SEVEN
publicações acadêmicas

VentiVet: Development of a graphical interface in MATLAB for small animal mechanical ventilators

Crossref doi
https://doi.org/10.56238/sevened2023.006-159

Amanda Fila de Lima
Highest Degree of Education: Medium/Technical
Academic institution: Federal Institute of Paraná

Brayan Agio de Miranda Andrade
Highest Degree of Education: Medium/Technical
Academic institution: Federal Institute of Paraná

Beatriz do Santos Pês
Highest Degree of Education: Doctorate
Academic institution: Federal Institute of Paraná

Diego Tefili
Highest Degree of Education: Master's Degree
Academic institution: Federal Institute of Paraná

Marcos Santos Hara
Highest Degree of Education: Doctorate
Academic institution: Federal Institute of Paraná

ABSTRACT
A pulmonary ventilator is a respiratory support equipment for surgical or clinical procedures, such as those that require general anesthesia, for example. The construction of this type of equipment can involve knowledge of several areas, such as closed-loop control systems, software development, sensing, and communication protocols. This paper presents the development of a pulmonary ventilator for small animals and the implementation of a graphical interface for monitoring and parameter adjustment. The veterinary ventilator built has an Arduino Uno microcontrolled platform, a pressure sensor and ball valves with proportional flow control, built with servo motors, performing the functions of expiratory and inspiratory valve, through a proportional control algorithm implemented in firmware. In addition, the graphical interface is developed in MATLAB, enabling parameter adjustments and visualization of ventilatory data and alarms, which are essential for the safe operation of the ventilator. Unlike other projects presented in the literature, the prototype uses compressed air and not manual, with the implementation of the PCV ventilation mode - ventilation with controlled pressure, in controlled cycles. The prototype also has a dynamic interface with a display of the measured pressure history on a real-time graph, as well as controls and adjustments.

Keywords: Mechanical ventilation, Proportional control, MATLAB, Graphical interface.

1 INTRODUCTION

Mechanical ventilation aims to provide assistance or replace part of the respiratory functions of its user, and can be invasive (prosthesis inserted directly into the airways) or non-invasive (mask as an instrument that performs gas exchange) (CARVALHO *et al.*, 2007).

The mode of operation of the prototype is Pressure *Controlled Ventilation* (PCV) in which the tidal volume (Vt) is variable. The cycles are predetermined by the time and the goal of inspiratory pressure, which must be reached at the beginning of inspiration and maintained throughout this phase (CARVALHO *et al.*, 2007). The prototype built has a digital interface with information regarding the patient's conditions and interactive buttons that allow the customization of the parameters according

to the characteristics of each patient. The prototype aims at a dynamic use, enabling ease of use, preliminarily focusing on the use in veterinary clinics.

In this project, the Arduino Uno microcontrolled platform and the MATLAB *software* are used , in its R2023a version, as it allows the development of applications that communicate directly with Arduino Uno. With these tools, a prototype was built that integrates microcontrolled electronic hardware composed of actuators and sensors, forming a closed-loop controller, as well as *dedicated firmware* and *software*. A specific communication protocol between the microcontroller and the computer was also defined for data exchange. The values read by the plate are stored in the computer, which displays them on the screen, where the operator can monitor the operation of the ventilation, receive alarms in case of failures and change process variables, such as respiratory rate and inspiration time.

The following sections describe the functioning of the respiratory system, the types of mechanical ventilation, the components used in the development of the prototype and the results obtained.

2 GROUNDS

A compilation of some of the knowledge necessary for the implementation of the prototype is presented here, ranging from the principles of mechanical ventilation necessary for a full understanding of the development to the components used and their relevant specifications.

2.1 THE RESPIRATORY SYSTEM

Breathing is an indispensable process for the full execution of tissue metabolic functions and essential for the body. The respiratory system provides the means for this to occur, through the upper airways, composed of the nasal cavities, pharynx and larynx, and the lower airways, composed of the trachea, bronchi, bronchioles and alveoli. The lower airways make up the structure of the lung, where the process of hematosis occurs, removing carbon dioxide (CO2) from the blood and introducing oxygen (O2). This occurs in the smallest lung structure, the alveoli.

Inspiration occurs through the contraction of the respiratory muscles, which causes a drop in alveolar pressure in relation to the external environment, inducing atmospheric air to enter the airways and access the interior of the lungs through its distension. Exhalation follows inspiration, which, on the contrary, causes the volume of air acquired by the relaxation of the muscles to come out, making the lungs return to their initial position through the elastic energy stored in the chest. Thus, respiration is carried out by the difference in pressure created in the upper airways (atmospheric pressure) and the alveoli (intrathoracic pressure) (CUNNINGHAM, 2014).

The breathing process occurs automatically and rhythmically, coordinated by the central nervous system. In different cases, which can range from the consequences of anesthesia to mechanical failure of the respiratory system, it is necessary to introduce mechanical ventilation to help the body carry out the entire breathing process (CAVALCANTE, 2020).

2.2 MECHANICAL VENTILATION

The first mechanical ventilator to be successfully used was the steel lung, in the midst of the poliomyelitis epidemic in 1926, which worked by means of pressure and a non-invasive method, but with little comfort for the patient (CASTRO, M., 2011). According to the advancement of models, the invasive method was adopted, with the introduction of a tube through the mouth or nose to the trachea, with the most common use by anesthesiologists (ROMERO, 2006). The area of use of mechanical ventilators expanded and, in the 1980s, with microcontrollers, it was possible to achieve a higher level of synchrony between patient and ventilator and the choice of different ventilatory modes (CASTRO, M., 2011).

Mechanical ventilation (MV) is now a way to offer ventilatory support for patients with acute or acute chronic respiratory failure. Among the objectives of MV use are: correction of hypoxemia and respiratory acidosis due to increased partial pressure of CO2 (PCO2) in the blood; relieve the effort required by the respiratory muscles to maintain the body in situations of great metabolic demand; reduce oxygen consumption, increasing comfort and allowing the execution of specific therapies (CARVALHO; TOUFEN; FRANCA, 2007). In view of the indispensability of mechanical ventilators for these treatments, according to Brazilian legislation, the presence of these devices in veterinary clinics is mandatory. (NATIONAL COUNCIL OF VETERINARY MEDICINE, 2013).

2.2.1 Principles

Mechanical ventilators inflate the airways with a tidal volume (Vt), where movement can occur by: pressure difference created in the upper airways and the alveoli, which are under less pressure (negative pressure ventilation); increased pressure in the upper airway (positive pressure ventilation). Positive pressure ventilation is more common.

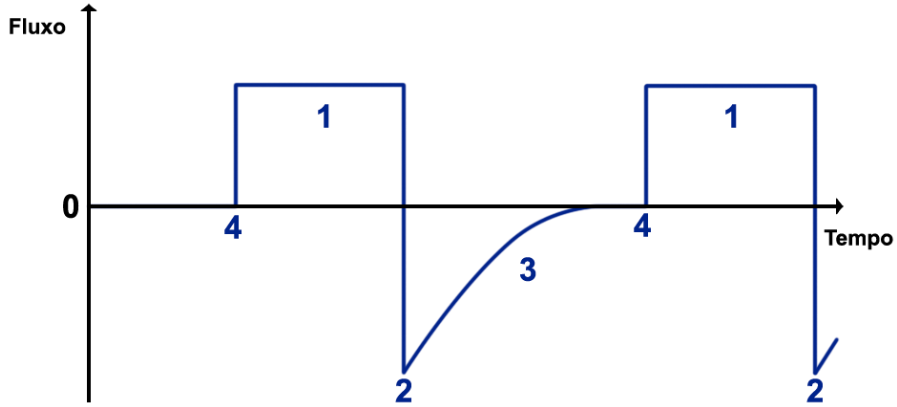
In the air supplied to the patient, the concentration of O2 (FIO2 - Fraction of Inspired Oxygen) can be controlled, necessary to achieve the arterial oxygen rate (PaO2 - Partial pressure of oxygen in arterial blood) that is needed. The speed at which air is administered (inspiratory flow) can also be controlled. The flow waveform (downward, square, upward or sinusoidal) can also be defined, chosen according to the situation.

Respiratory rate (F) is given by the number of ventilatory cycles that occur during one minute, and is a consequence of inspiratory time (Ti), which depends on flow and expiratory time (Te) (CARVALHO; TOUFEN; FRANCA, 2007).

2.2.2 Ventilatory cycle

The ventilatory cycle is the period from the beginning of an inhalation to the release of the ventilator. According to Figure 1, there are 4 periods that describe the ventilatory cycle (CARVALHO; TOUFEN; FRANCA, 2007):

Figure 1 - Flow curve with respiratory cycle indicators



Source: The authors, 2023..

1 - Inspiratory phase: period that begins when the respiratory valve of the mechanical ventilator is opened, allowing the lungs to be inflated according to their elasticity.

2 - Phase Change (cycling): moment in which the transition between the inspiratory phase and the expiratory phase occurs.

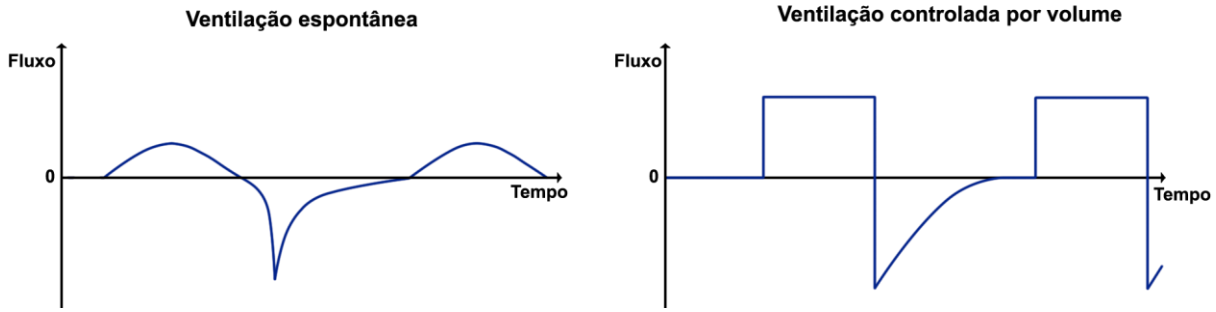
3 - Expiratory phase: occurs after the closure of the respiratory valve, followed by the immediate opening of the expiratory valve. It allows the volume of air introduced into the patient to exit naturally until it reaches the intrathoracic pressure equal to that of the environment or a value determined by the ventilator, called PEEP (*Positive End Expiratory Pressure*), which prevents the collapse of the patient's alveoli by completely emptying the lung.

4 - Change from expiratory phase to inspiratory phase (triggering): once the expiratory phase is over, the ventilator is triggered, opening the inspiratory valve and returning to the inspiratory phase, starting a new cycle.

2.2.3 Graphical Analysis

Flow curve. The flow is measured by pressure sensors present in the fan. It starts according to the device's settings, after a shot, until it reaches peak flow. The most used waveform is square, as shown in Figure 2, which represents the flow in spontaneous and controlled ventilation:

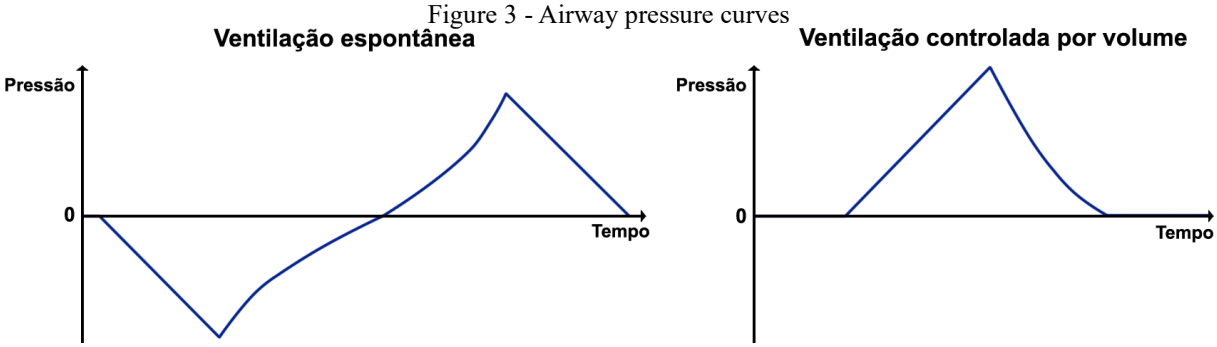
Figure 2 - Flux curves



Source: The authors, 2023.

Pressure curve. As observed in Figure 3, as the airflow enters the respiratory system, the inspiratory pressure increases, as it is necessary to overcome two components: a resistive one (due to the resistance to the flow of air passing through the airways) and an elastic one (due to the distension of the lungs and chest wall).

Figure 3 - Airway pressure curves



Source: The authors, 2023.

In spontaneous ventilation, there is a drop in pressure in the alveoli due to muscle contraction, which does not occur significantly in mechanical ventilation, since the ventilator can meet the patient's need to make an effort.

Volume curve. The volume curve shows the volume inhaled by the patient according to the adjusted parameters. Its maximum and minimum values must be constant, otherwise it may indicate leaks, air entrapment or circuit disconnection.

2.3 MAIN INDICATIONS FOR THE USE OF MECHANICAL VENTILATION

The criteria for applying mechanical ventilation depend completely on the objectives to be achieved. However, in more critical cases, in which there is no time for such assessments of respiratory function, the clinical impression is what should be taken into account the most when starting treatment. The main indications are:

- Resuscitation due to cardiorespiratory arrest;
- Hypoventilation and apnea;
- Respiratory failure due to intrinsic lung disease and hypoxemia;
- Mechanical failure of the respiratory system (muscle weakness, neuromuscular diseases, paralysis, unstable respiratory command, among other cases);
- Prevention of respiratory complications;
- Reduction of respiratory muscle work and muscle fatigue.

In short, the use of MV is indicated when the patient is unable to reach the recommended levels of O2 and CO2 in the blood autonomously (CARVALHO; TOUFEN; FRANCA, 2007).

MV is also used in patients undergoing general anesthesia, so that respiratory functions are not compromised during a surgical intervention. It is essential in such a case if hypnoanalgesics, hypnotics or inhalational anesthetic agents are used in doses high enough to interfere with the spontaneous respiratory cycle (DUGDALE, 2007). Likewise, it is essential when there is an indication for the administration of neuromuscular blocking agents to obtain muscle relaxation that facilitates surgical access.

Other indications include hyperadrenocorticism, in which animals may present generalized muscle weakness due to reduced sodium and potassium pumps in skeletal muscle. This respiratory depression can also be intensified when there is hepatomegaly, an accumulation of abdominal fat that opposes respiratory movements and increases diaphragmatic pressure, leading the animal to respiratory depression (LEAL, 2008 *apud* CASTRO, M., 2011).

2.4 RELEVANT VENTILATORY MODES

Different modes of operation of the mechanical ventilator can be applied and configured to best meet the patient's needs, from critical phases to recovery.

2.4.1 Mandatory Continuous Pressure Controlled Ventilation – Controlled Mode

The respiratory rate, the inspiratory time or the inhalation:exhalation ratio (Ti/Te), and the inspiratory pressure limit are fixed. Triggering occurs according to respiratory rate, but cycling occurs as a function of inspiratory time or Ti/Te ratio. Vt depends on the pre-established inspiratory pressure

and impedance of the respiratory system in question, together with the inspiratory time selected by the operator (CARVALHO; TOUFEN; FRANCA, 2007).

2.4.2 Mandatory Continuous Pressure Controlled Ventilation – Assisted-Controlled Mode

Unlike the controlled mode, the assisted-controlled mode gives the patient a certain autonomy, which can determine, up to a certain extent, the moment of the ventilator firing. Through an effort to breathe, the patient can create a sufficient pressure difference, determined by the operator, for the ventilator to initiate a trip. The Vt obtained will also depend on this effort. If there is no effort on the part of the patient, the ventilator will operate according to the determined respiratory rate (CARVALHO; TOUFEN; FRANCA, 2007).

2.5 MATLAB

MATLAB is a *software* developed by MathWorks, known for its versatility in various applications involving numerical calculations and matrix development. Having its own programming language, it is an environment focused on iterative data analysis, process development, visualization and algorithm development, among others. Through MATLAB, it is possible to create customizable interfaces for control and processing of data received in real time, enabling the plotting and visualization of graphs, matrix manipulation and various tasks, which can be controlled by an operator or not. The availability of various tool packages and code libraries provide functions that can solve various problems encountered in a process (MATHWORKS, 2023).

2.5.1 App Designer

App Designer is an integrated tool of MATLAB that allows you to create *graphical user interfaces (GUIs*) through functions such as "click and drag" of various visual elements that can be displayed on a screen that has several editing options. This tool contributes to the creation of interactive applications without needing extensive knowledge in the field of software development. Even so, the functions of each element on the screen, the alarms and the communication need to be programmed. Apps made on this platform can be exported and run on devices without MATLAB installation, as MATLAB requires a paid license, but only with the MATLAB Runtime, which is free. Thus, editing and programming depend on a license, but the execution of the program does not.

2.6 ARDUINO

Arduino is a microcontrolled platform that aims to prototype low-cost circuits through a simple interface, developed to easily communicate with Atmel microcontrollers. Its programming is done in a variant language of C/C++ and the platform has an open source IDE that helps users design their

applications. His jobs range from a simple turning on and off of LEDs to sophisticated projects that depend only on knowledge of Arduino and electronics (SILVA; AHMAD; CAVALCANTE., 2019).

The Arduino Uno board, one of the variants of the Arduino family, has 14 digital pins, of which 6 are capable of emitting a PWM signal, 6 analog inputs, a USB port, a 16 MHz processor and a reset button. Its nominal voltage is 5 V, with input voltage ranging from 7 to 12 V, within a limit of 6 to 20 V, and emits 20 mA on GPIO pins in direct current (DC). Its mass is 25 g (ARDUINO, 2023).

2.7 MPS20N0040D-D DIFFERENTIAL PRESSURE SENSOR

The MPS20N0040D-D pressure sensor is a solid-state type sensor of microelectromechanical technology, high precision, low cost and easy to use. Its readings reach up to 40 kPa and it is powered by 5 V and 1 mA in DC. The input impedance is 4 to 6 Ω and its output ranges from 50 to 100 mV (ADI; KITAGAWA, 2019).

2.8 SPHERE TYPE VALVE

Valves are mechanical devices that control the flow of fluids in pipelines, with industrial applications, hydraulic systems, and robotic positioning control. Unlike conventional valves, the ball-type valve controls the flow rate in proportion to a control signal via a spherical plug with an operating interval of 90°. This allows for precise adjustments to the pressure or flow rate of the fluid. These valves are low-cost and excellent for proportional control, with the disadvantage of not having a linear flow rate in relation to the control signal (SILVA, H., 2019).

2.9 MOTOR CONTROLLER

Its function is to generate a pulse *width modulation (*PWM) signal for the motor. This signal comes from the direct current (DC) supply, and is transformed into current, thus activating the phases of the servo motor (SOKIRA; JAFFE, 1990 *apud* DINAU, 2009). Its main characteristics are high precision and high performance, which are due to the closed-loop operation, which occurs according to the feedback of the position indicated by the sensor coupled to the shaft (CASTRO, F., 2016).

PWM modulation means that the average voltage value can be changed with each stator winding during the switching sequence (SOKIRA; JAFFE, 1990 *apud* DINAU, 2009), causing the maximum voltage to be turned on and off for each phase, in such a way that the average voltage corresponds to that indicated in the microcontroller code. Some of the advantages of using PWM to control the speed of a motor are (WIBERG, 2003 *apud* DINAU, 2009): great practicality in association with a microcontroller, no loss of power in the battery, if used at low speeds and higher torque in the motor, when the use of maximum voltage occurs in short intervals of time.

The PCA9685 integrated circuit is an implementation of a PWM controller that has 16 channels, making it possible to connect to various servo motors. The communication is of I2C type and has high accuracy and efficiency. The controller has two terminals that allow the external power supply of the servo motors, to avoid damaging the entire circuit to which it is connected and to protect against polarity reversal.

2.10 SERVO MOTOR

DC motors, when they receive voltage, produce magnetomotive force, which transforms electrical energy into mechanical energy. Its main objective is linked to speed control associated with torque (SIMONE; NASCIMENTO JUNIOR, 2000, 2006 *apud* CASTRO, F., 2016). The speed control of servo motors is directly linked to the amount of tension applied (CASTRO, F., 2016).

The MG995 model operates in DC, has brushes, high speed and does not depend on codes for its operation. It operates at a range of 120º, traveling 60º in 0.2 s with 8.5 kgf/cm powered by 4.8 V and 60º in 0.16 s with 10 kgf/cm at 6 V. It supports a maximum voltage of 7.2 V and operates at temperatures from 0 to 55 ºC. Its metal mechanisms promote longer life and it is shock resistant, consisting of two-row rollers (MG995, 2023).

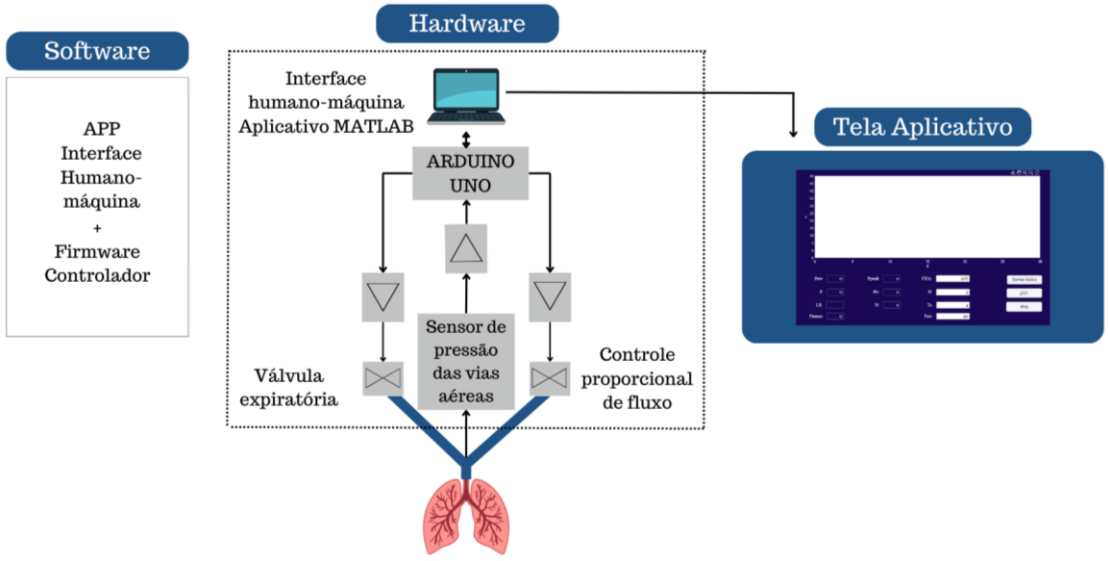
3 MATERIALS AND METHODS

After collecting information on pulmonary ventilators in small animals, the construction of a prototype of a pulmonary ventilator capable of providing controlled pressure ventilation (PCV) in controlled mode was initiated, because this is the most commonly used ventilatory mode and because of the possibility of its implementation without the use of flow sensors. but only with pressure sensors. Pressure sensors are simpler, cheaper, and easier to acquire.

Figure 4 illustrates the proposed structure for the prototype. The arrows indicate the direction of the flow of information and control commands, generically, from the sensor to the controller and from the controller to the valves. The communication between the controller and the interface device, which will be covered in more detail later, is bidirectional, so that the commands and parameter adjustments made by the operator are sent to the controller, which, in turn, must respond according to the new configuration. The controller and HMI exchange monitoring, diagnostics, and alarms information related to the process. The triangles on the controller ports represent signal conditioners. The expiratory valve should only assume the closed and fully open states, allowing, in the latter case, the natural compression of the patient's rib cage to expel the air. The inspiratory valve, however, must open partially so that the volume to be injected during the period of inspiration corresponds to that fixed by the operator. The peak inspiratory pressure, the pressure level for assisted cycle triggering,

and the inspiratory time, as well as the respiratory rate, are also manually adjustable parameters, and the valve opening level is calculated by the system.

Figure 4 - Block diagram of the project structure



Source: The authors, 2023.

For the development of the closed-loop controller of the mechanical fan, the Arduino Uno was chosen. Its affordable cost, extensive documentation and simplicity of use make the platform attractive for didactic projects involving pedagogical projects. In addition, it has some GPIOs (*General Purpose Input/Output*) ports, including analog ones. This is essential for reading the sensors required for the prototype to function, for controlling valves and other actuators, as well as for transmitting data to the developed application, which configures the interface. The application used for programming the Arduino Uno is the Arduino IDE itself.

Prior to the choice of the Arduino Uno, an ESP-32 was being utilized. However, its input and output voltage of 3.3 V was insufficient to power the HX711 converter and represents problems for the reading of the sensors, along with the incompatibility with other components that operate in 5 V. In addition, its application examples were considered quite limited, when compared to those available for Arduino-type boards.

For the development of a GUI, MATLAB was the chosen program. Initially, the Blynk software was considered, but essential functions for VentiVet were out of reach in the free version. In addition, the transmission of information took place through a cloud server, making response time a concern and the dependence on a permanent connection to the Internet would bring risks of critical failures in the process.

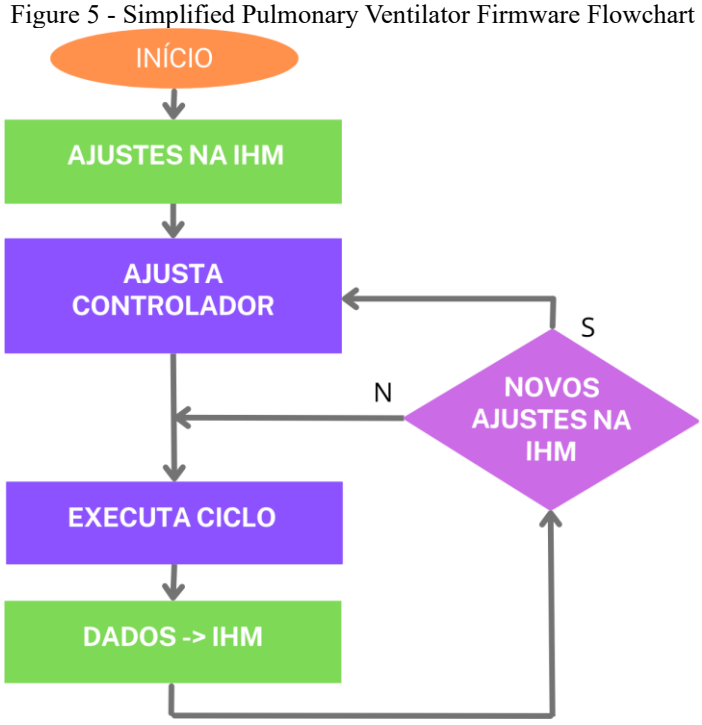
As a second option, through MATLAB, an interface was created for the operation of the veterinary ventilator, with numerical inputs and buttons with options for controlling various parameters, in addition to displaying, through graphs, the data collected by the ventilator sensors. The

data is transmitted from the Arduino Uno to the MATLAB through a serial connection and the synchronization of the processes is done through software interruptions. The app can be compiled and exported to run on devices that do not have MATLAB development software installed, but only the free MATLAB Runtime package.

In Figure 5 and Figure 6, simplified flowcharts of the *firmware* embedded in the Arduino Uno and the *software* developed for the HMI can be seen, respectively.

In the controller flowchart, it is evident that the control parameters are received from the HMI after adjustments made by the user. Subsequently, the closed-loop controller implements the inspiratory and expiratory phases of the respiratory cycle, monitoring the pressure sensor and acting on the proportional and expiratory valves. Throughout the entire cycle, the controller sends data to the HMI. The fan will change its settings only when the "Send Data" button is pressed.

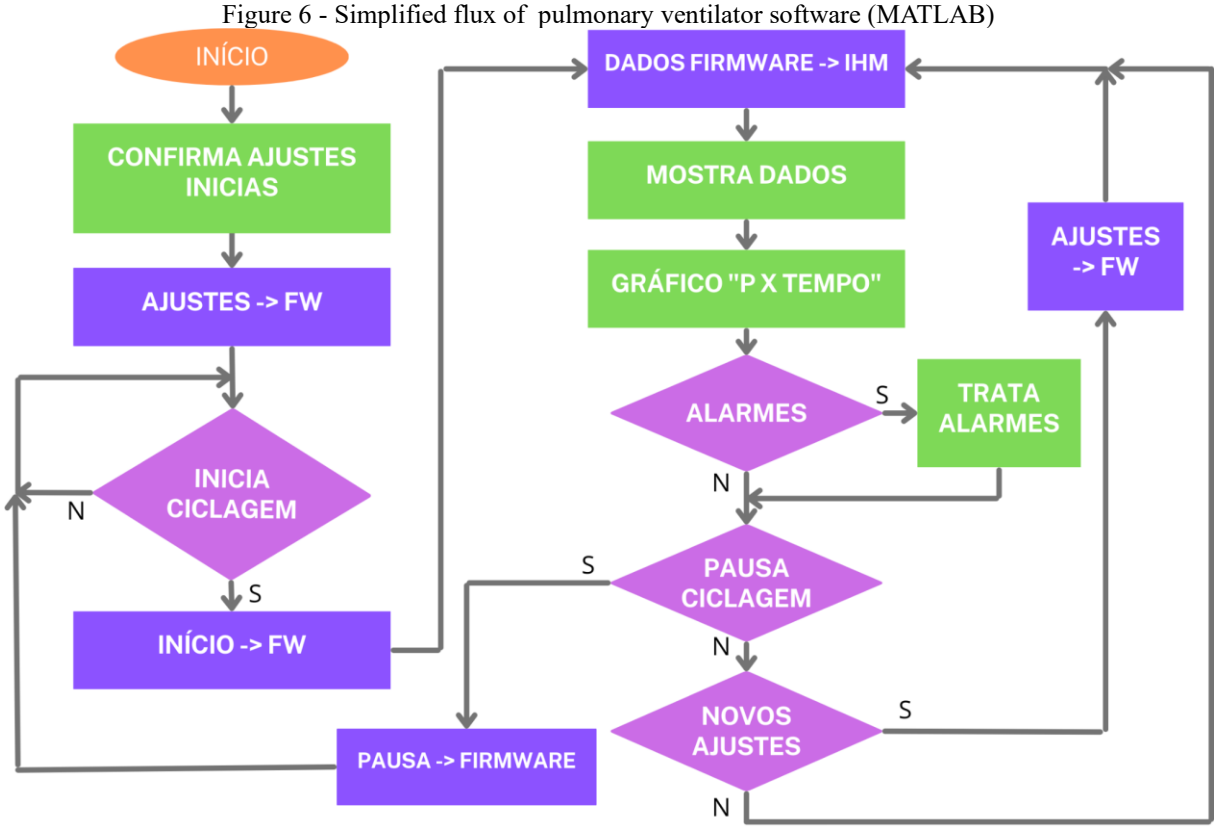
Figure 5 - Simplified Pulmonary Ventilator Firmware Flowchart



Source: The authors, 2023.

As shown in the HMI flowchart (Figure 6), the operator needs to confirm the initial settings before starting the controller operation. All cycling parameters are sent to the *firmware*, which initiates and controls the cycle itself, periodically sending data to be shown on the HMI. In addition to the numerical values, the measured values of the airway pressure are presented in a graph with the evolution of the parameter in the last 30 seconds. If alarms are triggered in the cycle, the HMI triggers the corresponding devices and dials to alert the operator. If the operator requests the interruption of the respiratory cycle, the *software* sends data to the *firmware*, which interrupts the cycle until it is requested to restart. Any new parameters informed by the operator are sent to the controller for its adjustment.

Figure 6 - Simplified flux of pulmonary ventilator software (MATLAB)



Source: The authors, 2023.

Flowcharts illustrate how *firmware* and *software* operate independently but relatedly. If there are problems in serial communication, the Arduino Uno's controller firmware continues to ventilate the patient according to the latest adjustments.

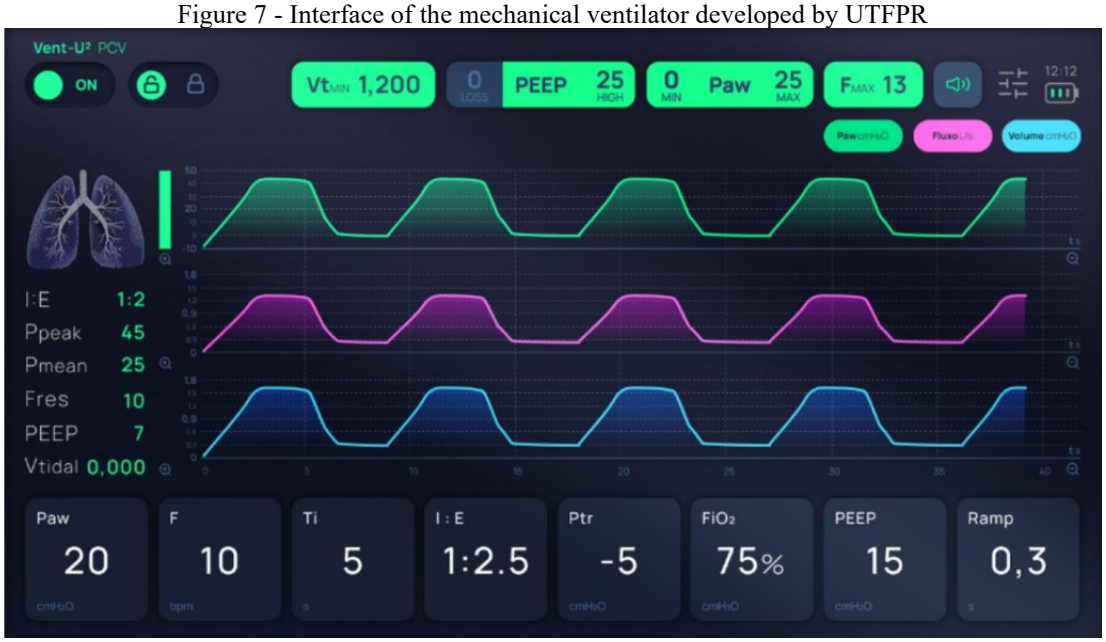
In order to ensure a constant cadence in the operations of reading sensors and actuating the control, interrupts of the controller are used, in order to prioritize these actions. A timer that runs parallel to the controller's firmware will trigger an interrupt every 10 ms, which will suspend lower-priority states, such as communication with the HMI and arithmetic operations of proportional control algorithm calculations. This constant rhythm is important to keep the sampling frequency stable, making the control system time-invariant, a condition outside of which the behavior of the control actually performed would differ from the calculated algorithm. This, in turn, will consist of a proportional controller that will target the upper airway pressure value, determined by the operator. This target pressure value is compared to the pressure measured by the sensor. Based on this difference, the level of inspiratory valve opening is calculated.

3.1 PARAMETERS

In the following topics, the parameters involved in the ventilation process, which are relevant to the operator, prototype operation, and communication between devices, will be addressed. The HMI was inspired by the prototype of a mechanical ventilator developed by the Federal Technological

University of Paraná (UTFPR). The interface of the same, made by the University, is available in Figure 7.

Figure 7 - Interface of the mechanical ventilator developed by UTFPR

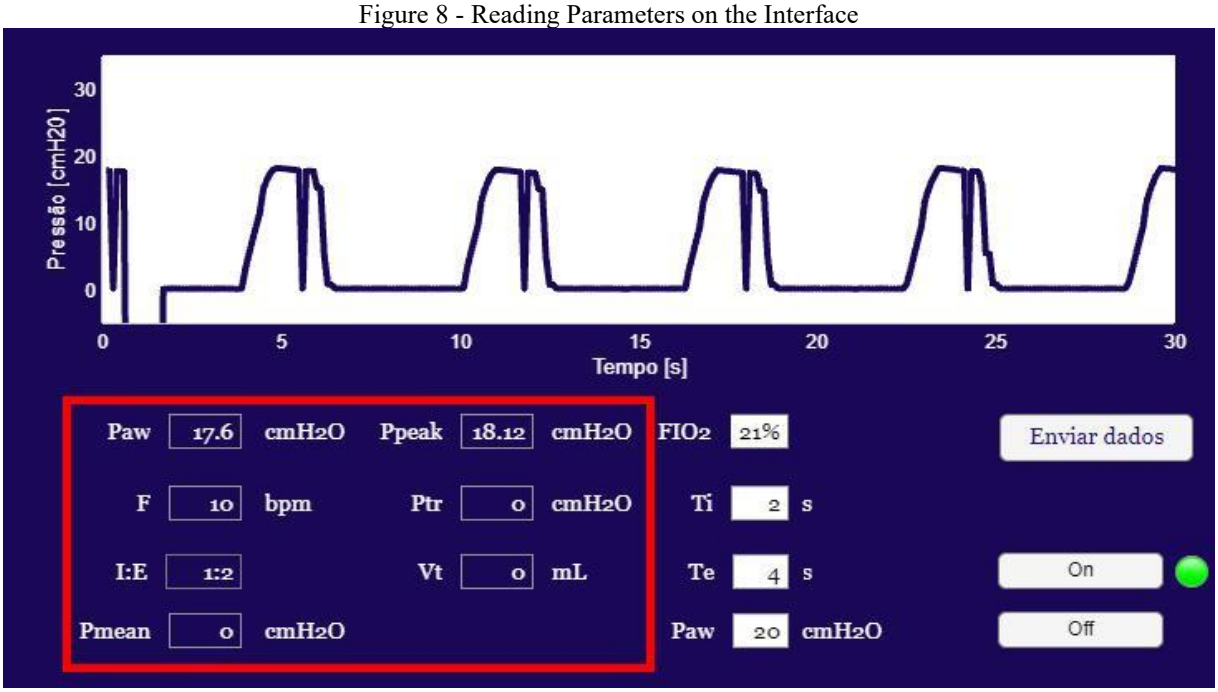


Fonte: UTFPR, 2020.

3.1.1 Reading Parameters

The non-editable parameters consist of numerical boxes, which are not related to any change in the configuration of the respiratory process, but rather to the reading of their physical quantities. Figure 8 demonstrates an example of how these parameters can be visualized with the prototype in operation.

Figure 8 - Reading Parameters on the Interface



Source: The authors, 2023

There are four functional reading parameters present in the interface. They are:

Paw (*Airway Pressure*). This parameter is linked to the reading of the pressure sensor, being changed in real time, and expressed in centimeters of water [cmH2O], according to the variation of pressure within the test lung, which is very similar to a lung of a living being, since its internal pressure would also be altered proportionally to the operation of the ventilator.

F (respiratory rate). the frequency corresponds to the number of breathing cycles sent from the VentiVet to the patient per minute, and is directly linked to the Ti and Te values, which can be edited by the user. The frequency is expressed in bpm - *breaths per minute* .

As previously mentioned, the rate cannot be disassociated from the inspiratory and expiratory time. With this, the calculation of breaths per minute is done as follows:

- Reading of Ti[s] and Te[s] after pressing the "send data" button;
- sum of the values read (since a respiratory cycle corresponds to the beginning of the inspiratory time until the end of the expiratory time);
- division d 60 (number of seconds in a minute) by the value generated after summing,
- disposition of the value generated in the numerical box.

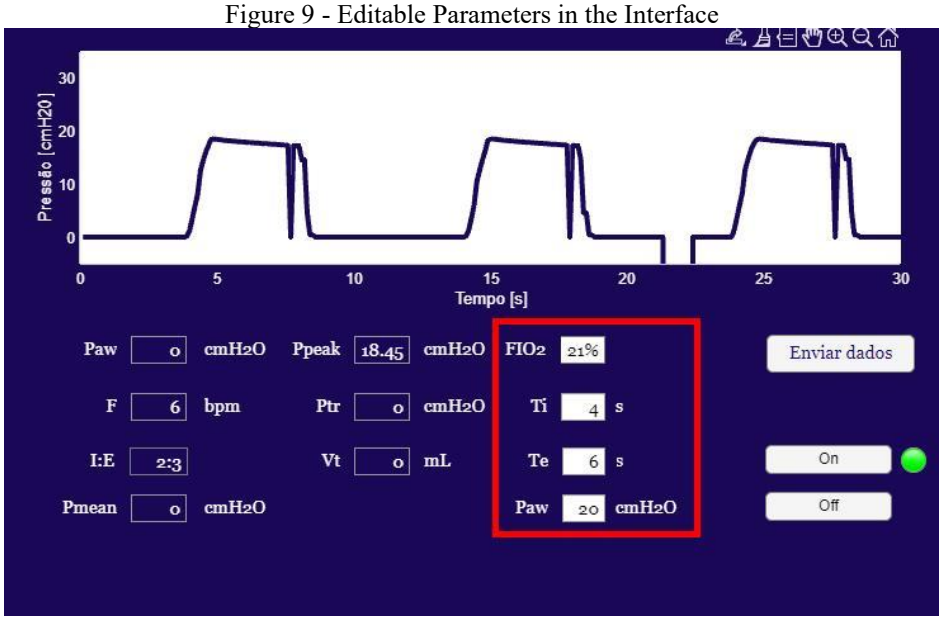
I:E (inhalation-exhalation ratio). like frequency, I:E is also intrinsically linked to the values of Ti and Te. The operation of this reading is as follows: first, a change in the values of Ti and Te is necessary, as they start at zero. After that, there is a reading of these values and the greatest common divisor of them is calculated by the *gdc* (*greatest common divisor*) function, so that the ratio is in its most simplified form. In this way, Ti and Te are divided separately by the largest common divisor, and the variables are transformed into strings. Thus, the I:E ratio can finally be displayed in the HMI.

Ppeak (peak pressure). The value expressed in this parameter refers to the highest pressure level reached in that respiratory cycle and is reset at the beginning of each cycle. To perform the reading at the correct time and develop an ideal logic in the algorithm, it is necessary to immediately identify what phase the respiratory cycle is in. For this, one more piece of information was added to the messages sent by Arduino: whether the process is in the expiratory or inspiratory phase. This quantity is expressed in centimeters of water (cmH2O.

3.1.2 Editable parameters

Editable parameters, such as reading parameters, are also numeric boxes, with the difference that their values can be edited by the operator. Changing these values allows you to directly change cycling. The red rectangle in Figure 9 highlights the location of these elements in the interface:

Figure 9 - Editable Parameters in the Interface



Source: The authors, 2023

Ti and Te (expiratory time and inspiratory time). The determination of the inspiratory and expiratory values, in seconds [s], together with the pressure objective, are the basis for the operation of the prototype. The operation of such parameters occurs according to the Arduino code and their change is made directly by MATLAB, which communicates with the microcontroller.

Paw (editable version). It is used to determine the target airway pressure in centimeters of water [cmH2O]. The servo motor that controls the inspiratory valve will make adjustments to its opening so that the pressure objective is reached, according to the microcontroller's programming.

3.2 ARDUINO-MATLAB COMMUNICATION

Communication between the Arduino Uno and the interface in MATLAB is carried out by serial communication through the USB port of the device that displays the interface and the microcontroller.

Although it was possible to write and read information from the analog and digital ports of the Arduino through MATLAB by algorithms, communication was limited to this, with no possibility of changing the *firmware*, being insufficient to carry out the desired applications. To eliminate this obstacle, a communication protocol was developed inspired by the Modbus protocol, common in the integration of industrial networks and widely used in industrial automation.

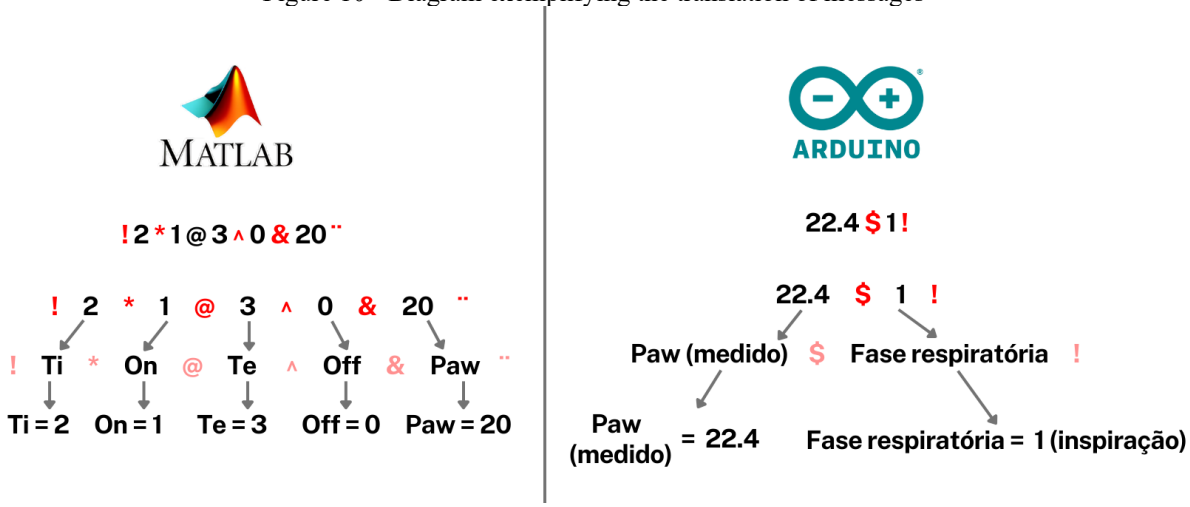
Communication can be considered the most challenging achievement developed in this project, as no examples were found that used simple methods and there were doubts about the possibility of achieving an ideal communication for the prototype. One of the ways to send and receive information between MATLAB and Arduino is through the creation of a text file using the "fopen" function, in the code executed in MATLAB, which writes the information in a text file that needs to be accessed constantly. However, this does not have the same efficiency as the communication protocol developed.

The protocol created consists of sending and receiving messages between the algorithm in MATLAB and Arduino. These messages are purely *strings* containing the information and parameters needed to control and monitor the breathing process. Here's a real example of messages sent from MATLAB to Arduino and the other way around, respectively:

- MATLAB for Arduino Uno: !2*1@3^0&20¨
- Arduino Uno for MATLAB: 22.4$1!

Figure 10 visually demonstrates the separation of the messages and the identification of the parameters present in each one.

Figure 10 - Diagram exemplifying the translation of messages



Source: The authors, 2023.

The principle for both algorithms is the same: concatenate information gathered from the process for which it is responsible and send it as a *string* to the other. In this *string,* the information is separated by special characters.

Despite achieving the same result and having the same beginning, the interpretation processes in each algorithm are different. In Arduino, there are predefined functions that have made the job significantly easier, allowing the separation of the information in the *string* from the delimitation of different special characters[1], as exemplified in the following single line of code:

```
New_InspP_Tg = Received_Info.substring(Received_Info.indexOf('&') + 1,
Received_Info.indexOf('¨')).toInt();
```

The information that carries the pressure target, defined by the interface, which is between the characters "&" and "¨", is extracted from the *string* and saved in a new variable that will be compared

[1] Special characters can be changed freely as long as both programs are configured accordingly. In the version described in this text, the only limitation found is the use of equal characters.

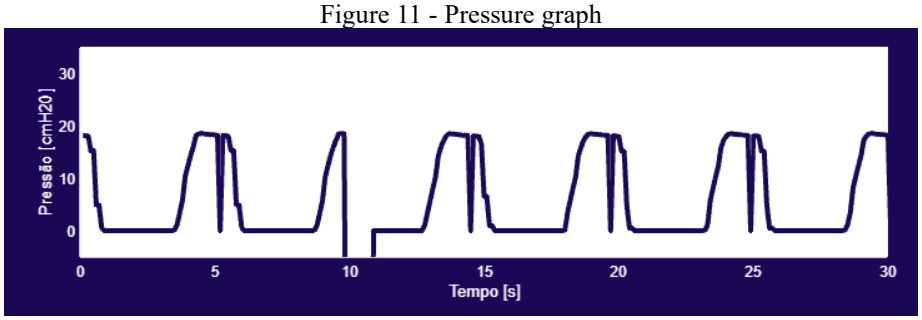
with the current pressure target and then updated. Decoding the data does not have to be done in a specific order.

In MATLAB, messages are separated with only a single special character, and a second special character indicates the end of the message[2]. The *string* is broken into parts from the location of these characters, and the information is stored in an array. The values referring to each piece of information sent are accessed through the pre-defined indexing of this array, which is updated every 100ms, which is the send rate configured in *the* Arduino *timer software* interruption.

3.3 PRESSURE CHART

The construction of the graph (Figure 11) in the interface is done by reading the values stored in an array that is used to plot the graph.

Figure 11 - Pressure graph



Source: The authors (2023)

The graph is updated every 100ms and its horizontal axis is on a scale of 30s, with the vertical axis displaying the pressure taken at the moment in centimeters of water (cmH2O). The pressure is taken by the HX711 pressure sensor, which is automatically calibrated to always have the initial pressure as a null reference.

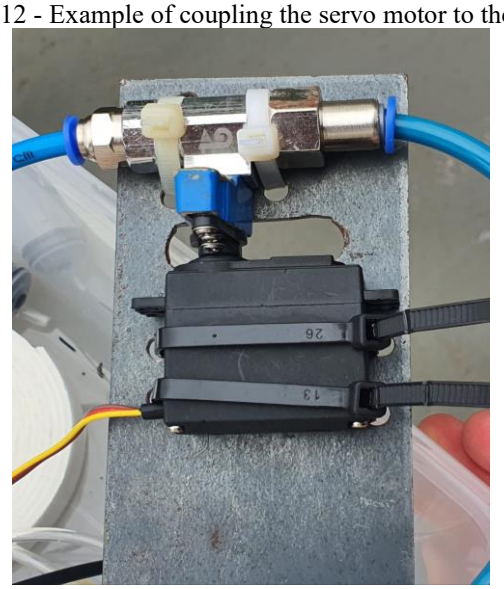
3.4 OPENING AND CLOSING VALVES

The inspiratory and expiratory valves are both of the ball type. Only the inspiratory valve is included in a proportional control, performed by the PID algorithm configured in the *microcontroller* 's firmware. The same type of control is not necessary for the expiratory valve, since it will assume only two states: open or closed, considering that there is still no functional PEEP in the prototype.

The valves are coupled to MG995 servo motors, as shown in Figure 12. The control of these servos must be precise, since if their movements are insufficient, the valves remain open and pose a great risk to the patient. And, if their movements go beyond what is necessary, the servo shaft will lock on the valve lever and its internal mechanisms will be damaged, compromising the entire process.

[2] In the case of the MATLAB code presented, the $ (dollar sign) was chosen to separate the information, and the ! (exclamation point) to signal the end of the sent message.

Figure 12 - Example of coupling the servo motor to the valve



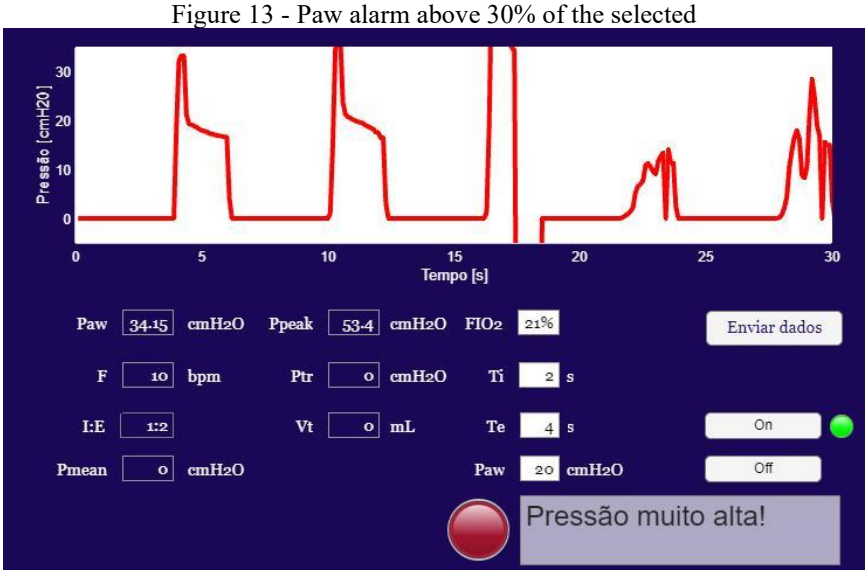
Source: The authors, 2023

The servos receive a PWM signal from the PWM driver PCA9685, which in turn is controlled by the *firmware*, configured according to the settings in the HMI. This controller is powered by an external 5 V supply.

3.5 ALARMS

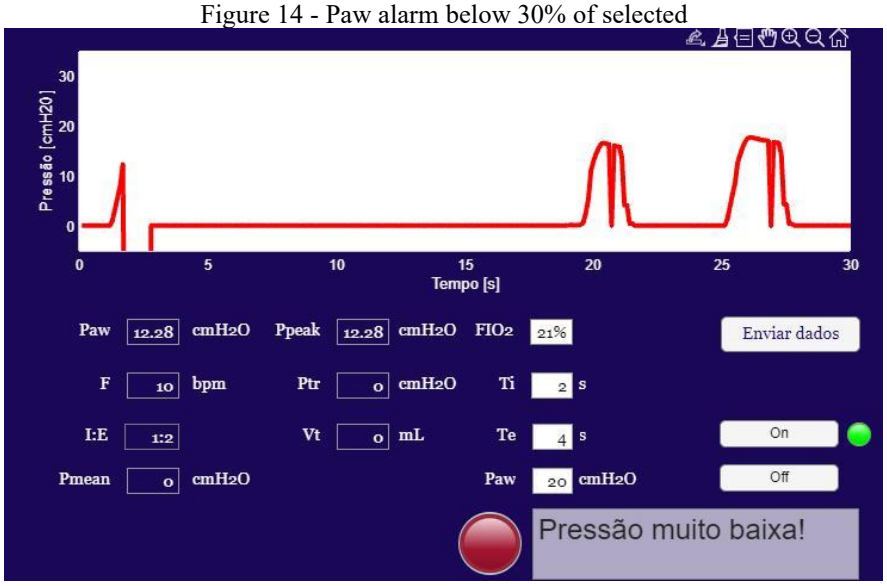
The prototype includes alarms to identify irregularities in the breathing process and eventually solve them. In the prototype version described here, two events are identified: airway pressure that is too high or too low. When these irregularities are identified, the graph line turns red and a warning is displayed in the corner of the interface, as can be seen in the simulations illustrated in Figures 13 and 14. These alarms are activated when the Paw is 30% above or 30% below the selected pressure level.

Figure 13 - Paw alarm above 30% of the selected



Source: The authors, 2023.

Figure 14 - Paw alarm below 30% of selected

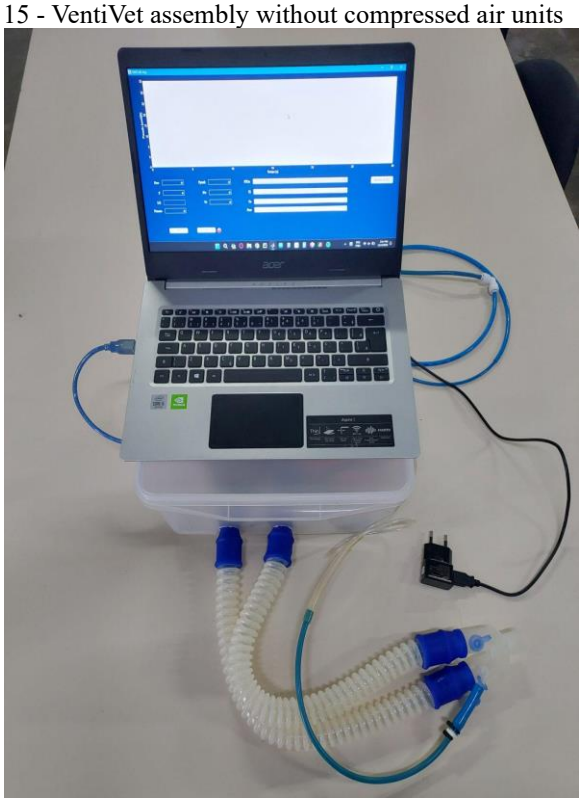


Source: The authors, 2023.

4 RESULTS AND DISCUSSION

The prototype developed, shown in Figure 15, stands out for its portability and accessibility, especially in relation to cost. Because, unlike the devices available on the market, VentiVet can have its interface displayed on any computer, including those that do not have a MATLAB software license. The fan can also be transported easily and safely, as long as all its electronic components are adhered to the box that carries them, and the device that displays the interface is, most of the time, quite compact.

Figure 15 - VentiVet assembly without compressed air units



Source: The authors, 2023

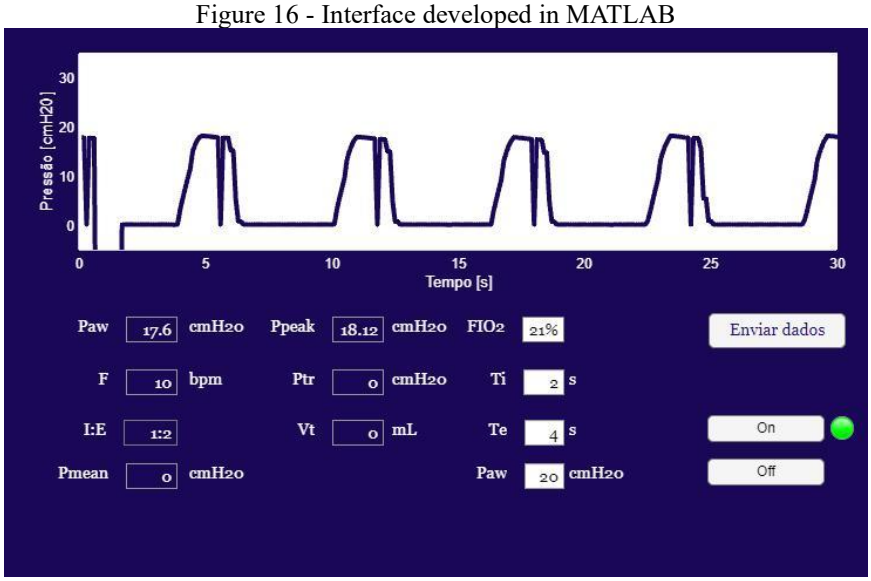
The communication protocol developed with the use of special characters as the identification key for each parameter proved to be very satisfactory in the application. Because it associates both codes used quite sufficiently and quickly, bringing more safety in the operation of the fan.

The configuration of software interrupts in the *firmware* initially posed a problem, since the *timers* that the Arduino Uno has are responsible for functions beyond those programmed, interfering with parts of the process, such as the operation of analog ports and serial communication if changed.

To get around these problems, Timer *1 timer* was used. A reliable timer is indispensable, as it is through this timer that the graph is kept up to date at constant intervals.

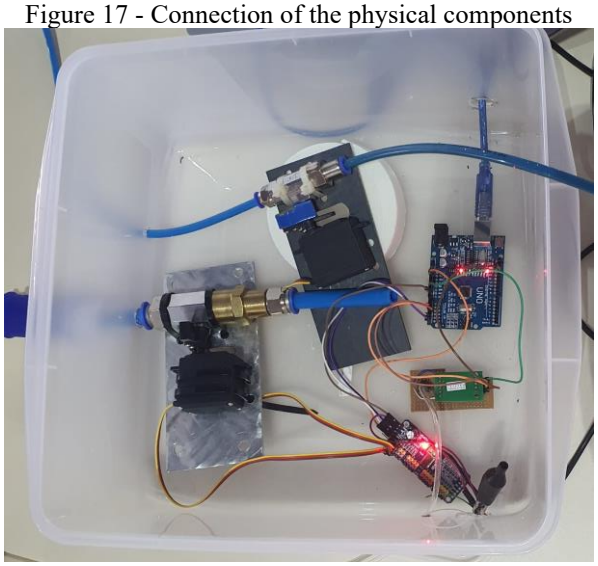
The objective of building a graphical interface for the use of the prototype was successfully completed, as illustrated in Figure 16. The set of buttons, editable, reading and graphic parameters have an uncomplicated interaction dynamic, allowing any veterinary professional to operate the VentiVet, even without any previous knowledge of the technical area that covers the construction of the device. This is due to the layout of the interface components, which aims to be intuitive. Finally, Figure 17 shows the physical assembly of the circuit.

Figure 16 - Interface developed in MATLAB



Source: the authors, 2023.

Figure 17 - Connection of the physical components



Source: The authors, 2023.

5 CONCLUSIONS

The construction of the prototype demonstrates the importance of an interface associated with a life support device such as this one, allowing the change of its operating configurations to suit each case. The combination of the *hardware*, which includes the servo motors, pressure sensor, pneumatic valves and other components, together with the *software*, which is related to the programming of MATLAB, and *firmware*, which corresponds to the code of the Arduino Uno microcontroller, allowed the creation of a mechanical ventilator of intuitive, efficient and, especially, low cost use.

The use of the ball valve stood out as a strategic alternative that not only provided satisfactory control of the air flow, but also contributed to the reduction of the costs of the construction of the prototype. This economic aspect is extremely relevant, considering that VentiVet can be used in environments where financial resources are limited, such as philanthropic veterinary clinics, for example.

In addition, the graphical interface built into MATLAB proved to be a fundamental element for the use of the device, as it considerably facilitates the interaction between the operator and the mechanical ventilator. The simplicity and effectiveness of this interface make the equipment accessible even to veterinary medicine professionals who do not have great experience in the technological area. This expands the potential application of VentiVet in different contexts.

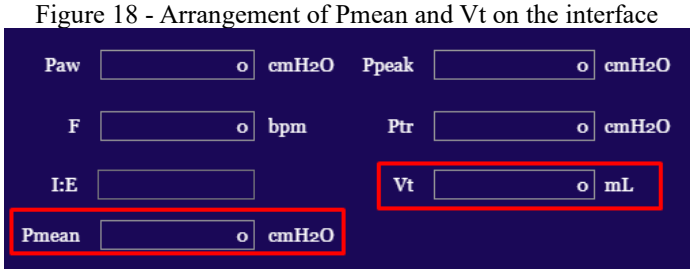
In summary, this work not only brought a new way to produce a veterinary mechanical ventilator, but also highlighted the importance of considering affordability and ease of use and development of technologies aimed at the area of animal health. Despite its satisfactory operation, the equipment can still be improved by the use of:

- Alarms: as mentioned earlier, there are already alarms and security protocols implemented in the prototype. However, these can be improved. For example, with the activation of

audible beacons if the operation or any reading made by the fan is out of compliance. In this way, the operation would be even safer;

- Ventilatory modes: currently, the only ventilatory mode in which VentiVet operates is the controlled one, which does not allow any type of autonomy to the patient. In order to improve the prototype, it is planned to implement the assisted-controlled mode, which covers a wider range of treatments, making it more inclusive;
- Addition of different types of sensors: the pressure sensor currently used cannot read some parameters that need to be implemented, such as Vt and Pmean (which are already arranged in the interface, as illustrated in Figure 18, but without any code associated with their operation). The implementation of these sensors will also allow the inclusion of graphs for flow and volume monitoring. To solve this problem, it is necessary to install a flow sensor in the prototype. Such an association is still pending, because sensors such as these have a high cost for the required reading range.

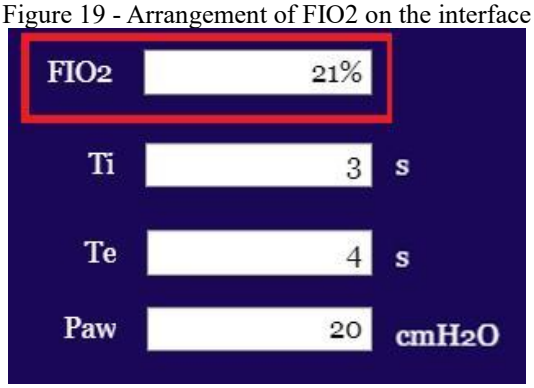
Figure 18 - Arrangement of Pmean and Vt on the interface



Source: The authors, 2023.

- Addition of an oxygen cylinder: for a satisfactory operation of the prototype, it is necessary that an oxygen cylinder be associated with the compressor. In this way, the FIO_2 can be adjusted, according to the patient's condition. The adjustment box for this parameter is also already arranged in the HMI, as shown in Figure 19, but without any associated code. Its marking is constantly at 21%, as it is equivalent to the fraction of oxygen present in the atmospheric air used by the compressor. With the implementation of this change, it will become necessary to add a second proportional inspiratory valve, in order to control the intake of compressed air and oxygen gas separately;

Figure 19 - Arrangement of FIO2 on the interface



Source: The authors, 2023.

- Prototype control via Wi-Fi or Bluetooth: initially, when idealizing the project, it was planned to use an ESP-32 microcontroller board, due to the possibility of Wi-Fi and/or Bluetooth integration, which would prove useful mainly in the display of alarms, even if at a distance. Due to technical complications previously mentioned, and considering that the Arduino board has a much wider range of application examples, it was not possible to bring this idealization to reality. Therefore, for the improvement of the project, this implementation is necessary;
- Reorganization of the *hardware* to become more compact: the box used to maintain the organization of the *hardware* currently has some limitations in terms of saving space used by the prototype. For a more convenient use, it is necessary to reorganize the components, using mostly vertical space, with the replacement of the box originally used.
- Addition of PEEP: PEEP control is essential for the safe use of the mechanical ventilator, considering that its performance is linked to the preservation or recovery of the health of the alveoli. The PEEP will work like any other editable parameter, in which the numerical value entered must correspond to the positive pressure maintained in the lung at the end of expiration.

REFERENCES

ADI, Puput Dani Prasetyo; KITAGAWA, Akio. ZigBee Radio Frequency (RF) Performance on Raspberry Pi 3 for Internet of Things (IoT) based Blood Pressure Sensors Monitoring. International Journal of Advanced Computer Science and Applications, v. 10, n. 5, p. 18-27, 2019.

ARDUINO. Arduino Uno Rev3, 2023. Disponível em: https://store.arduino.cc/products/arduino-uno-rev3. Acesso em: 11 nov. 2023.

CARVALHO, Carlos Roberto Ribeiro de; TOUFEN, Carlos Junior; FRANCA, Suelene Aires. Ventilação mecânica: princípios, análise gráfica e modalidades ventilatórias. Jornal brasileiro de pneumologia, v. 33, p. 54-70, 2007.

CASA DA ROBÓTICA. Módulo PWM PCA9685 Servo Motor Driver 16 Canais I2c, 2023. Disponível em: https://www.casadarobotica.com/sensores-e-modulos/modulos/outros/modulo-pwm-pca9685-servo-motor-driver-16-canais-i2c. Acesso em: 29 nov. 2023.

CASTRO, Fabiano Pimentel de. Servo motores: visão geral, controle e aplicação. 2016. 38 f. Trabalho de Conclusão de Curso (Especialização em Automação Industrial) - Universidade Tecnológica Federal do Paraná, Curitiba, 2016.

CASTRO, Marina Lopes. Princípios básicos da ventilação mecânica em cães. Monografia (Especialista no Curso de Pós-graduação Latu sensu em Residência em Medicina Veterinária) - Universidade Federal de Minas Gerais, 2011.

CAVALCANTE, Andressa Vallery Setubal de Oliveira Nunes. Análise do drive respiratório neural em indivíduos hipertensos durante a ventilação máxima. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte, 2020.

CONSELHO FEDERAL DE MEDICINA VETERINÁRIA. Resolução Nº1015, de 9 de novembro de 2012. [Conceitua e estabelece condições para o funcionamento de estabelecimentos médico-veterinários de atendimento a pequenos animais e dá outras providências.]. Diário Oficial da União: Seção 1, Brasília, DF, ano 150, n. 22, p.172-173, 31 jan. 2013.

CONSULTA DOG VET. Sistema respiratório, 2017. Disponível em: https://consultadogvet.wordpress.com/2017/02/23/sistema-respiratorio/. Acesso em: 7 dez. 2023.

DINAU, Priscilla Caroline Moutinho. Projeto de implementação de controle de posição e velocidade de servomotores DC brushless e proposta de posicionamento dinâmico de um modelo em escala de plataforma semi-submersível.Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação). Universidade Federal do Rio de Janeiro, 2009.

DUGDALE, Alex. The ins and outs of ventilation 1. Basic principles. In Practice, v. 29, n. 4, p. 186-193, 2007.

HOBBY GOIÁS. Kit 4 Servo Mg996r Full Engrenagens Metal 12kg Mg996 Mg995, 2023. Disponível em:https://www.hobbygoias.com.br/MLB-2140585040-kit-4-servo-mg996r-full-engrenagens-metal-12kg-mg996-mg995-_JM. Acesso em: 30 nov. 2023.

HX711. *In*: AVIA Semiconductor. Disponível em: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf. Acesso em: 26 nov. 2023.

MATHWORKS. MATLAB, 2023. Disponível em: https://www.mathworks.com/products/matlab.html. Acesso em: 23 nov. 2023.

MELLEMA, Matthew S. Ventilator waveforms. Topics in companion animal medicine, v. 28, n. 3, p. 112-123, 2013.

MG995 Datasheet (PDF) - List of Unclassifed Manufacturers, Alldatasheet, 2023. Disponível em: https://www.alldatasheet.com/datasheet-pdf/pdf/1132435/ETC2/MG995.html. Acesso em: 29 nov. 2023.

MULTILÓGICA-SHOP, Placa Uno R3 SMD com cabo USB. Disponível em: https://multilogica-shop.com/produtos/placa-uno-r3-smd-arduino-compativel/. Acesso em: 29 nov. 2023.

ROMERO, Jorge Calderón. Confiabilidade Metrológica de Ventiladores Pulmonares. 2006. Tese de Doutorado. Dissertação (Mestrado em Metrologia para Qualidade e Inovação), Pontifícia Universidade Católica–PUC, Rio de Janeiro, 2006.

SILVA, Heuler Andrade. et al. DESENVOLVIMENTO DE PROTÓTIPO PARA CONTROLE DE TEMPERATURA PARA AQUECEDORES HÍDRICOS. *In*: IX CONGRESSO DE ENGENHARIAS DA UNIVERSIDADE FEDERAL DE SãO JOãO DEL-REI, 2019, São João del-Rei. Anais eletrônicos - Campinas, Galoá, 2019. Disponível em: https://proceedings.science/coen-2019/trabalhos/desenvolvimento-de-prototipo-para-controle-de-temperatura-para-aquecedores-hidri?lang=pt-br. Acesso em: 23 nov. 2023.

SILVA, Rogério Oliveira da; ARAUJO, Warley Monteiro; CAVALCANTE, Maxwell Machado. Visão geral sobre microcontroladores e prototipagem com arduino. Tecnologias em Projeção, v. 10, n. 1, p. 36-46, 2019.

UTFPR. VENT-U² Ventilador Pulmonar concebido na Universidade para você, 2020. Disponível em: https://acao.utfpr.edu.br/acao/13/. Acesso em: 7 dez. 2023.