.drawString(Preal,345,10,4);


                            }

//-----------------------------------------reset graph variables-----------------------
-----
g=0;
totalpress1_g=0;
totalflowins_g=0;
totalflowesp_g=0;
                    }

//-------------------------------------------------end graphing data-------------------
-----------------


sampleflowins[i]=analogRead(flow_ins);          //Pressure sensor for ins flow
measurements are accumulated
totalflowins=totalflowins+sampleflowins[i];


sampleflowesp[i]=analogRead(flow_esp);          //Pressure sensor for esp flow
measurements are accumulated
totalflowesp=totalflowesp+sampleflowesp[i];


samplepressure1[i] = analogRead(sensorpress1);   //Pressure sensor 1 measurements are
accumulated
totalpress1 = totalpress1 + samplepressure1[i];


samplepressure2[i] = analogRead(sensorpress2);   //Pressure sensor 2 measurements are
accumulated
totalpress2 = totalpress2 + samplepressure2[i];


ctroltime = 3;  //control time variable can be adjusted with the potentiometer (3s-7s)
```

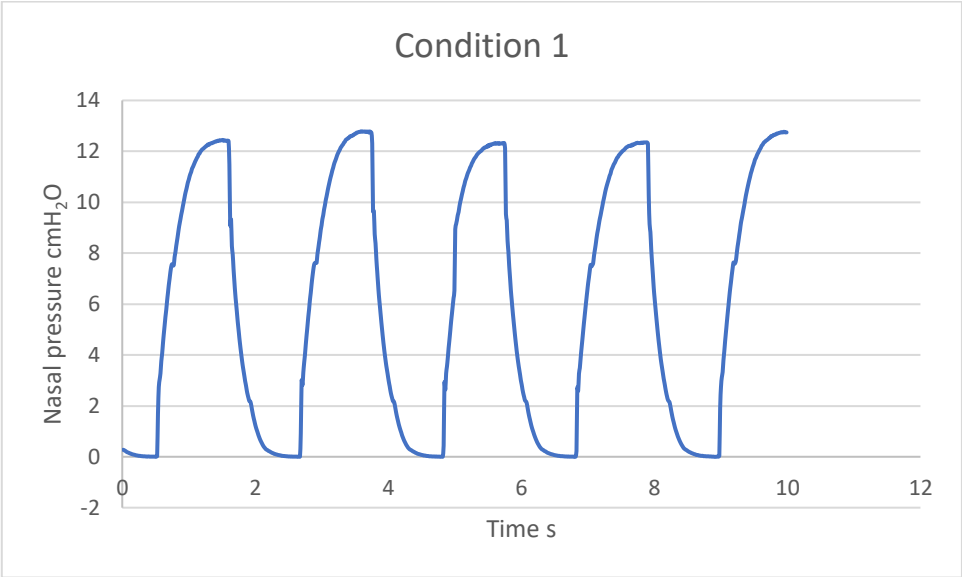
```
i++;


//---------------------------- control pid-----------------------------------------------
---


if (micros()-(t0)>=Tm)  {


          tiempo=(micros()-t0);

          t0=micros();

          pressinzero1 = (totalpress1/i)-zero1;               // pressure sensor 1
should be unplugged (connected to atmospheric pressure)and correct the displayed value
(zero desviations correction)

          pressinzero2 = (totalpress2/i)-zero2;               // pressure sensor 2
should be unplugged (connected to atmospheric pressure)and correct the displayed value
(zero desviations correction)

          P_sensor1=abs(pressinzero1*4.8*calprescmH2o/1023);

          P_sensor2=abs(pressinzero2*4.8*calprescmH2o/1023);


          ipap=map(analogRead(analog_ipap),0,1023,67,402);//ipap range: 4-24 cmH2O (67-
402)

          //ipap= 0.833*analogRead(analog_ipap)+170;

          P_ipap =(ipap*calprescmH2o*4.8/1023.00);       //ipap pressure in cmH2O.

          porcentaje=(100*(analogRead(analog_porcentaje)/1023.00))-1;     //cycling
potentiometer: values 0-100%


          if(porcentaje<0){porcentaje=0;}


          if (flow>0){ flow=100*sqrt(flow);}

              else{flow=-100*sqrt(-flow);}  //flow- linearization


          int flow_out = map(flow,-20000,40000,0,255);    // flow_out is adjusted
between 0-255 for doing the PWM (DIO9)

          analogWrite(analog_flow,flow_out);

          consigna=ipap;

          ctrolvalve=digitalRead(valve);




          if (ctrolvalve ==LOW)              {Input =pressinzero2; }

              else{Input =pressinzero1;}
```

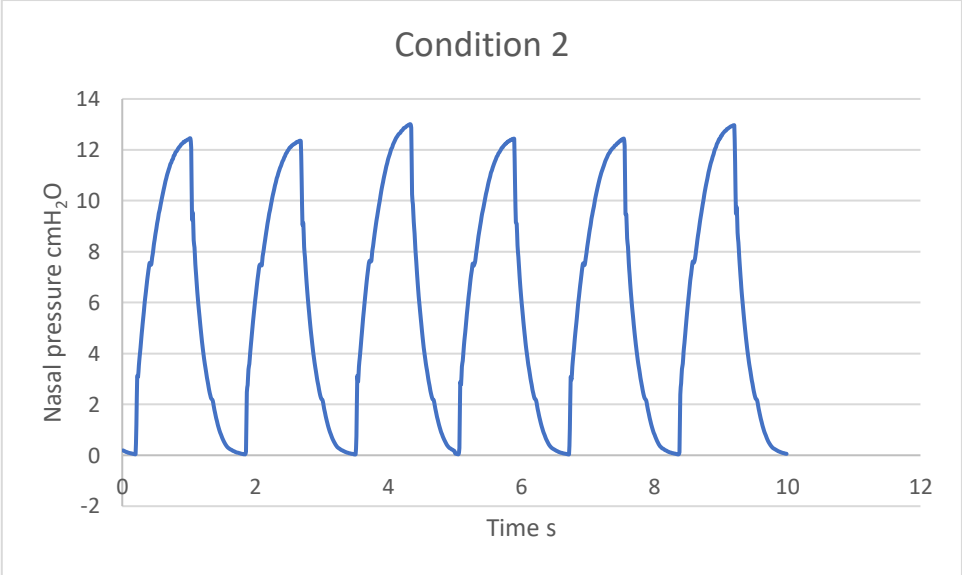
```
          Setpoint= consigna;       //setpoint for PID is ipap or epap depending on the
mode
          myPID.Compute();

          analogWrite(analog,Output);                          //DIA11 actuates the
blower

          if (autoipap==0) {dtostrf(ctroltime,1,0,timestring);}

          totalpress1=0;
          totalpress2=0;
          i=0;

                     }

//-------------------------------------------------end control pid-------------------------
-------------------------------------------------

selectmode=digitalRead(mode);

if (selectmode ==0)   {

        if ( mode1 != selectmode)  {

                  tft.drawString("Auto       ",380, 10,4);  //Control mode
                  mode1=selectmode;
                                   }

        if (millis()-(t1)>=(periodo))  {

              digitalWrite(valve , LOW);
              float tiempon=(millis()-t1);
              t1=millis();
                                     }
        if (millis()-(t1)>=(periodo*(1-duttyuno)))  {
```

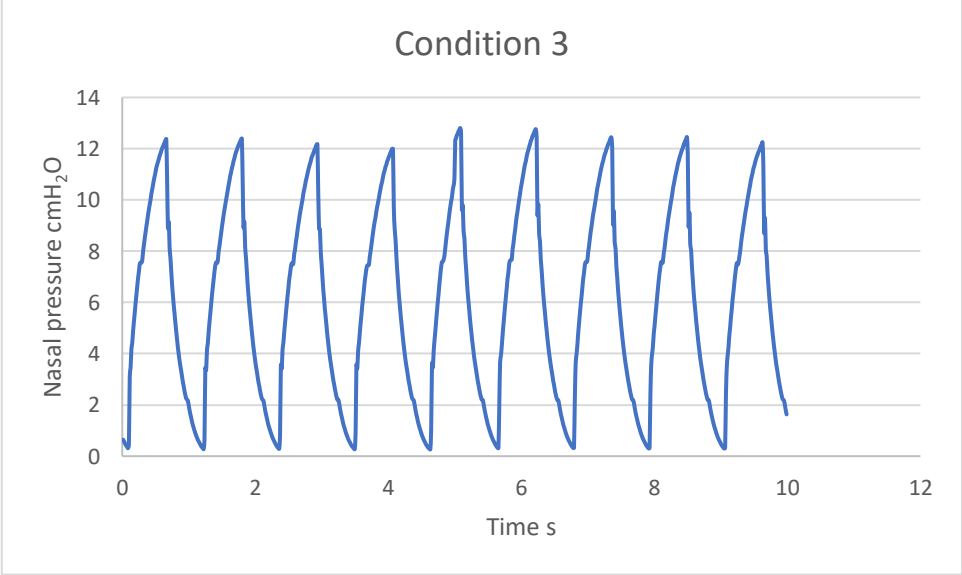
```
                digitalWrite(valve , HIGH);
                float tiempoff=(millis()-t1);

                                                    }


                            }


if (selectmode==1)  {    // Support mode (with trigger)

          if ( mode1 != selectmode)  {

                  tft.drawString("Patient   ",360,10,4);
                   mode1=selectmode;
                                      }


flow_hp = mifiltroHP(flow,tiempo/1000000,RC_hp);                          // HP filtering
flow signal
flow_bp = mifiltroLP(flow_hp,tiempo/1000000,RC_lp);                      // LP filtering
flow signal


                        }


            }//closed loop
```

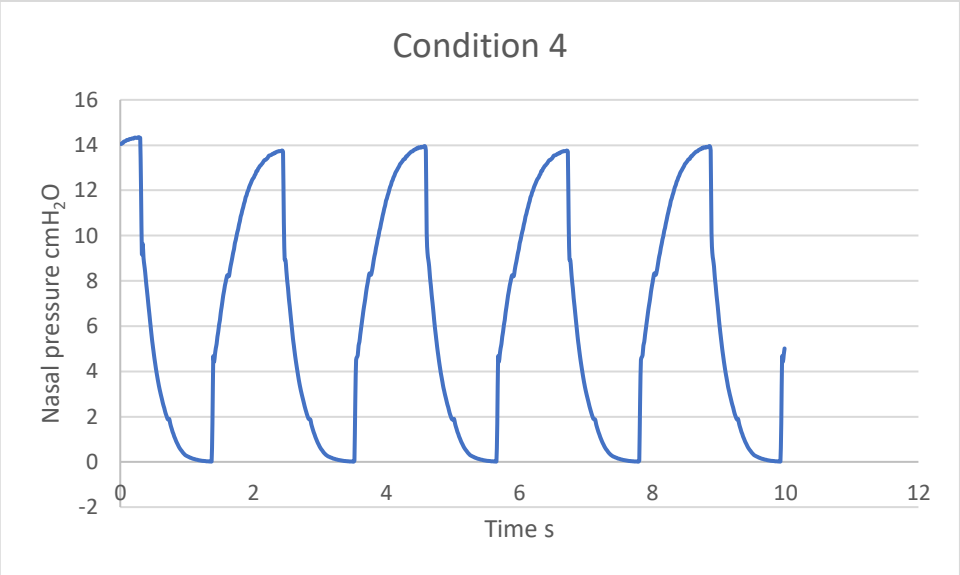
```
double mifiltroHP(double dato,double dt,double RC) {       // HP filtering function

double alpha = RC/(RC+dt);

data_hp[1] = dato;

    // High Pass Filter
    data_filt_hp[1] = alpha * (data_filt_hp[0] + data_hp[1] - data_hp[0]);

    // Store the previous data in correct index
    data_hp[0] = data_hp[1];
    data_filt_hp[0] = data_filt_hp[1];

  return (data_filt_hp[1]);

                                                    }

double mifiltroLP(double dato,double dt,double RC) {      // LP filtering function

double alpha = dt/(RC+dt);

data_lp[1] = dato;

    // low Pass Filter
    data_filt_lp[1] = alpha *data_lp[1]+(data_filt_lp[0]*(1-alpha));

    // Store the previous data in correct index
    data_lp[0] = data_lp[1];
    data_filt_lp[0] = data_filt_lp[1];

  return (data_filt_lp[1]);

                                                    }
```

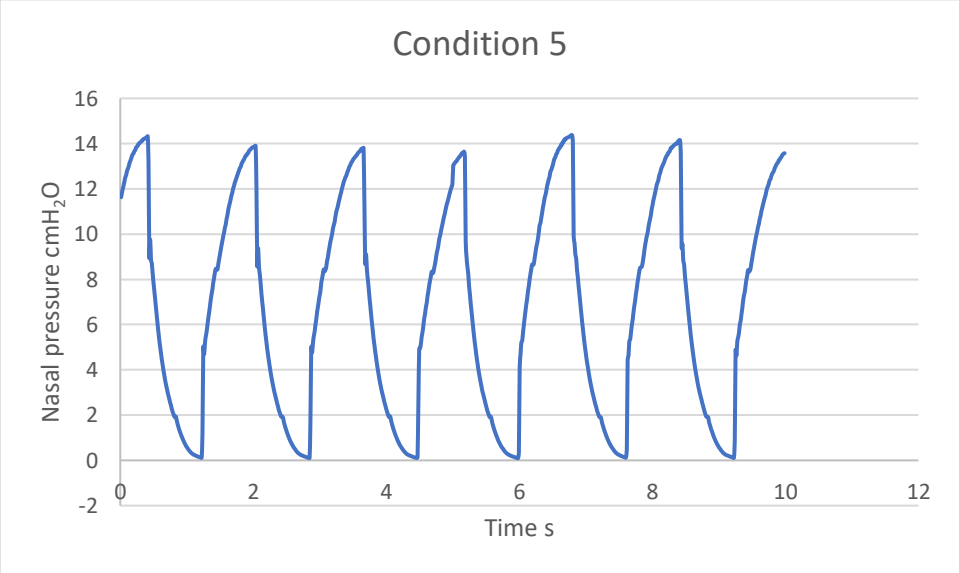
Annex 6 – Pressure tests of the ventilator for all conditions

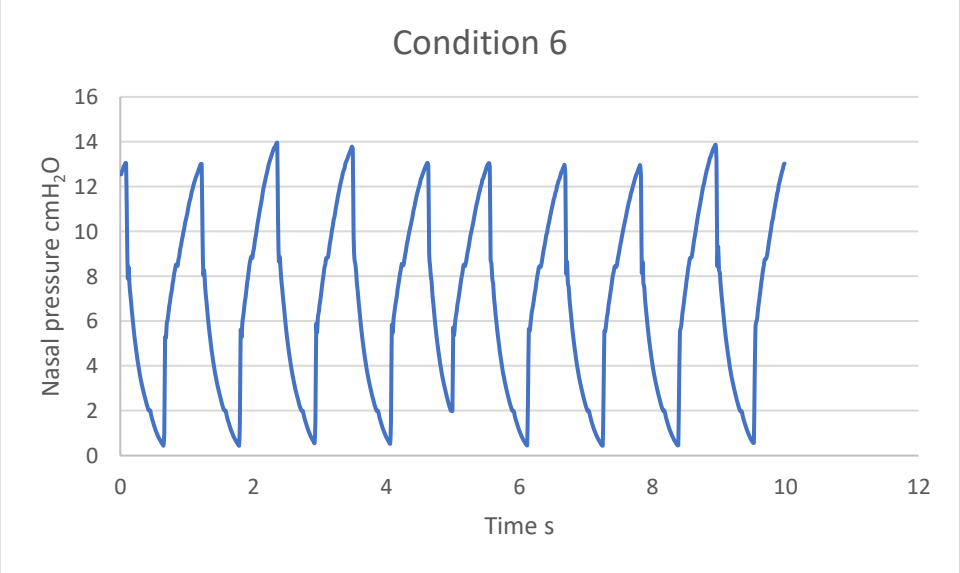


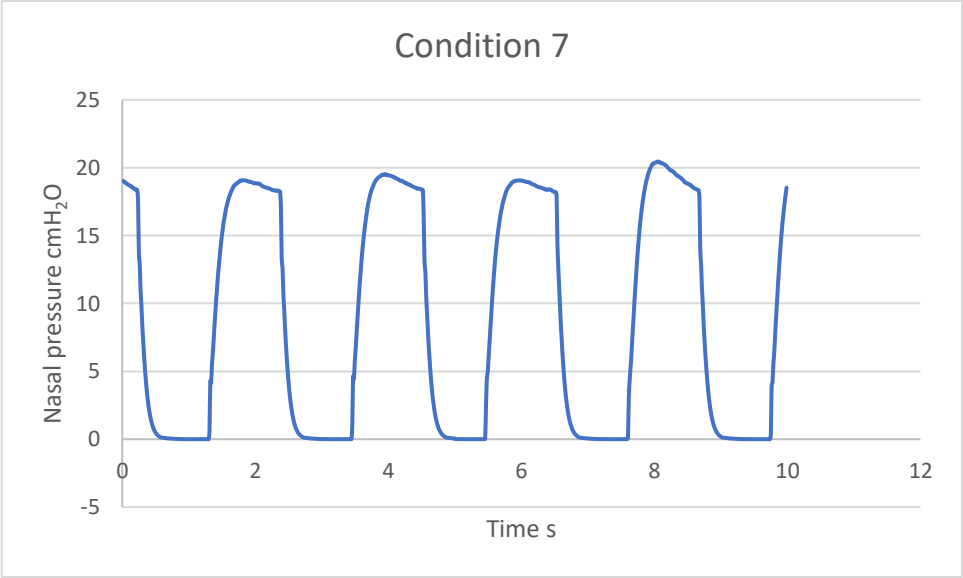


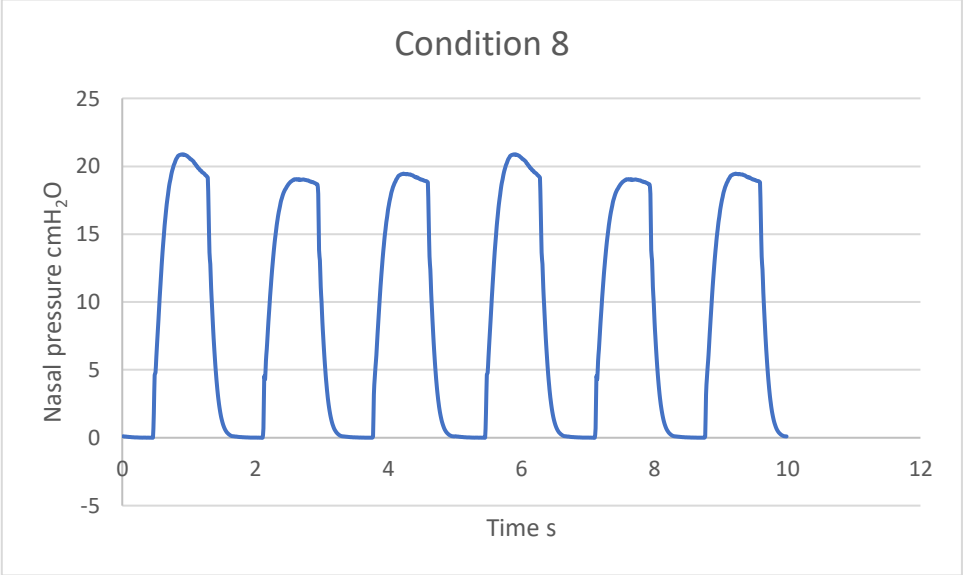


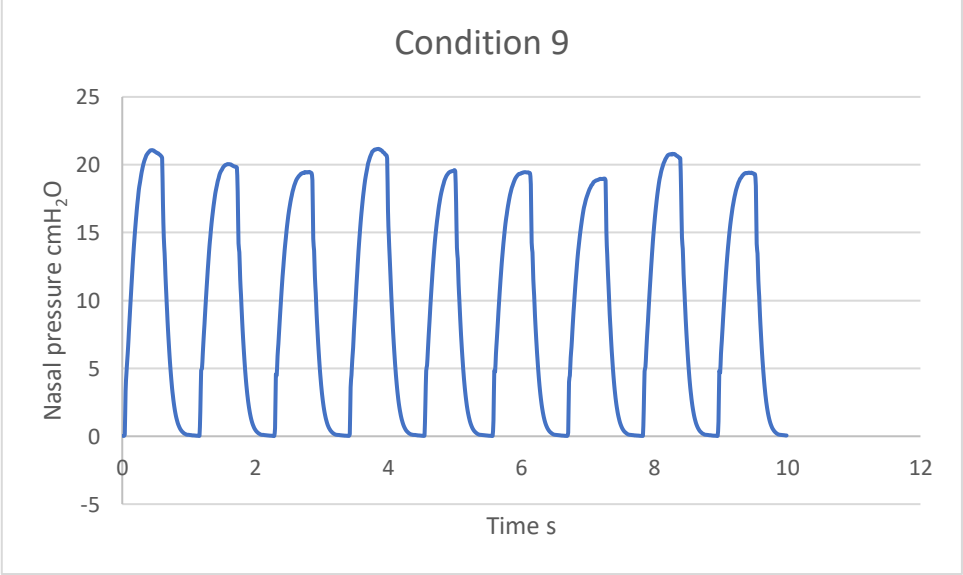

Condition 4
Nasal pressure cmH2O
16
14
12
10
8
6
4
2
0
-2
0
2
4
6
8
10
12
Time s

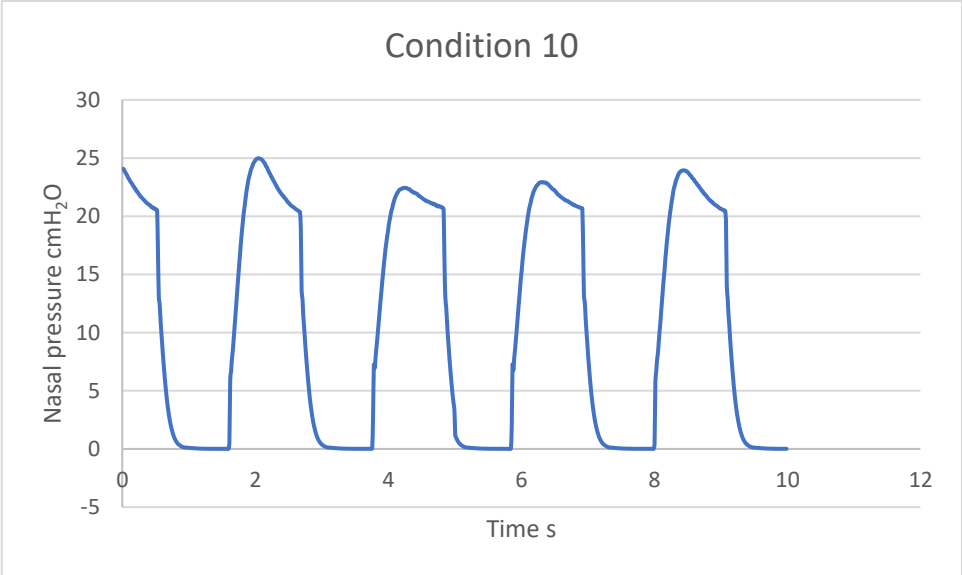

Condition 5
Nasal pressure cmH2O
16
14
12
10
8
6
4
2
0
-2
0
2
4
6
8
10
12
Time s

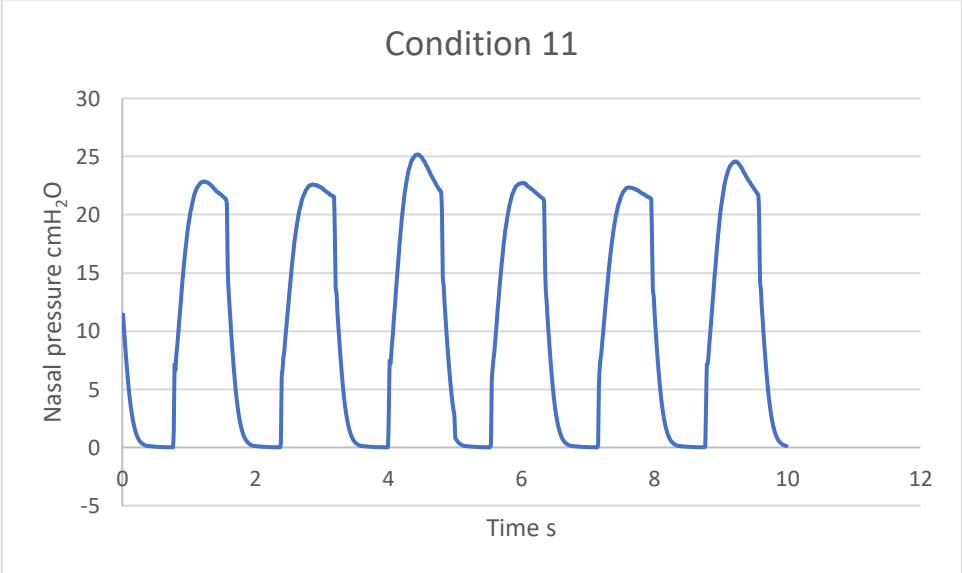

Condition 6
Nasal pressure cmH2O
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
Time s

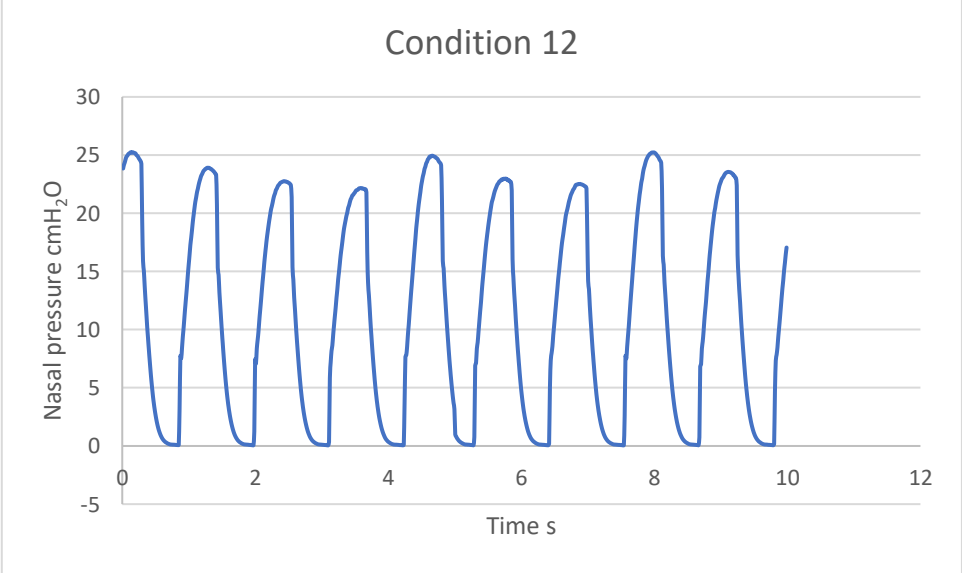

Condition 7
Nasal pressure cmH2O
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s


Condition 8
Nasal pressure cmH2O
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s

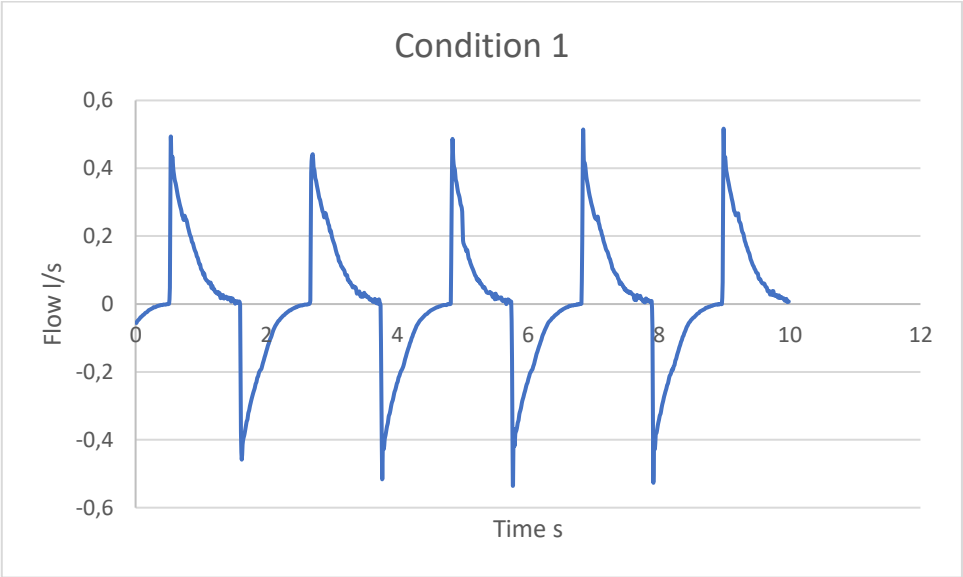

Condition 9
Nasal pressure cmH2O
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s

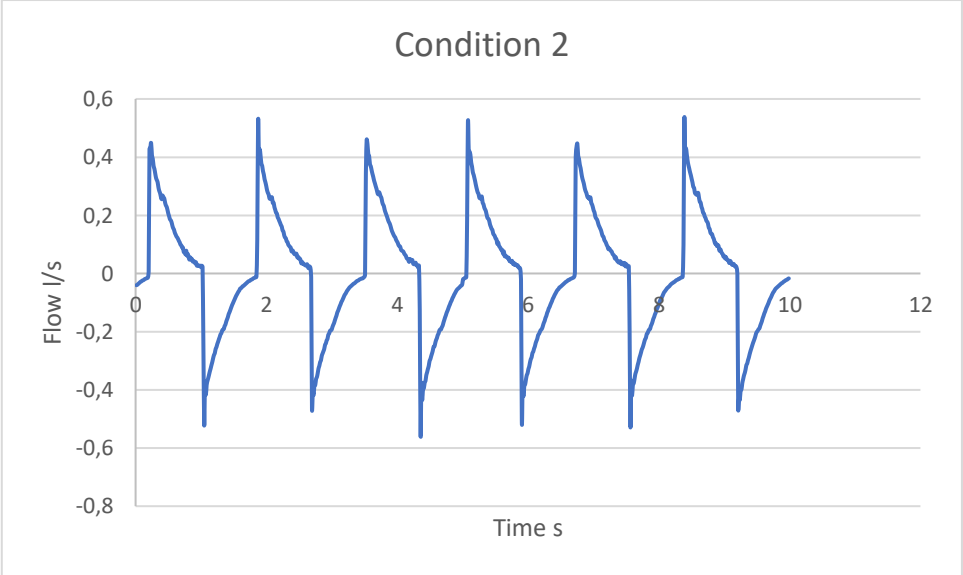

Condition 10
Nasal pressure cmH2O
30
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s

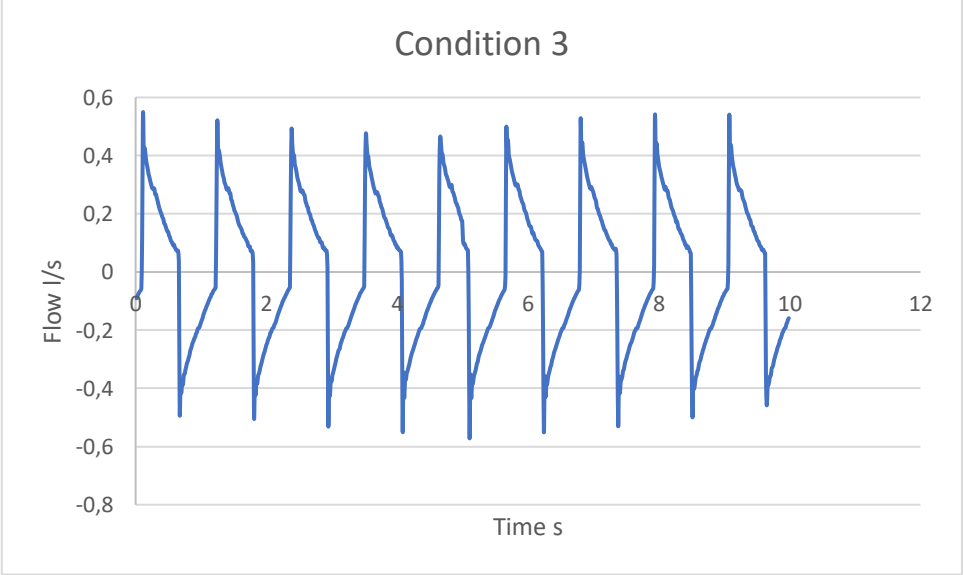

Condition 11
Nasal pressure cmH2O
30
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s

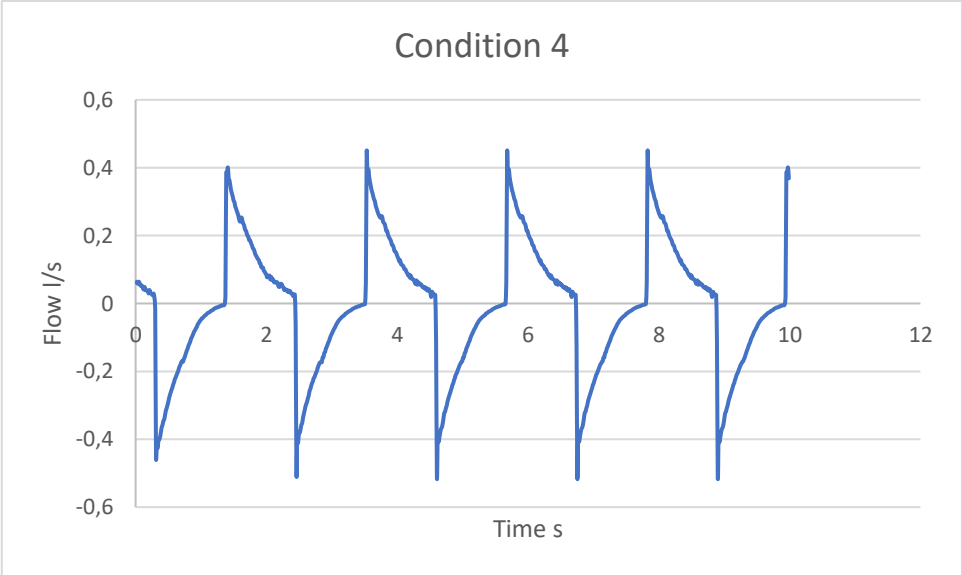

Condition 12
Nasal pressure cmH2O
30
25
20
15
10
5
0
-5
0
2
4
6
8
10
12
Time s

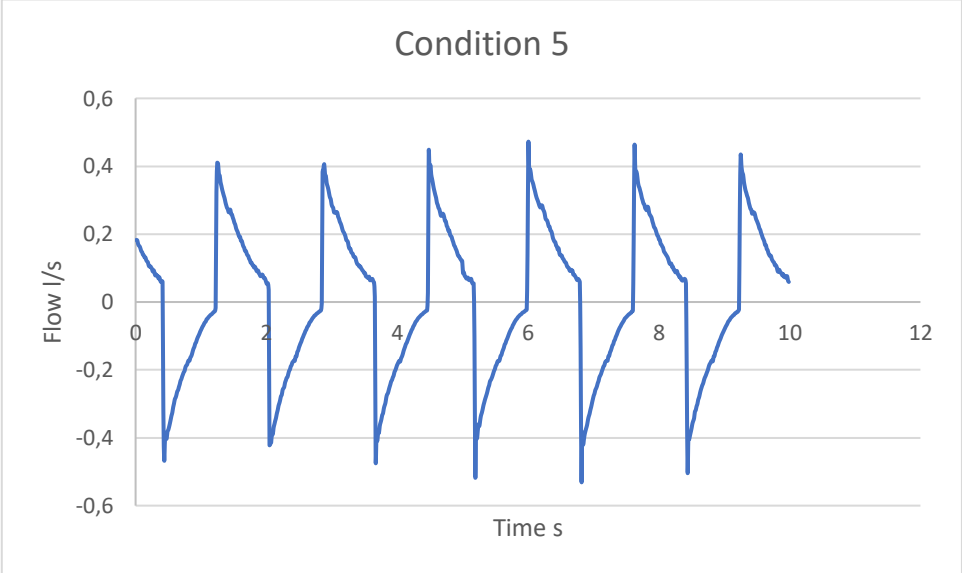
Annex 7 – Flow tests of the ventilator for all conditions

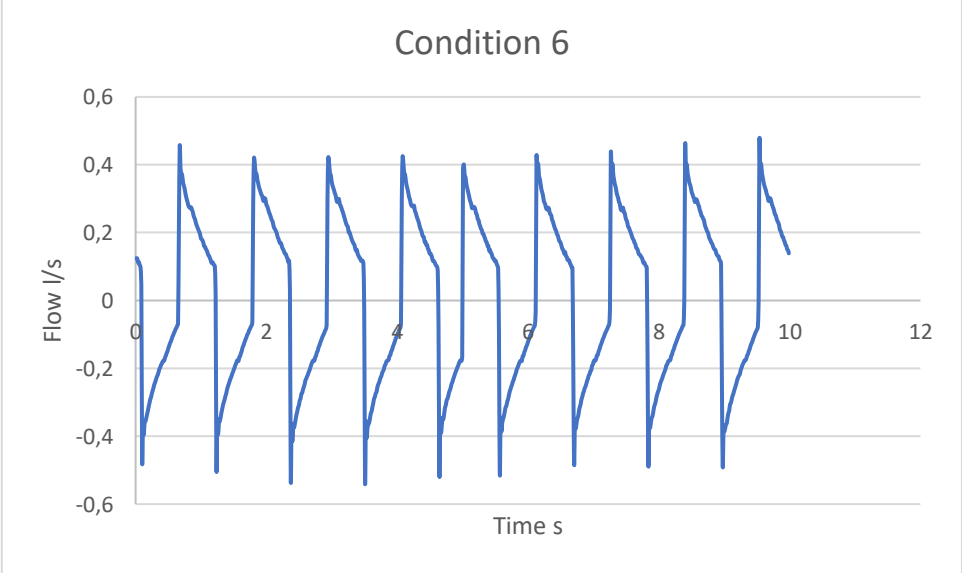




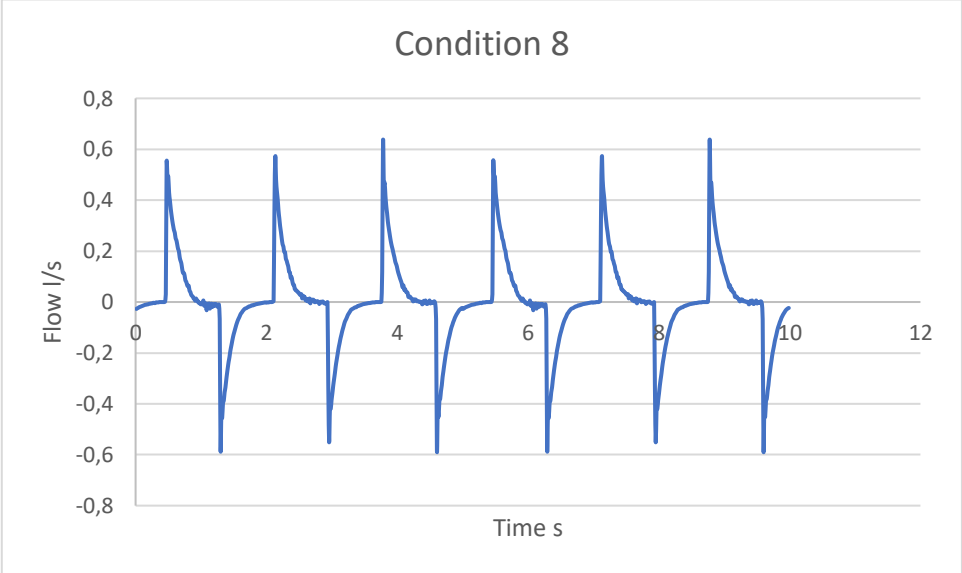


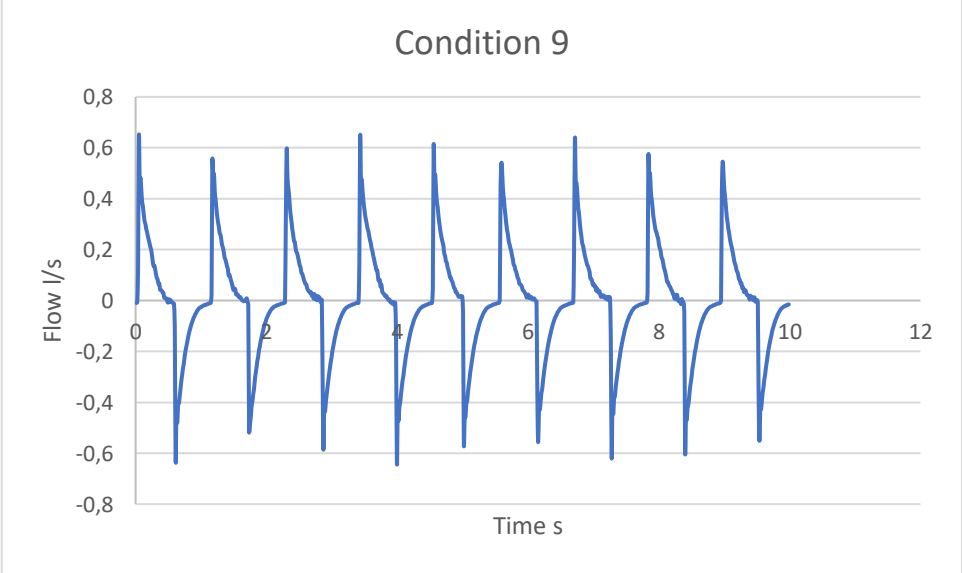

Condition 4
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
Flow l/s
0
2
4
6
8
10
12
Time s

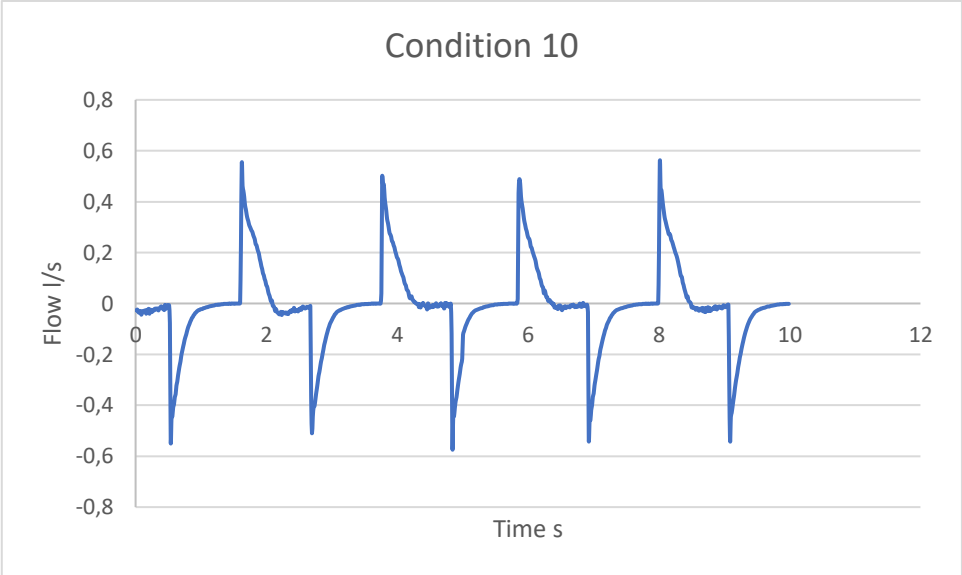

Condition 5
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
Flow l/s
0
2
4
6
8
10
12
Time s

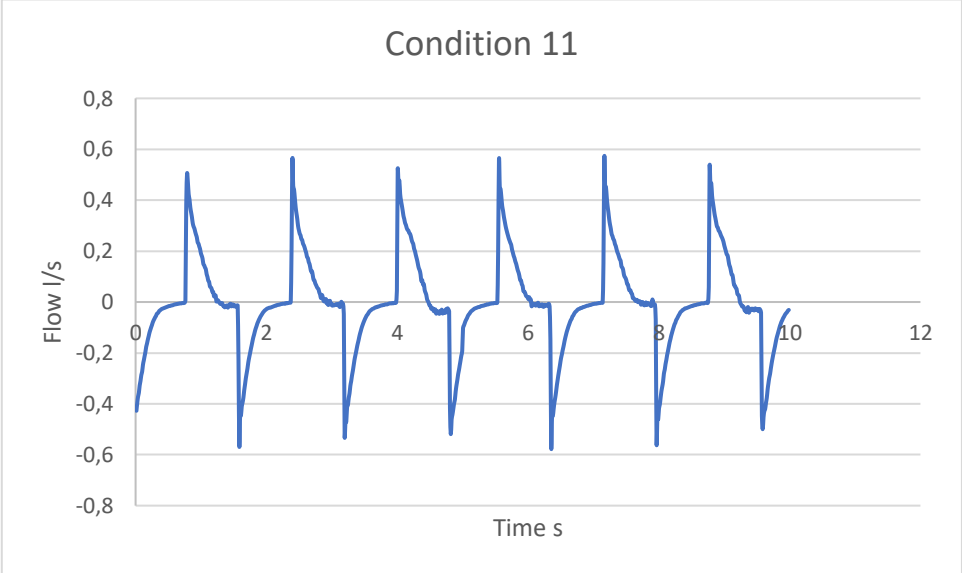

Condition 6
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
Flow l/s
0
2
4
6
8
10
12
Time s

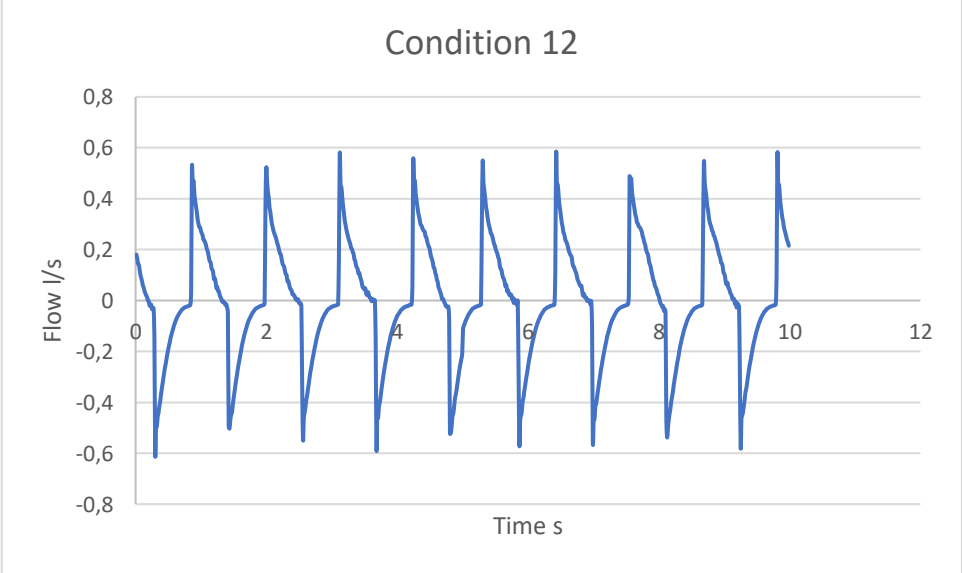

Condition 7
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s


Condition 8
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s


Condition 9
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s


Condition 10
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s


Condition 11
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s


Condition 12
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
Flow l/s
0
2
4
6
8
10
12
Time s

Annex 8 – PERT diagram