MODEL-GUIDED EXTREMUM SEEKING CONTROL FOR MECHANICAL VENTILATORS

Ryan Keith Wardell
University of Windsor

MODEL-GUIDED EXTREMUM SEEKING CONTROL FOR MECHANICAL VENTILATORS

by

Ryan K. Wardell

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Mechanical, Automotive & Materials Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2023

MODEL-GUIDED EXTREMUM SEEKING CONTROL FOR MECHANICAL VENTILATORS

by

Ryan K. Wardell

APPROVED BY:

X. Chen
Department of Electrical & Computer Engineering

J. Ahamed
Department of Mechanical, Automotive & Materials Engineering

J. Defoe, Advisor
Department of Mechanical, Automotive & Materials Engineering

May 19, 2023

Declaration of Co-Authorship / Previous Publication

I Co-Authorship

I hereby declare that this thesis incorporates material that is the result of joint research, as follows: The thesis was authored by Ryan K. Wardell under the supervision of professor Dr. J. Defoe. In Chapter 2, Palak Saini designed the first version of the experimental setup which was further refined and reassembled by myself. In all other cases, the key ideas, primary contributions, numerical designs, data analysis, interpretation, and writing were performed by the author; Dr. J. Defoe provided feedback on refinement of ideas and editing of the manuscript.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II Previous Publication

This thesis includes 2 original papers that are to be published/submitted for publication in peer reviewed journals, as follows:

Thesis Chapter	Publication title/full citation	Publication status
Chapter 2	Wardell R.; Saini, P.; Defoe, J. Model-guided Extremum Seeking Control for Mechanical Ventilators Part 1: Measured Performance	To be submitted to the Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME
Chapter 3	Wardell R.; Defoe, J. Model-guided Extremum Seeking Control for Mechanical Ventilators Part 2: Numerical Assessment of Impacts of Geometry Changes on Performance	To be submitted to the Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Mechanical ventilators are an important lifesaving medical device and because of the COVID-19 pandemic, they were in a massive demand for them to help the patients most severely affected by the disease. A large number of open-source hardware ventilator projects were established during the pandemic, most of which did not provide sufficient information on the design of their control systems. In this thesis, a model-guided extremum seeking control scheme is proposed for use on a open-source hardware ventilator design. The performance of the control scheme is intended to not be significantly affected by changes to the patient, nor to variations in the ventilators geometry. To test the robustness of the model-guided extremum seeking control scheme to patient variation, a prototype ventilator was built, using the hardware design from one of the open-source hardware ventilator projects, and experiments were conducted using a test lung. In this process it was determined that the control scheme was able to achieve its goal of converging unto a targeted volumetric flow rate during inspiration for all volume-controlled ventilation tests performed. To test the robustness of the model-guided extremum seeking control scheme to variations in the ventilator design, a numerical model was created that replicates the experimental device. The numerical model was able to reproduce the experimental results with a 16% difference for the tidal volumes and a 14% difference for the positive end expiratory pressures. Various ventilator parameters were investigated, and it was found that the only geometric parameters that had a significant effect on the ventilators performance were the diameter of the main ventilator piping and the length of the expiratory piping. Through all the tests performed via the numerical model, the model-guided extremum seeking control scheme was always able to converge upon the target volumetric flow rate during inspiration.

Acknowledgments

I am grateful to have worked with an incredibly passionate and supportive supervisor, Dr. Jeff Defoe. His work knowledge, and passion towards engineering is what influenced me to pursue education past the undergraduate level. Additionally, I would like to thank my committee members Dr. Jalal Ahamed and Dr. Xiang Chen for both their time in reviewing my thesis, and specifically Dr. Chen for his comments and suggestions made during my thesis proposal that significantly improved the quality of my work and education.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

A Area

Q Volumetric flow rate

D Diameter

L Length

V Voltage

Vol Volume

p Pressure

b y-intercept

m slope

s Spring stiffness coefficient

c Dash-pot damper coefficient

θ Control input

k Integration gain

x Lung displacement

σ Standard deviation

ω Frequency

ϕ Phase angle

τ Time constant

ϵ Relative roughness

Subscripts

out	Outputted measurement
meas	Measured
min	Minimum
max	Maximum
sup	Supply
i	Interface
insp	Inspiration
exp	Expiration
tidal	Tidal volume
airway	At the patient's mask
demod	Demodulation
mod	Modulation
s	Sampled
branch	Inspiratory branch section
tubing	Tubing from solenoid valve to the normally-open expiratory control valve

Superscripts

$*$	Value at extremum point in objective function

Abbreviations

PID	Proportional Integral Derivative
ISO	International Organization for Standardization
MVM	Mechanical Ventilator Milano
PEEP	Positive end expiratory pressure
OVSI	Oxygen and Ventilator System Initiative
OSH	Open-source Hardware
VCV	Volume controlled ventilation
PCV	Pressure controlled ventilation
BAP	Baseline Airway Pressure
BPM	Breaths per minute
ESC	Extremum seeking control
MGESC	Model-guided extremum seeking control
MFESC	Model-free Extremum seeking control
OL	Open-loop
PV	Proportional valve
NPR	Negative pressure relief valve
APLV	Airway pressure limiting valve
SV	Three-way solenoid valve
NOV	Normally-open Expiratory Valve
EPRV	Expiratory pressure regulating valve
FSS	Full scale span
FS	Full scale
DC	Direct current
PWM	pulse-width modulation
RP4	Raspberry Pi 4 8GB B+
DAC	Digital-to-analog converter

ADC Analog-to-digital converter

LPF Low-pass filter

HPF High-pass filter

FiO2 Fraction of inspiratory Oxygen

RMS Root-mean-squared

Chapter 1

INTRODUCTION

Because of the COVID-19 pandemic, there was a global shortage of hospital grade ventilators. Academia and industry came together during this time to develop and support new, low-cost, open-source hardware ventilator designs. By investigating many such projects, it was determined that most did not provide detail on the ventilator's control design, and the ones that did used proportional/integral/differential (PID) based controllers. It became clear that developing an adaptive control system that would be impervious to both uncertainty in the hardware design of the ventilator, and variations in the patient's condition would be desirable. The use of an extremum seeking based controller (ESC) was explored as a model-free control approach; however, from reviewing the literature on ESC it was found that the introduction of a simple model to provide the ESC with an initial guess for the control input could significantly improve the controller's convergence time [1]. With a mechanical ventilator being an important life saving medical device it was clear that faster convergence times was a desirable feature. So, a model-guided ESC (MGESC) scheme was to be created with the overall goal of no significant loss of performance when applied to different ventilator designs and patients.

1.1 Objective and High-Level Approach

The objective of this thesis is to create a MGESC that can adapt and perform on various open-source ventilator designs and under various adult patient lung conditions. To do so, a completely open-source mechanical ventilator was built based on an open-hardware design from Mechanical Ventilator Milano (MVM) [2], and a dual adult test lung was purchased from Michigan Instruments [3] to test the MGESC against ISO tests meant for emergency ventilators operating in volume controlled ventilation modes [4]. The performance of the ventilator was measured by comparing the ventilator's measurements (of each breath's tidal volume and the positive end expiratory pressure (PEEP)) to the test lung's measurements as well as their target value. To test the MGESC on various ventilator geometries and designs, a numerical model of the entire ventilator experimental setup was created in Siemens' Simcenter Amesim [5]. The accuracy of the model was determined by comparing the ISO test results performed experimentally to the numerical results achieved in Amesim. Various geometric and non-geometric parameters of the ventilator were changed in the model, and the change in performance of the MGESC was assessed by comparing the tidal volume and PEEP values against their target for the ISO test used.

1.2 Major Findings and Conclusions

In Chapter 2, it was found that after removing outliers, the tidal volume measured within 21% of the test lung's measurements and the PEEP measured within 13% of the test lung's measurements. The introduction of a model to provide the ESC with an initial control input reduced convergence time by a minimum of 29 breaths, and this result was achieved even though it was determined that the model used consistently under predicts the control input needed to achieve the target volumetric flow rate. The settling time for the flow rate during inspiration was unaffected by the type of control used, and the settling time was a function of the control input. Most importantly, the MGESC was able to converge upon the specified target inspiratory volumetric flow rate for all tests; solidifying that the control scheme is able to perform across the range of parameters defined by the ISO standard.

In Chapter 3, it was determined that an increase in error between the numerical and experimental tidal volume calculations occurred when the lung's resistance setting is increased and when the target tidal volume is decreased. A part of the reason for this was because the Amesim model does not accurately capture the more rapid rise time associated with the lower flow rate target because the modelling of the main inflow-controlling proportional valve is kept constant, although its behavior suggests that it is a function of the target volumetric flow rate. From Chapter 2, it is known that the experimental expiratory pressure release valve that controls the baseline airway pressure could only go as low as 5.4 hPa; but, the valve in the numerical model follows the ISO test BAP setting of 5 hPa, resulting in the error in PEEP calculations between the experimental and numerical results being inflated for tests with a BAP of 5 hPa. Discounting cases with these sources of error, the numerical model was able to achieve tidal volumes within 16% and PEEP values within 14% of the experimental results. The ISO tests with the best performance was used for testing variations of the ventilator parameters. Of the geometric parameters investigated, only two were found to have a significant effect on the ventilator's performance: the main diameter of piping, and the length of the expiratory piping section. The MGESC was able to adapt to all of the ranges of parameters tested and was always able to converge upon the target flow rate. However, undesirable fluctuations in flow rate occurred when the length of the expiratory section was less than 50 cm. Similar to the results in Chapter 2, it was determined that the model that provides the initial control input guess to the ESC is not sensitive enough to changes in the supply pressure.

1.3 Thesis Outline

This thesis is split into two major chapters. In Chapter 2, the MGESC is introduced and tested on the ventilator design that is fully described within the paper. The MGESC is used with the open-source ventilator design and tested with the test lung across a range of varying settings listed in the relevant ISO standard. In Chapter 3, the MGESC scheme and the ventilator design from Chapter 2 is modelled in numerical simulation software in order to test the control system against various ventilator parameters. The complete numerical model and how it was created is described and its results are compared against the experimental results in Chapter 2 to determine its accuracy. Once that is completed various parameters of the model are changed within a logically chosen range and the control scheme's performance is assessed. Lastly in Chapter 4, conclusions are drawn from this work and recommendations for improving both the ventilators accuracy and the model's accuracy are given.

1.4 Bibliography

[1] Ying Tan, Xiang Chen, Youying Hua, and Qingyuan Tan. Model-guided extremum seeking case studies. *International Journal of Adaptive Control and Signal Processing*, 36(3):708–728, December 2021.

[2] A. Abba et al. The novel mechanical ventilator milano for the COVID-19 pandemic. *Physics of Fluids*, 33(3):037122, mar 2021.

[3] *Training & Test Lung Training & Test Lung Operation Manual*. Michigan Instruments, January 2021.

[4] Iso 80601-2-12:2020. *International Organization for Standardization*, 2021.

[5] Siemens. *Siemens Simcenter Amesim 2020.2*, 2020.

Chapter 2

MODEL-GUIDED EXTREMUM SEEKING CONTROL FOR MECHANICAL VENTILATORS PART 1: MEASURED PERFORMANCE

2.1 Introduction

People ill with respiratory diseases such as SARS-COV-2 (COVID-19) sometimes require a ventilator [1]. The early days of the COVID-19 pandemic led to concerns around a shortage of ventilators [2]. Due to the perceived need for very large numbers of ventilators, many government and industry-backed projects were initiated with the aim of creating an “open-source” design for a mechanical ventilator at lower cost than typical ventilators, and which could be produced in large quantities at need. New ventilators can cost up to \$50,000 USD [3], and production of these machines typically had lead times of around 8 weeks pre-pandemic and during the pandemic the lead times increased to several months [4]. Good examples of open-source ventilator projects are the Mechanical Ventilator Milano (MVM) [5] and Cambridge OVSI [6].

A common aspect of these projects is that rather than being truly "open-source" in the sense that all details are provided to enable others to reproduce the designs, they are only open-source hardware (OSH) designs, excluding major design details necessary to create a fully-operational ventilator. Santos, Zacharia, and Cota stated that these projects' lack of adherence to best practices and lack of development towards certification could reduce the viability of OSH ventilators as a tool in overcoming pandemic-related healthcare challenges [7]. In their paper, they review the major OSH ventilator projects and assess each project's real-life viability by comparing the milestones the projects have achieved on their way to actually being used in a medical environment. One of the better-performing ventilator projects listed in this paper was the MVM. MVM's design is simple, low-cost, and consists of almost all off-the-shelf components. However, its downfall comes in the design of the control system and software. In their paper, little to no information regarding the control system design is included, and no information regarding the software design architecture. This trend is consistent with similar projects. Stanford University's O2U ventilator's entire control system is described simply as a "closed feedback loop" [8]. Imperial College London's ventilator design had no information on their control system [9], and the University of Cambridge's OVSI also includes no information on their control system design [6]. Projects that include information on the design of their control system generally use a proportional/integral/derivative (PID) controller [10, 5, 11].

When the system being controlled is a ventilator and patient, large model uncertainties can exist due to the large patient-to-patient variation that can occur; this makes tuning a PID controller challenging in this application. For healthy adult lungs, the compliance can range from 0.01 to 0.1 L/hPa, and the resistance can range from 5 to 50 hPa/L/s [12]. This motivates consideration of more robust control schemes such as adaptive control and H-infinity. Adaptive control system parameters are adjusted automatically to compensate for changing conditions [13], and have been found to yield shorter settling times than PID when the parameters of the system being controlled vary significantly from the values assumed in the PID design[14]. Like PID controllers, H-infinity also need a model of the system to function and their control parameters are constant, but they have increased robustness due to being designed for a worst-case operating point [15].

A control approach which does not require a model may be advantageous for a ventilator due to the large variations in patients; a popular model-free adaptive control method is extremum-seeking-control (ESC) [16]. Using a model-free adaptive control approach such as ESC not only helps to address patient-to-patient variation, but may enable a single controller to be implemented for variations of a given ventilator design. A key challenge to address with applying ESC for a ventilator application is the convergence time, since an inherent drawback of the model-free nature of this data-driven approach is a slow convergence rate [17]. To a great extent, this can be addressed by using model-guided ESC (MGESC), in which a system model, which can be approximate and need not derive from first principles, is integrated into an ESC [17]. In this paper, we apply MGESC to the OSH design from the MVM project [5].

There are two main categories of ventilation modes: pressure-controlled ventilation (PCV) and volume-controlled ventilation (VCV); in this paper we consider only VCV. In VCV, the inspiratory flow rate is regulated, and the expiratory flow rate is a consequence of the pressure difference between the lung and the baseline airway pressure (BAP) setting, which regulates the minimum pressure during the expiratory phase.

The key findings are that the ventilator's measurements were comparable to the test lung's measurements to within 21% for the tidal volume calculations and 10% for the positive end expiratory pressure calculations for the five out of the seven tests performed. The other two tests had too large of difference to continue with for the remainder of the results. The model used to find the initial control input for the ESC consistently under predicts, but still leads to an increase in convergence time of 29 breaths or more over the model-free ESC scheme. The settling time of the flow rate during inspiration is a function of the control input and is unaffected by the control scheme. The flow rate does not settle to its target value until the end of inspiration, thus the tidal volumes are consistently under their target. To help increase the tidal volume, and thus the average flow rate during inspiration, it was found that increasing the supply pressure would do so. Overall, the proposed MGESC scheme is able to converge upon its targeted volumetric flow rate during inspiration for all VCV tests performed.

The remainder of this paper is organized as follows. Section 2.2 introduces the high-level hardware design of the ventilator and then goes into detail on each actuating, sensing, and computing component. Section 2.3 introduces the ESC structure along with the model used for the final MGESC scheme. Section 2.4 explains the software design of the ventilator design and the programs used to complete the experiments. Section 2.5 explains how the experiments will be conducted and the tools used to test the ventilator. Section 2.6 explains how the uncertainty of each measurement is calculated following the propagation of uncertainty principle. Finally in Section 2.7 the results of the experiment are shown and discussed leading into Section 2.8 where the important conclusions and take-aways from this study are listed.

2.2 MVM Hardware Implementation

Using the MVM hardware design layout [5], similar commercially available hardware was purchased and a prototype ventilator was fabricated. This section provides a brief overview of the hardware implementation, focusing on aspects in which the design differs from the MVM's OSH design. Figure 2-1 is a schematic of the hardware layout that illustrates the gas pathways, sensors, and actuators. Arrows indicate the direction of flow. The key components in Figure 2-1 and their functions are explained briefly in Table 2-1. A detailed diagram showing each ventilator hardware component, with assembly instructions, and a bill of materials are given in Appendix A.

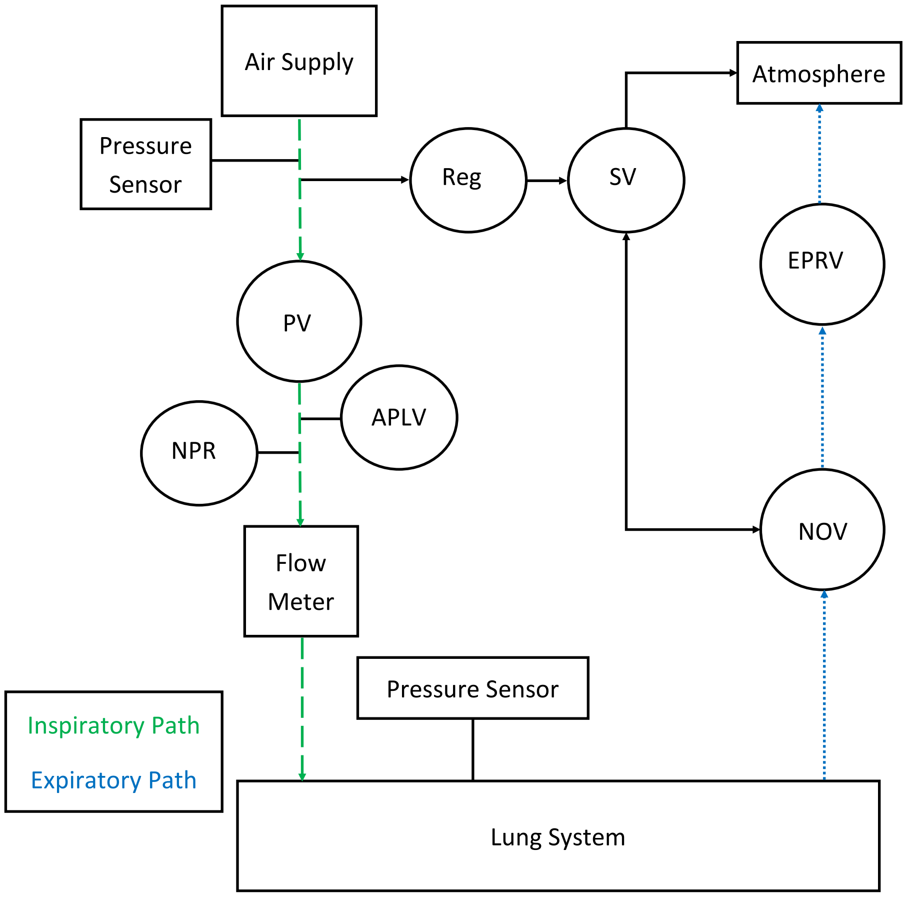

Figure 2-1: Overview of the current implementation of the MVM

Table 2-1: Key ventilator components

Component	Description
Proportional valve (PV)	Controls airflow through the inspiratory path
Negative pressure relief valve (NPR)	Allows for the patient to take spontaneous breathes
Airway pressure limiting valve (APLV)	Safety measure to limit pressure in the inspiratory path
Three-way solenoid valve (SV)	Control valve responsible for actuating NOV
Normally-open valve (NOV)	Controls airflow though the expiratory path
Expiratory pressure regulating valve (EPRV)	Controls the minimum pressure in the expiratory path to the BAP setting

The mechanical functionality of our ventilator is almost identical to that of the MVM [5], so we focus next on differences between our hardware implementation and that described by Galbiati, Abba and Zardoni. Our ventilator is a prototype and is not suitable for medical use, though the intention is for the control scheme to be applicable to medical-grade products based on the MVM design. Our prototype ventilator lacks a GUI and the control system does not include any alarms or system warnings which need to be in place as described in the ISO standard [18]. A piece of hardware that was left out which is needed for a medical-grade ventilator is a condensation trap in the expiratory path to remove moisture from a real patient's expiration. Additionally, a pressure sensor upstream of PV was included since the valve response is a function of supply pressure, and this is used in the model of the system discussed in Section 2.3.1.

2.2.1 Sensors

Our design uses three sensors: the flow meter, the supply pressure transducer, and the lung airway pressure transducer.

The flow meter is the SFM3300-AW from Sensirion, and is capable of measuring bi-directional flow from -250 to 250 standard liters per minute (sL/min) [19]. The measurement uncertainties for the flow meter, listed in it's specification manual [19], can be found in Table 2-2.

Table 2-2: Flow meter uncertainties

Flow Rate Q (sL/min)	Resolution (% Measured Value)	Accuracy (% Measured Value)	Noise (% Measured Value)
$Q \leq 25$	0.07	4	2.25
$100 > Q > 25$	0.07	4	4
$Q > 100$	0.07	8.5	4

The supply pressure transducer is a Honeywell SSCDANN100PAAA5, and the lung airway pressure transducer is a Honeywell SSCDRRN001PDAA5 [20]. Both pressure transducers require a 5.00 ± 0.25 V direct current (DC) supply voltage, and have a minimum output resolution of 0.03% of the full scale span of the pressure range (%FSS) [20]. The accuracy of both transducers are the same and is listed as a maximum of 0.25 %FSS, which includes uncertainty associated with non-linearity, hysteresis, and repeatability [20]. The output voltage signals are [20]

$$V_{out} = \left(0.8\frac{p_{meas} - p_{min}}{p_{max} - p_{min}} + 0.10\right) V_{sup} \tag{2.1}$$

where p_{min} and p_{max} are the minimum and maximum pressures the transducers can measure, respectively, p_{meas} is the pressure being measured, and V_{sup} is the supply voltage. The supply pressure transducer has a range of 0 to 689 kPa absolute. The airway inlet pressure transducer has a range of -6.89 to 6.89 kPa gauge.

2.2.2 Actuators

There are three actuating valves in the ventilator's design: the proportional valve (PV), the normally open valve (NOV), and the 3-way solenoid valve (SV) that actuates the NOV. The PV is the AP-7211-QW2-U711-OX2 made by Camozzi, which has an orifice diameter of 2.4 mm, can operate with a maximum supply pressure of 4 bar gauge, and has a maximum flow capacity of 113 normal liters per minute (nL/min) [21]. The PV is actuated by a 24 V pulse-width modulation (PWM) signal. The hysteresis for the PV is 10% of the full scale of the control input (%FS), and the repeatability is 7% FS [21]. The NOV is a 3-way directional control spool valve, 6124K401 from McMaster-Carr, with a minimum actuation pressure of 200 kPa, and that can be either normally open or normally closed based on how it is connected [22]. Thus, the regulator depicted in Figure 2-1 is set to 241.317 kPa to allow for proper actuation of the valve. The solenoid valve used is a compact threaded 3-way solenoid diverting valve, 2565N15 from McMaster-Carr, that is actuated by a 0 or 24 V DC signal [23]. Both the PV and SV are directly actuated by the control system hardware, whereas the NOV is actuated by pressurization/depressurization of its supply line, controlled by the SV.

2.2.3 Power and Signal Management

The main power source for the ventilator is a 24 V, 5 A power supply, 1470-3098-ND purchased from Digikey, that was branched off to power all the electronic components of the ventilator, apart from the Raspberry Pi. A voltage step-down board, LM2596 purchased from Amazon, is used to bring the voltage down to the recommended voltage for the pressure transducers. The main computer in the ventilator is a Raspberry Pi 4 8GB B+ (RP4) which requires a power source at 5 V with a minimum available current of 3 A [24], which exceeds what the step-down board could supply, so a separate 5 V 3 A USB type-C charger is used. The power supply connectivity is illustrated in Figure 2-2.

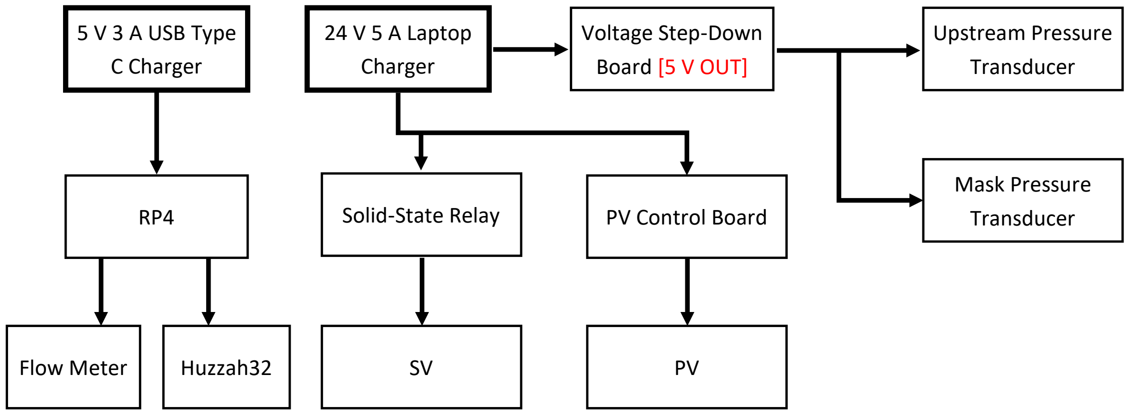

Figure 2-2: Power distribution schematic

As the RP4 has only digital outputs, a micro-controller, the Adafruit Huzzah32 [25], is included due to its ability to read and send analog voltage signals, critical for actuating the PV and for reading the pressure transducers outputs. The Huzzah32 is powered by the RP4 via USB, and the two devices also communicate via the same connection. A complete wiring diagram is located in Appendix A.

The RP4 uses digital 3.3V logic [24], but the SV requires a 25 V DC signal. We employ a solid-state relay (DC60S3 from Crydom) in connection with the 24 V power supply . The relay wiring diagram is included in Appendix A.

The PV is powered and controlled by the electronic control for proportional valve board 130-222 made by Camozzi [26]. The control board requires a 4 to 20 mA analog current input that corresponds to 0 to 100% FS for the PV, and the board has an internal resistance of 185.3 Ω [26]. The analog current input for the control board is supplied from the Huzzah32. Digital-to-analog (DAC) channel 25 is used to send the control signal to the PV's control board which has an 8-bit digital range; the corresponding analog output is 0-3.3 V DC which gives a resolution for the control signal of 0.013 V. As a result, the maximum analog current signal is 18 mA or 87.5% FS. From the proportional valve's characteristic in Figure 2-4 (discussed in detail later) it is shown that a signal of 87.5% FS would result in a flow rate of approximately 100 L/min, at nominal supply pressure, which is significantly higher than 30 L/min, which is the maximum flow rate being tested; thus, this control method for the PV is sufficient.

The ADC pins read 0 to 3.3V analog signals and converts them to 12 bit digital readings, giving a measurement resolution of 0.00081 V. To avoid over-voltage to the ADC pins since they are reading from 5 V systems, voltage dividers consisting of two resistors arranged in series so that an input voltage is divided by some known value that is a function of the two resistances. There is an uncertainty of 6% of the measured voltage for the ADC pins on the Huzzah32 (below 3 V) [25]. In Table 2-3 a description of the three measurements read onto the Huzzah32 ADC pins is given.

Table 2-3: Huzzah32 ADC pin descriptions

Pin No.	Measurement	Measurement range	Voltage divider	Pin voltage range
32	Supply pressure transducer	419 - 501 kPa	3/4	2.09 - 2.68 V
12	Lung airway pressure transducer	0 - 2.942 kPag	3/4	1.78 - 2.64 V
33	Pressure transducer power supply	4.75 - 5.25 V	3/5	2.85 - 3.15 V

The pin voltage range in the table are the voltage that is read onto the Huzzah32. The minimum value of the pressure transducer ranges are calculated using Equation 2.1 using the minimum possible pressure transducer supply voltage of 4.75 V, then the maximum value of the range is calculated using the maximum possible supply voltage of 5.25 V. As you can see from the pin voltage range column, the values are all under the maximum possible readable voltage of 3.3V, and both pressure transducer voltage readings are under 3 V giving the least uncertainty associated with the ADC pins.

2.3 Model-Free and Model-Guided ESC

In its original formulation, ESC is a model-free, real-time adaptive control algorithm that is most effective when the system dynamics are unknown [27]. While most methods for adaptive control aim to drive the output process to a known set point, the goal of ESC is to find a system input that maps to an extremum, either a minimum or maximum [28]. Using the inputs and/or outputs of the model, an objective function can be created and this is what the ESC seeks to regulate to an extremum. The basic concept of how ESC does this is can be simplified into 4 stages: modulation, system response, demodulation, and state regulation. In the first stage, the system is perturbed by a low-amplitude periodic input signal. Next the system response yields a new objective function value. In demodulation, the objective function is multiplied by the same periodic function but with a different, typically larger, amplitude. Finally, the demodulated signal passes through a low-pass filter (LPF) to remove high-frequency noise, and goes through a state regulator, commonly an integrator, that by using the demodulated signal, can iteratively seek an extremum point [28, 27]. Much more detail of the functioning of ESC can be found in the literature, e.g. refs [29, 27]. In this paper, we employ a simple discrete ESC scheme with a sinusoidal perturbation signal, also referred to as a dither signal, and include a low-pass filter (LPF), but not a high-pass filter (HPF) as can be found in some implementations. Figure 2-3 illustrates the basic ESC scheme that has been implemented, where the system has output y and input u; t is time. The design parameters for this control scheme are the perturbation signal amplitudes (a_{demod} and a_{mod}), the perturbation signal frequency (ω), the perturbation signal phase angle (φ), the LPF cut-off frequency (f_{cutoff}), and the integration gain (k).

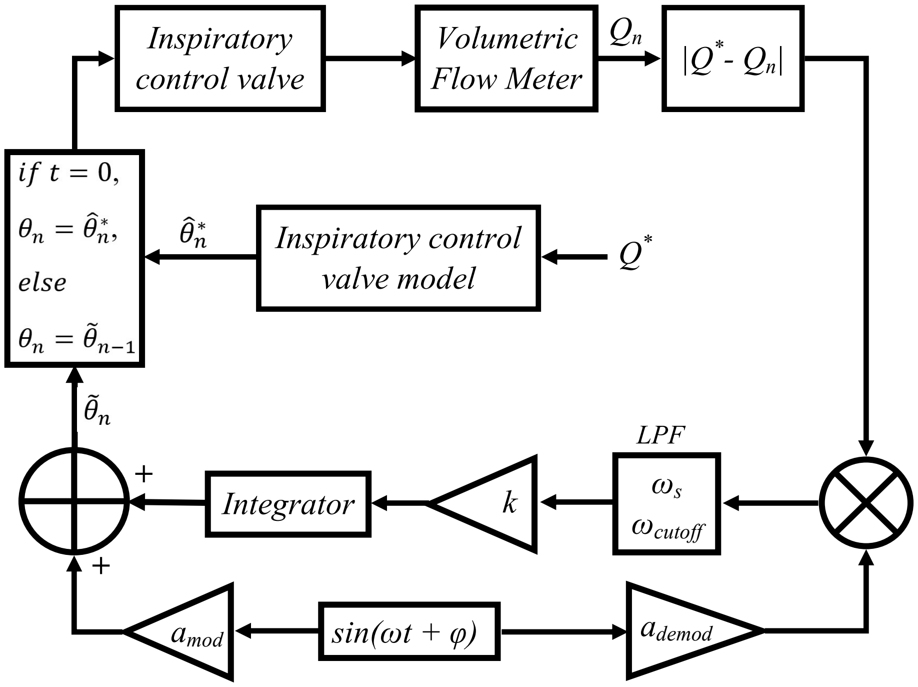

Figure 2-3: Schematic of ESC implementation

Krstić and Wang [29] indicate that the cut-off frequency of the filter needs to be lower than the frequency of the perturbation signal, the integration gain must be low, and the amplitude of the periodic perturbation must be small for better convergence and stability. Ariyur and Krsticacute [27] give guidelines indicating that the amplitude of the demodulation signal must be much greater than the amplitude of the modulation signal, and that the phase angles must satisfy the criteria of $\cos(\varphi_{demod} - \varphi_{mod}) > 0$. We found a HPF not to be necessary for adequate performance. The design parameters allow for a trade between convergence time, stability, and transient behavior to be made while choosing the values [28]. The use of a model to provide an initial guess ($\hat{\theta}^*$) for the ESC scheme is to ensure the objective function is initially not far from its extremum (θ^*) to decrease convergence times enough so that the base ESC parameters can focus on stability.

2.3.1 System Model

With supply pressures in the acceptable range of 419 kPa to 501 kPa, the flow will always be choked at the PV. Downstream of the PV the flow Mach numbers are much less than unity, so the flow downstream of the PV can be considered incompressible. Thus, the system model for MGESC is solely based on the characteristics of the PV. The characteristics of the PV are given in its specifications [21], from which Figure 2-4 is adapted.

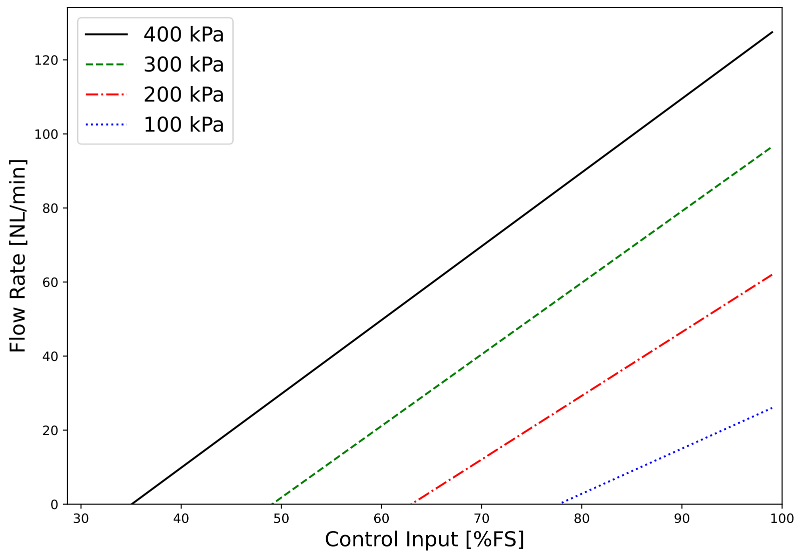

Figure 2-4: Proportional valve characteristic

From the four lines, a relationship was created to estimate the slope (m) and y-intercept (b) of the valve's characteristic at a specific supply pressure. To solve for the slope given the gauge supply pressure in units of bar, a 2nd order polynomial function, $m = -0.1105p_{sup}^2 + 0.8058p_{sup} + 0.5309$, was used with a correlation coefficient of $R^2 = 0.9975$. A linear relation was used to calculate the y-intercept given the gauge supply pressure in units of bar, $b = 33.896p_{sup} - 6.7395$, which has a correlation coefficient of $R^2 = 0.9988$. Using these two equations, and knowing the upstream supply pressure, the valve's behavior can be estimated for any gauge supply pressure between 1 to 4 bar. The purpose of this model is to give an initial guess of the PV %FS control input value ($\hat{\theta}^*$) required for the target flow rate (Q^*) based on the measured supply pressure which is shown in Equation 2.2.

$$\hat{\theta}^* = \frac{Q^* - 33.896p_{sup} + 6.7395}{-0.1105p_{sup}^2 + 0.8058p_{sup} + 0.5309} + 12 \tag{2.2}$$

The additional constant value of 12%FS added to the prediction was found to be necessary to match measured to predicted flow rates in steady-state tests with our valve; it is possible this value would need to be calibrated for each individual valve unit.

2.3.2 Our ESC Parameters

The objective function for the chosen ESC scheme is simply the absolute value of the difference between the target flow rate and the current flow rate. This objective function results in a peak at which the control input would result in the target flow rate being measured. The ESC parameters are selected according to the guidelines laid out in refs. [27, 29]. The selected parameters are: $a_{mod} = 0.0001$, $a_{demod} = 0.01$, $k = 1$, $\varphi = 0$, and $\omega_{cutoff} = 0.4\pi$ rad/s. The only parameter that does not stay constant throughout testing is the perturbation signal frequency ω, which is chosen as a function of the breaths per minute (BPM) value set:

$$\omega = \begin{cases} 2BPM & BPM \leq 14 \\ BPM & BPM > 14 \end{cases}.$$

Using these MGESC parameters resulted in the best performance for the control scheme.

2.4 Software

The RP4 uses the Raspbian GNU/Linux 10 (Buster) operating system, and the main ventilator program is written in Python 3.7.3. The python packages used are listed in Appendix A.3, with their versions and use cases noted. The programming for the Huzzah32 was performed on the RP4 using the Arduino IDE software version 1.8.15. Communication between the RP4 and the Huzzah32 is via serial communication through the USB cable connected to both devices with a baud rate of 9600, 8 bit byte size, no parity, 1 stop bit, and a timeout of 1 second. A key implementation detail of note is that the Huzzah32 could read in its inputs much faster than the Python code on the RP4 could execute, causing the measurement buffer to fill. This would introduce a time lag while reading in these values to the RP4. This problem is solved by clearing the measurement buffer in the Python script on the RP4 at initialization and at the beginning of each inspiratory cycle. Additionally, the data transferred to the RP4 is sent with start and stop bits to ensure no data is lost or corrupted in the communication.

The flow meter is powered by the RP4, and communicates with it through a USB connection as introduced in Section 2.2.3. A software program was provided by the manufacturer to read the flow measurements for a flow meter running a similar hardware chip [30], such that the code could be used for our flow meter with minor updates to the scripts provided. The full Python script and Huzzah32 code are available in Appendix A.3. Appendix A.3 also includes code for an additional Python script that acts as an emergency shutoff program that closes the PV in case the main script crashes.

A modified version of the Python script is used to perform open-loop and model-free tests; these are commented out, but included, in the code in Appendix A.3. For the open-loop test cases the control input is set by using the model's initial guess and then remains at that value for the entirety of each inspiratory cycle. For the model-free test cases, the initial control input guess is given as a constant value that is just large enough that the valve would open and there would be non-zero flow through the PV. Using the valve's characteristic, at a supply pressure of 446 kPa, a value of 60% FS was found to work as the initial guess for the model-free control scheme.

Due to the processing time of sending and receiving measurements, as well as code execution, the fastest time step that could be used for the inspiratory cycle is 0.025 seconds. The time step for the expiratory cycle was kept as the same as the inspiratory cycle for simplicity in post-processing.

2.5 Testing Requirements

International Organization for Standardization (ISO) Standards dictate ventilator testing requirements. Test lung machines that mimic human lungs and trachea are normally used for ventilator testing.

2.5.1 Test Lung

The test lung used for the work reported on in this paper is the Michigan Instruments Dual Adult Lung [31]. The device has compliance and resistance capabilities suitable for the range of adult patients. The test lung's compliance can be varied continuously from 10.2 to 102 mL/hPa, and the resistance values available are 5.1, 20.4, and 51 hPa/L/s. The test lung can be connected through USB to a device running a Windows operating system and the Michigan Instruments PneuView 3 software. The software allows for the live readings of lung information and ventilator settings. A beta version of the software, PneuView 3.4, is used in this paper, as it can record live data at a rate of 167 data points per second and export this data to a comma-separated values (.csv) file. The uncertainty and resolution of the lung measurements made with this system are listed in Table 2-4 [12].

Table 2-4: Test lung measurement uncertainties

Value	Resolution	Uncertainty
Flow Rate [L/min]	0.1	±2%
Volume [mL]	0.1	±3%
Pressure [hPa]	0.1	±5%

2.5.2 ISO Standard

The standard used is ISO 80601-2-12:2020(en) which is for Medical electrical equipment. Relevant for ventilators is Part 2-12 which states the particular requirements for basic safety and essential performance of critical care ventilators. Additionally, since we are only testing out ventilator in a VCV mode only section 201.12.1.101 of the ISO standard is used in the creation of our testing guidelines since it is for volume-control inflation-type ventilators. Moving forward, when it is mentioned that we are following the ISO standard, it is specifically referring to ISO 80601-2-12:2020(en), part 2-12, section 201.12.1.101. The ISO standard states that the results displayed should at least include (1) the maximum error of the inspiratory volume in relation to the set tidal volume; (2) the maximum error of the PEEP in relation to the set value of BAP; (3) the maximum error of the inspiratory oxygen (O2) concentration at the patient-connection port in relation to the set value; and (4) the disclosed accuracy shall include the effects of the range of the rated input oxygen concentration.

As discussed in 2.2, our prototype ventilator does not include a gas blender or oxygen sensor so the FiO2 cannot be measured or controlled, thus the third and fourth criteria are omitted from the results. Table 2-5 gives the seven tests included in the ISO standard that are able to be performed given the test lung's compliance and resistance ranges. The standard states that the results can be split into categories based on the ranges of intended tidal volume with the categories being: equal or greater than 300 ml, less than or equal to 50 ml, or between or equal to 300 and 50 ml. Thus, all of the results are able to be grouped together as they are all 300 mL or greater.

Table 2-5: VCV ISO testing parameters

		Test lung parameters		Ventilator settings			
Test No.	Intended tidal volume (ml)	Compliance ml/hPa ± 10%	Linear resistance hPa/l/s ± 10%	Set rate BPM	Inspiratory time Δt_{insp} s	Inspiratory pressure Δp_{insp} hPa	BAP hPa
1	500	50	5	20	1	10	5
2	500	50	20	12	1	15	10
3	500	20	5	20	1	25	5
4	500	20	20	20	1	25	10
5	300	20	20	20	1	15	5
6	300	20	50	12	1	25	10
7	300	10	50	20	1	30	10

The standard also provides step by step instructions to follow while performing each test. Given the ventilator and test lung capabilities, the key points are that each test should be ran for a minimum of 30 breaths, the tests should not start until steady-state conditions are achieved, and that the PEEP should be calculated as the average of the airway pressure measurements over the last 50 ms of the expiratory phase.

2.6 Uncertainty Quantification

As indicated in Section 2.5.2, the results that need to be included for each ISO test are the tidal volume and PEEP measurements. The tidal volumes are calculated using the flow meter measurements, and the PEEP calculations use the airway pressure sensor measurements; both sensors have their uncertainties listed in Section 2.2.1.

The tidal volume is the change in volume during each inspiratory cycle, and the volume is calculated using a trapezoidal integration of the flow rate measurements during each inspiration period. By propagating uncertainties, the uncertainty of each tidal volume calculation is

$$\sigma_{tidal} = \sqrt{\sum \left(\frac{\sigma_Q \Delta t \left(1 + Q_{n-1}\right)}{2} \right)^2} \tag{2.3}$$

where, σ_Q is the uncertainty in the flow measurement, Δt is the inspiratory cycle time-step, and Q_{n-1} is the previous flow measurement during inspiration. Note that for the first flow measurement during the inspiratory cycle, $Q_{n-1} = 0$. In Table 2-2, the noise and accuracy of the measurement changes depending on the measured flow rate and a range is given for each. The noise and accuracy uncertainties σ_{noise} and $\sigma_{accuracy}$ used are the mean values of the ranges, and the uncertainty for each flow measurement is then calculated using,

$$\sigma_Q = \sqrt{\sigma_{noise}^2 + \sigma_{accuracy}^2}. \tag{2.4}$$

Since the ventilator's control software time step is constant at 25 ms, the PEEP is calculated as the average of the last two mask pressure measurements to adhere to the ISO standard requirements. The uncertainty in the lung airway pressure voltage read on the Huzzah32 comes from measurements of the supply voltage and the transducer output, the voltage division, as well as the uncertainty in reading the ADC pin on the Huzzah32. The resistors used in the voltage dividers were measured using an Ohm-meter with a resolution of 0.1 Ω; so, the error in the measurement from the voltage divider was deemed negligible compared to the other sources of error. Resulting in the uncertainty of the lung airway pressure voltage read on the Huzzah32 being,

$$\sigma_{V_{airway}} = \sqrt{\left(\sigma_{V_{out}}\right)^2 + \left(\sigma_{pin12}\right)^2} \tag{2.5}$$

where $\sigma_{V_{out}}$ is given as 0.25% in 2.2.1, and the uncertainty of pin 12 on the Huzzah32 was measured using a voltage supply and comparing the known supply voltage sent to the pin and what was read by the pin. The average error of these measurements was taken which resulted in σ_{pin12} being 4.1% of the measured value. The same process was performed on pin 33 which reads the voltage supply of the pressure transducer. The uncertainty of the voltage supply measurement is solely the uncertainty of the reading on pin 33, since the voltage division error is assumed to be negligible, σ_{pin33} was found to be 4.6% of the measured value. Following the propagation of uncertainty analysis and knowing the PEEP is calculated as the average of the last two pressure measurements of the expiratory cycle and how the mask pressure measurement is calculated using Equation 2.1, the expected uncertainty of the PEEP value can be found using

$$\sigma_{PEEP} = \sqrt{\left(\frac{\sigma_{p_{airway,1}}\left(1 + p_{airway,2}\right)}{2}\right)^2 + \left(\frac{\sigma_{p_{airway,2}}\left(1 + p_{airway,1}\right)}{2}\right)^2} \tag{2.6}$$

where $p_{airway,1}$, and $p_{airway,2}$ are the last and second last pressure measurements during the expiratory cycle, and $\sigma_{p_{airway,1}}$ and $\sigma_{p_{airway,2}}$ are their associated uncertainties. The uncertainty of a pressure measurement can be calculated using,

$$\sigma_{p_{airway}} = \sqrt{\left(\frac{3.125\sigma_{V_{airway}}}{V_{supply}}\right)^2 + \left(\frac{-3.125V_{airway}\sigma_{V_{supply}}}{V_{supply}^2}\right)^2} \tag{2.7}$$

where $\sigma_{V_{supply}}$ is equal to σ_{pin33}, V_{supply} is the current voltage supply measurement read on pin 33, and V_{airway} is the voltage read on pin 12.

2.7 Results and Discussion

2.7.1 Assessment of Ventilator Measurement Capability

In this section, we assess the tidal volume and PEEP value measurements from the ventilator's sensors by comparing to the values measured by the test lung. Data is presented for each of the seven ISO tests using the MGESC scheme. The data presented is for the last 30 of 40 breaths in each test. Table 2-6 contains the average of the difference between the ventilator and lung's tidal volume and PEEP calculations normalized by the target value for each ISO test.

Table 2-6: Error between ventilator and lung measurements

ISO Test	Average Tidal Volume Error %	Average PEEP Error %
1	5.61	9.20
2	7.87	-5.84
3	11.5	17.30
4	20.86	2.91
5	20.9	5.91
6	19.66	-10.99
7	33.71	-12.15

From the table it is shown that for the tidal volume calculations the ventilator measurements are always greater than the lung measurements. The tidal volume calculations for ISO tests 1 and 2 have an average difference of 8% and below, tests 3, 4, 5, and 6, have an average difference of 21% and under, then ISO test 7 has an average difference of 34%. Referring to Table 2-5, it is probable that the test lung's compliance setting has an inversely proportional relationship with the difference in tidal volume measurement from the ventilator to the test lung. The difference in the tidal volume calculations for test 4 and 5 are very similar which is a good indication that the target flow rate does not effect the measurement accuracy of the flow rate. While conducting the tests with the prototype ventilator it was found that there was a large amount of leakage for all tests between the ventilators flow meter and the entrance to the test lung where its flow meter is. Between the two flow meters there are a lot plastic connections which began to crack over time which has led to air being able to be felt coming out of the joints. One joint in particular was at the connection to the test lung, the lung's resistance couplers needed to be swapped for almost every test causing the connection to the each coupler to become more loose and cracked. This connection was not able to be permanently sealed as the couplers needed to be changed to perform other ISO tests; so we moved forward in the results by using the ventilator's tidal volume calculations in all future comparisons.

For the differences in the PEEP calculations it is found that the minimum error was 2.91%, the average error was 9.19%, and the maximum error was 17.3%. All of the ISO tests had an error of less than 13% besides ISO test 3 with the maximum error of 17.3%. The ventilator measurements are larger than the lung's measurements for tests 1, 3, 4, and 5 then they are lower for tests 2, 6, and 7. This behaviour does not correlate with any parameter settings displayed in Table 2-5. Since ISO test 3 has the largest difference in PEEP measurements, each PEEP calculation for the ventilator and lung are plotted with their associated uncertainty for ISO test 3 in Figure 2-5.

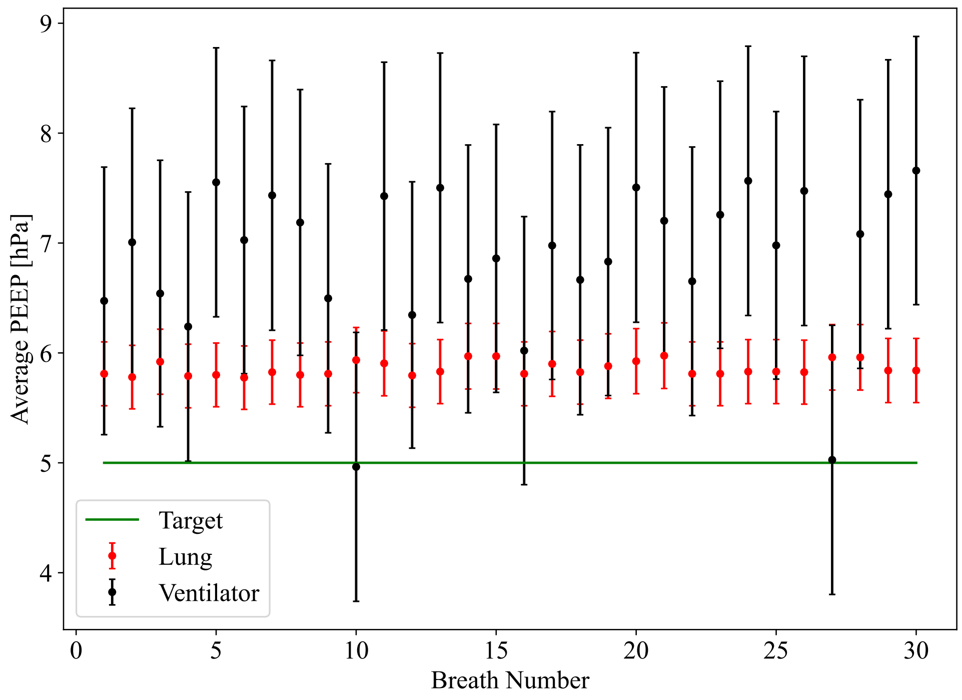

Figure 2-5: PEEP calculation differences for ISO test 3

In the figure, it is clear that the ventilator PEEP measurements are different than the test lungs PEEP measurements from the distance between the markers for each breath. However, the figure also shows that 21 of the 30 breaths have the uncertainty bars overlap each other. Since this is the ISO test with the largest error in the PEEP value this shows that even for the worst test case the results are acceptable. However, moving forward in the results, the PEEP measurements from the test lung will be used as they are more accurate and reliable.

2.7.2 Model-Guided ESC

In this section the performance of the ventilator with MGESC is assessed for the 7 ISO tests. In this and the remaining presentation of the data, ensemble averages of the flow rate over the course of the inspiratory portion of a breath cycle are used to examine the details of the ventilator and control system's response. The ensemble averaged flow rate is

$$\langle Q\left(t\right)\rangle = \frac{1}{N}\sum_{k=1}^{N} Q\left(t + \left(k-1\right)\frac{60}{BPM}\right)$$

where Q is the instantaneous flow rate, $0 \leq t < \Delta t_i$, and the index k captures the time associated with the kth out of N breaths. For our data, $N = 30$, and $\Delta t_i = 1$ s from Table 2-5. In Figure 2-6, the ensemble averages for ISO test 1, 2, 3, 5, and 6 are shown along with the standard deviation of the ensemble averaged flow rate, and its associated measurement uncertainty as shaded regions surrounding the averaged flow rate for only ISO tests 1 and 5. The standard deviation and uncertainty regions are only shown for test 1 and 5 because all of the tests at the target tidal volume of 500 mL and all of the tests at 300 mL behave the same so only one test was shown that best represents each group. Moving forward for the model-guided, model-free, and open-loop results only results from test 1 and 5 will be shown as they are representative of the other tests at their respective target tidal volume category.

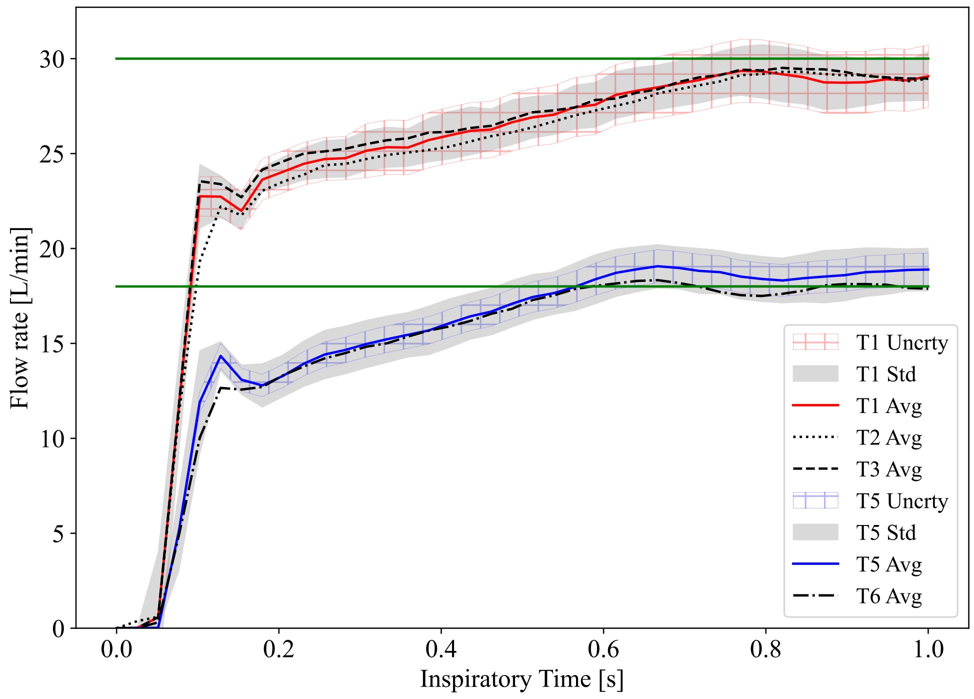

Figure 2-6: MGESC ensemble average flow rate with uncertainty and standard deviation for ISO Test 1, 2, 3, 5, and 6

At first glance it is immediately evident that for all of the tests, the control system was able to converge to the target flow rate. The first thing that is evident is that the control scheme is targeting the flow rate derived from the target tidal volume, rather than the actual volume. This will result in the tidal volumes being under target for most of the tests. The system response is able to approach get within $\sim 20\%$ of the target flow rate within 0.2 s, but then gradually converge to near the target flow rate, leading to overall tidal volumes below target. It is seen that the tests with a target flow rate of 30 L/min all reach nearly steady operation 0.8 seconds into inspiration, and at this point the ensemble averages, and their respective standard deviation and uncertainty bands, clearly show the flow rate converging onto its target. For the tests with a target of 18 L/min, the ensemble averaged flow rates converge at or above the target flow rate 0.6 seconds into inspiration, leading to larger overall tidal volumes, in line with the data in Table 2-6. Again the uncertainty bands cover the target flow rate over that final 0.4 s period for all three tests. The actual uncertainties are smaller for the lower target flow rate, while the standard deviations are similar regardless of target flow rate. This is intuitive as we know from Table 2-2 that the uncertainty of the flow rate measurement increases when the flow is greater than 25 L/min since the noise increases from 2.25 to 4%, as well as the uncertainty scaling with the magnitude of the flow rate. Then the standard deviation should mainly be driven by the ESC perturbation amplitudes a, which does not change from test to test thus, the standard deviation regions are similar. With the goal of the MGESC scheme to target a specific inspiratory volumetric flow rate, it is clear that for all of the tested ISO tests that the goal is achieved.

2.7.3 Comparison of Open-Loop Control and Model-Guided ESC

In this section, we present open-loop results to demonstrate the impact that the MGESC has on ventilator performance and to find out if there are limitations of the MGESC scheme which stem from the hardware used. For the open-loop tests, the control input is determined solely from the valve model (Equation 3.1). Figure 2-7 shows the results of the open-loop tests compared to the MGESC tests, following the style of Figure 2-6.

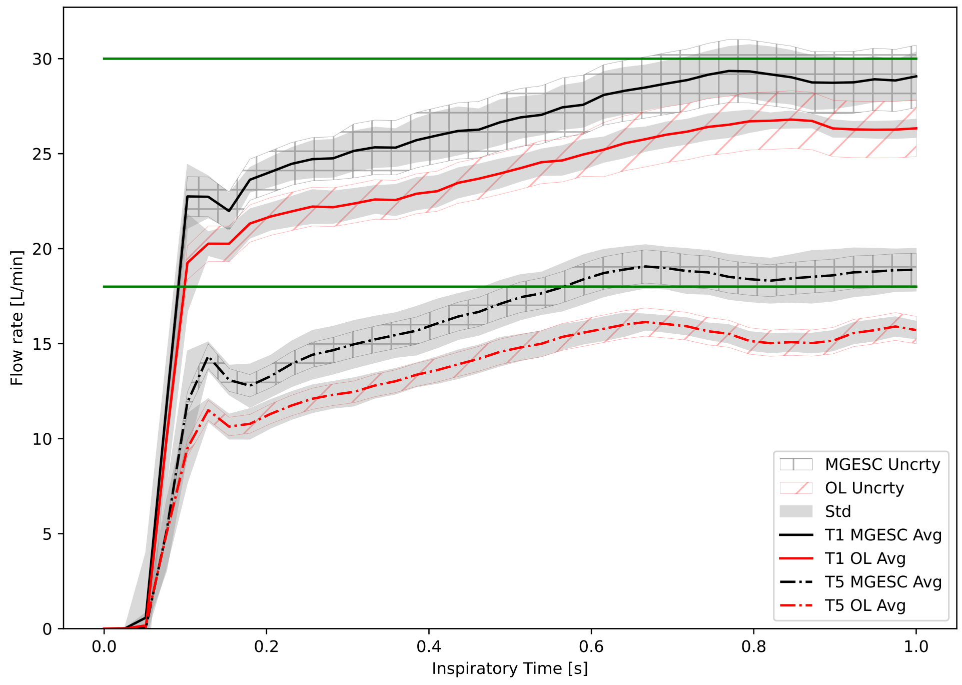

Figure 2-7: Open-Loop ensemble average flow rate

Comparing the shape of the ensemble average flow rates shown in this figure it is evident that the flow rate behaves in the same qualitative manner in both the MGESC and open-loop control schemes, and since in the open-loop cases a constant control input is used the entire time, the rapid initial rise and then longer, slow rise is seen to be the PV's opening transient behavior. Although the transient behaviour's are similar, the open-loop results all under-predict the target flow rate, using the initial guess calculated in Equation 2.2. Now, despite a constant control input, in the first 0.2 seconds of inspiration the standard deviation regions have similar sizes comparing the MGESC to open-loop results. In the above section reviewing the MGESC results it is said how the standard deviation region should be mainly driven by the oscillating nature of ESC. However, with this comparison it is clear that the standard deviation is not entirely driven by the control scheme used, and in the first 0.2 seconds of inspiration it is driven by what is most likely measurement uncertainty and non-repeatability of the valve's initial opening transient.

In Figure 2-7, it is shown how the settling time for the flow rate is roughly 0.6 s for the 18 L/min target and 0.8 s for the 30 L/min target. This settling time seems to be unaffected by the use of a control scheme. This illustrates that the MGESC scheme rapidly finds the appropriate control input, but cannot accelerate the valve's transient behavior. This can be further confirmed by reviewing the ensemble average of the control input during inspiration for ISO test 1, shown in Figure 2-8.

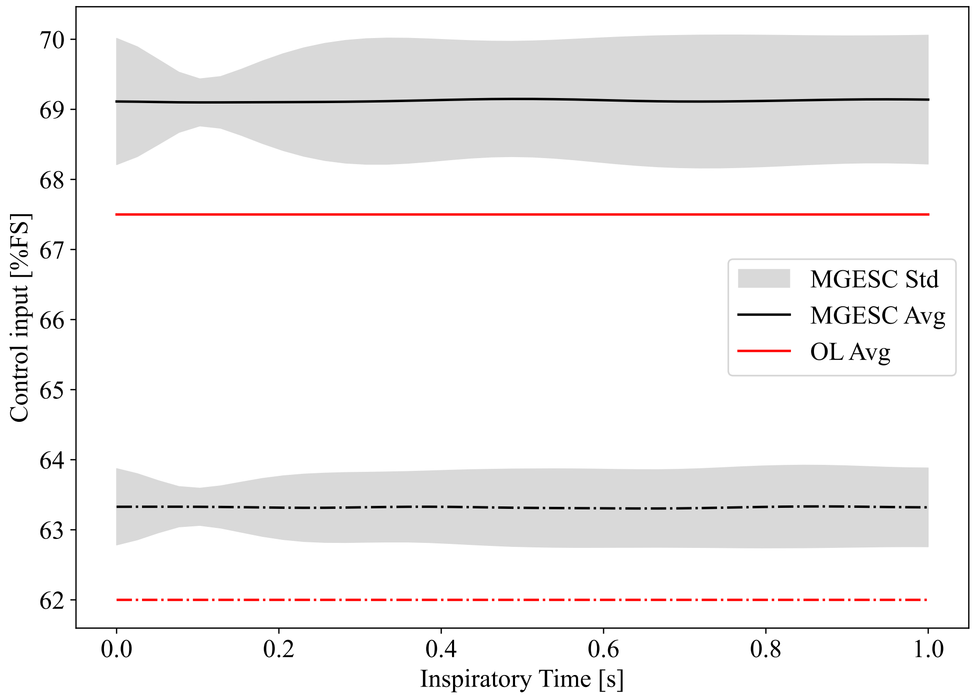

Figure 2-8: Ensemble average control input during inspiration for MGESC and OL ISO Test 1

From the figure it is clear that the ensemble average of the control input during inspiration stays practically constant further supporting that the used of the MGESC control scheme does not accelerate the valve's transient behavior. An interesting result found in the figure is that when the flow rate reaches a kinking point, in which it's fast response turns to a slow response, the standard deviation of the ensemble average of control input narrows at the same time during inspiration. This means that at around 0.1 seconds the control scheme gives the most consistent control input, not explain why the valve's transient behavior changes at this point.

2.7.4 Comparison of Model-Free and Model-Guided ESC

In this section we compare model-free ESC (MFESC) and MGESC to demonstrate the impact of the model on system performance. Figure 2-9 shows the results of the MFESC tests compared to the MGESC tests, following the style of Figure 2-7.

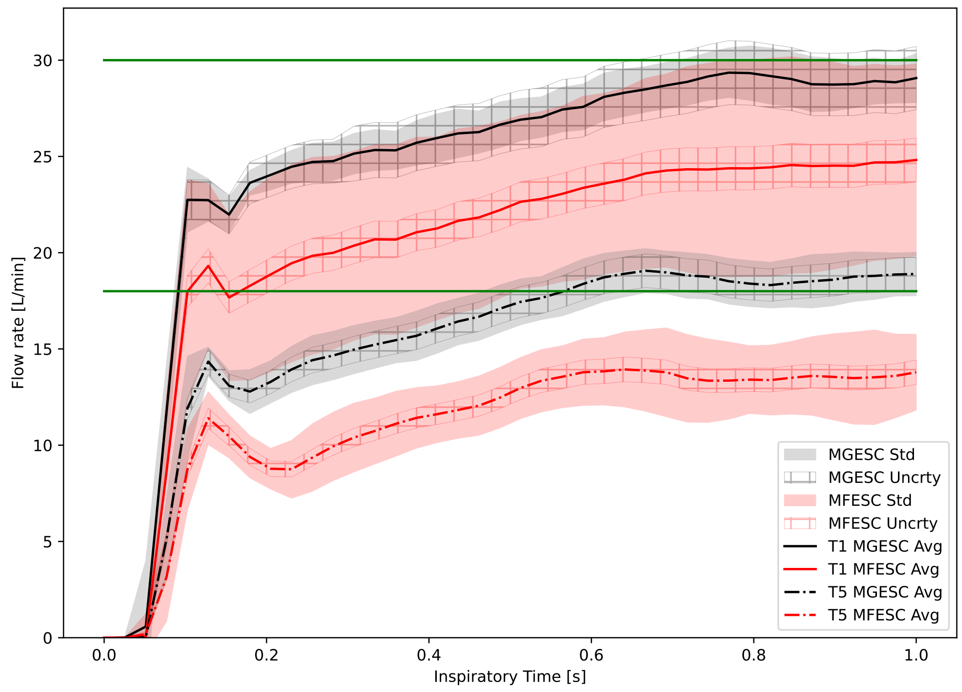

Figure 2-9: Model-free ensemble average flow rate

From the figure it is clear that the standard deviation for the ensemble average for each test is significantly larger than the uncertainty. This is intuitive because the flow rates are increasing from breath to breath as they approach the target flow rate, shown in Figure 2-10 which illustrates the average flow rate during each breath with its standard deviation as the y-axis error bar.

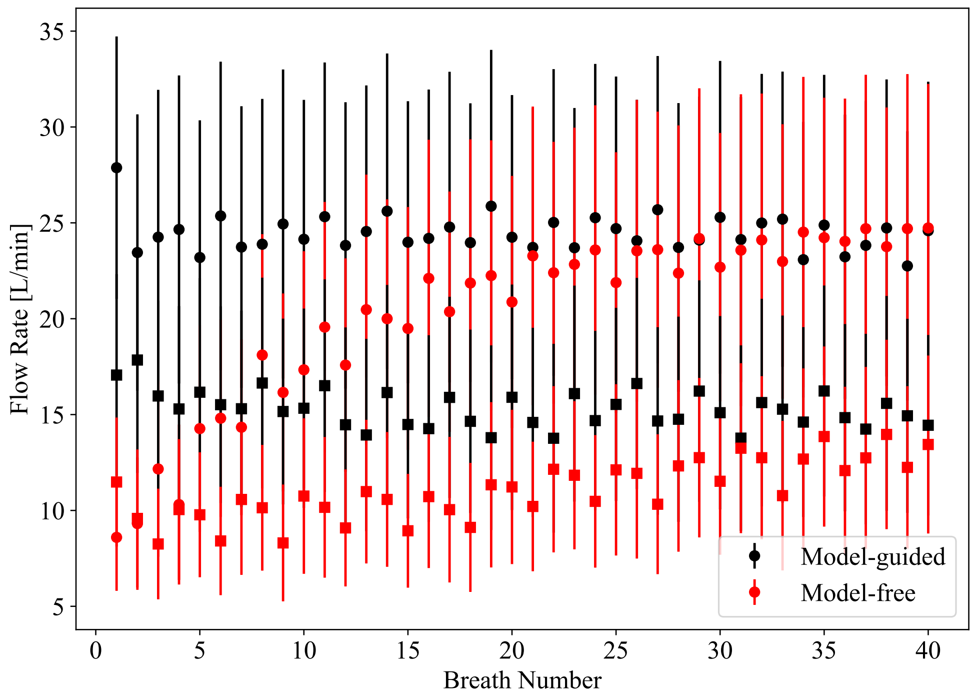

Figure 2-10: Model-free average control input per breath with the standard deviation as error bars for ISO tests 1 and 5

The standard deviation region for the model-free tests, shown as the red shaded area, is large and the upper region, just reaches the target flow rate for the 30 L/min and slightly undershoots the 18 L/min tests in Figure 2-9 which coincides with the results in Figure 2-10 as the average flow rate for each breath converges for test 1 but does not for test 5 for the model-free cases. Convergence will be defined for the model-guided and model-free cases, using the data illustrated in Figure 2-10, as when the average flow rate for a breath settles while the standard deviation above the average includes the target flow rate. With the natural transient of the valve, the target flow rate can not be achieved for the entire duration of inspiration and thus the average will always be below the target. Using this method, the MGESC scheme for ISO test 1 converges after 2 breaths, and for ISO test 5 it converges after

11 breaths. Then for the MFESC scheme ISO test 1 converges after 31 breaths, and test 5 converges fails to converge within 40 breaths. All of the convergence times for the ISO tests for both the MGESC and MFESC schemes are listed in Table 2-7.

Table 2-7: Model-Guided and Model-Free ESC convergence times

Test No.	Convergence Time [# of Breaths]	
	Model-Guided ESC	Model-Free ESC
1	2	31
2	6	40++
3	2	40+
5	11	40+
6	22	40++

Reviewing the model-guided ESC's convergence times it is found that the control scheme converges fastest for ISO test 1 and 3, followed by test 2 then test 5, and finally test 6 with the slowest convergence of 22 breaths. For the model-free ESC tests only ISO test 1 converged within the 40 breaths at 31 breaths, then ISO test 3 and 5 appeared to be converging shortly after the 40 breath mark. ISO test 2 and 6, both having a BPM of 12, appeared to be a long way from converging. So, because the model-free tests were unable to converge for most tests within the 40 breaths tested, it is clear the value added from using a model to provide the ESC scheme with an initial starting control input.

2.7.5 Varying Supply Pressure for ISO Test 2

Using the MGESC scheme and the parameters listed for ISO test 2, the supply pressure is varied from 419 kPa to 501 kPa to determine the robustness of the control scheme to variations in supply pressure. ISO test 2 parameters were used as the MGESC has good performance for these parameters compared with the other ISO tests. Now, both the valve's initial guess for the target volumetric flow rate and the 'just closed' value, for the expiratory control input that results in zero flow, are calculated using Equation 3.1. Using the average of the volumetric flow rate during inspiration for each breath, found in Figure 2-11, the effect of the varying supply pressure on the flow rate during inspiration will be illustrated.

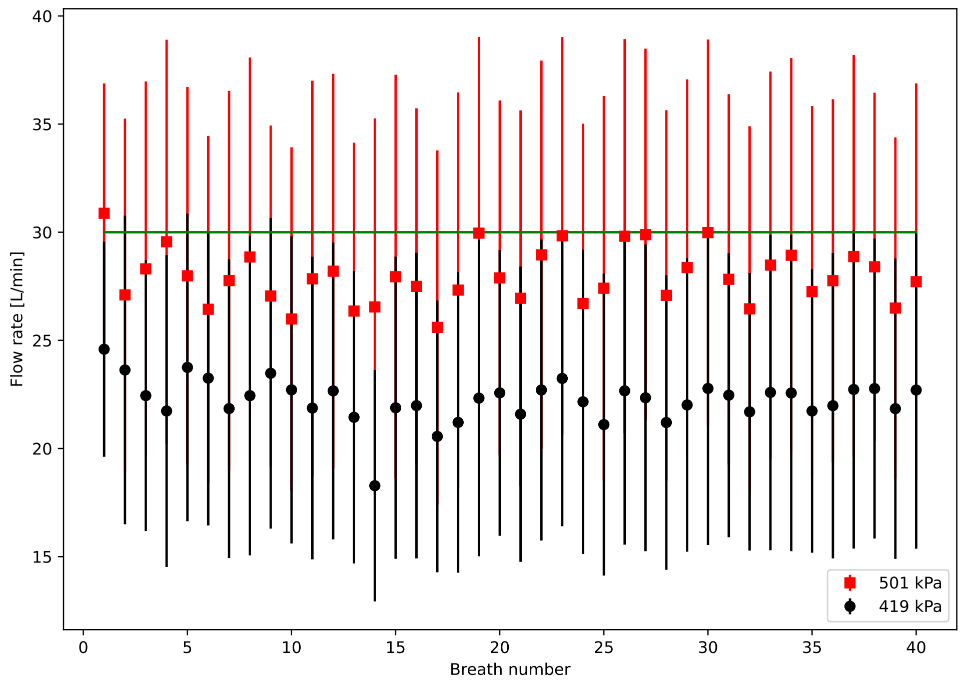

Figure 2-11: Varying supply pressure for ISO test 2

From this figure it is clear that by increasing the supply pressure, the average flow rate during inspiration is increased for all breaths. Although the average flow rate is significantly lower than the target flow rate, the standard deviation bar for the 419 kPa test undershoots the target by a small amount for the majority of breaths. With the average flow rate lower than target and not reaching the target flow rate during inspiration for most breaths, the MGESC scheme operating at a supply pressure of 419 kPa will significantly undershoot the target tidal volume. Whereas at a supply pressure of 501 kPa the average flow rate will be larger than it is at nominal supply pressure and from Figure 2-11 at this higher supply pressure the results will be much closer to the target tidal volume than at the nominal supply pressure.

2.7.6 Quantifying the Controller Robustness

From Section 2.7.2, it is known that the MGESC works as intended for all 7 ISO tests. Each ISO test can be treated as a unique patient and thus a unique disturbance for the controller. The average achieved tidal volume as a fraction of the target value for each test indicates the performance of the ventilator, while the standard deviation of these normalized averages is an indication of the MGESC's robustness to variations in the patient. These quantities are given in Table 2-8.

Table 2-8: Quantification of ventilator performance and robustness

ISO Test	Average Normalized Tidal Volume
1	0.89
2	0.96
3	0.90
4	1.00
5	1.00
6	0.98
7	1.12
Average	0.98
StDev.	0.077

From the average values in the table, it is found that the ventilator undershoots the intended tidal volume by a maximum of 11% and overshoots the intended tidal volume by a maximum of 12%. Considering that the control scheme targets volumetric flow rate rather than tidal volume, the +12%/-11% range is indicative of adequate performance. The standard deviation of these normalized averages is also given in the table (7.7%), and this is an indication of how much the ventilator performance varies based on variations in target tidal volume and patient parameters. Thus, we can expect the ventilator to typically reach within +/-7.7% of the target tidal volume.

2.8 Conclusions

Implementing the proposed model-guided ESC scheme on the open-source ventilator design and performing the available ISO standard tests resulted in the control scheme being able to converge across the range of test lung parameters. The ventilator measurements were accurate to their target values and comparable to the measurements gathered by the test lung with the largest average error in the difference between the ventilator tidal volume calculation and the test lung's calculation being 34%, then for the PEEP calculation it was 17.30%. With the two tests removed the difference in tidal volume calculations from the ventilator to the test lung are within 21% for the tidal volume, and within 13% for the PEEP for ISO tests 1, 2, 4, 5, and 6. Despite the ventilator measurements for ISO test 3 and 7 being inaccurate, the MGESC scheme was still able to converge upon a specified target inspiratory volumetric flow rate. For the robustness of the controller it was determined that a standard deviation in tidal volume measurement of 7.7% of the intended tidal volume was the maximum contribution from varying the patient parameters.

Key observations on the control scheme and ventilator hardware were made in the results section whilst comparing the MGESC scheme to the MFESC and open-loop control schemes. It was found that the settling time for the flow rate during inspiration is 0.6 seconds for the tests with a 18 L/min target flow rate and 0.8

seconds for the tests with a 30 L/min target. These times were unaffected by the use of a control scheme as the same settling times where found in the open-loop results were the control input is held constant. Although the settling times were unaffected by the control used, they were found to be a function of the proportional valve's control input. This meaning that despite the control scheme changing the control input, if a constant input is given throughout the breath the flow rate during one inspiratory breath would look similar during that settling time, if the flow rates are comparable. This illustrates that the MGESC scheme rapidly finds the appropriate control input, but cannot accelerate the valve's transient behavior. The opening transient behaviour of PV was found to be largely non-repetitive by reviewing the open-loop test result's standard deviation of the ensemble average flow rate during inspiration. The ability of the MGESC to quickly find the appropriate control input is due to the use of a model. This is confirmed when the converge time of the MGESC scheme was compared with the convergence time of the MFESC scheme for all valid ISO tests. It was determined that the introduction of a model to provide an initial guess to the ESC scheme reduces the convergence time by a minimum of 29 breaths. From the open-loop results it is found that the model used under predicts consistently and needs to be increased from its current value of +12%FS. Reviewing the MGESC results it is found that during inspiration the target flow rate is only achieved in the later part of the breath, and thus the tidal volumes are constantly undershoot their target. A change in the ESC's objective function to include the tidal volume calculations would essentially increase the average flow rate during inspiration to the target flow rate. The average flow rate during inspiration was also found to increase as the supply pressure was increased from nominal. Overall, the ventilator design is able to provide accurate measurements to the control system, and the proposed model-guided ESC scheme is able to achieve convergence across the range of varying lung parameters given by the ISO standard.

2.9 Bibliography

[1] Health Canada Services. Ventilators for patients with COVID-19, April 2022.

[2] Medical Device Shortages During the COVID-19 Public Health Emergency, December 2022.

[3] Peter Loftus. Ventilator makers ramp up production amid coronavirus crunch. *Wall Street Journal*, March 2020.

[4] Nonhlanhla Dube, Qiujun Li, Kostas Selviaridis, and Marianne Jahre. One crisis, different paths to supply resilience: The case of ventilator procurement for the COVID-19 pandemic. *Journal of Purchasing and Supply Management*, 28(5):100773, dec 2022.

[5] A. Abba et al. The novel mechanical ventilator milano for the COVID-19 pandemic. *Physics of Fluids*, 33(3):037122, mar 2021.

[6] University of Cambridge. The oxygen and ventilator system initiative (ovsi). 2022.

[7] Maikon Lorran Santos, Leonardo Rakauskas Zacharias, and VinÃcius Rosa Cota. Open-source hardware to face COVID-19 pandemic: the need to do more and better. *Research on Biomedical Engineering*, 38(1):127–138, February 2021.

[8] Samuel J. Raymond, Sam Baker, Yuzhe Liu, Mauricio J. Bustamante, Brett Ley, Michael J. Horzewski, David B. Camarillo, and David N. Cornfield. A low-cost, highly functional, emergency use ventilator for the COVID-19 crisis. *PLOS ONE*, 17(3):e0266173, March 2022.

[9] Michael Madekurozwa, Willy V. Bonneuil, Jennifer Frattolin, Daniel J. Watson, Axel C. Moore, Molly M. Stevens, James Moore, Jakob Mathiszig-Lee, and Joseph van Batenburg-Sherwood. A Novel Ventilator Design for COVID-19 and Resource-Limited Settings. *Frontiers in Medical Technology*, 3, October 2021.

[10] Sara Zulfiqar, Hamza Nadeem, Zamen Tahir, Minnaam Mazhar, and K. M. Hasan. Portable, Low Cost, Closed-Loop Mechanical Ventilation Using Feedback from Optically Isolated Analog Sensors. In *TENCON 2018 - 2018 IEEE Region 10 Conference*. IEEE, October 2018.

[11] Julienne LaChance, Manuel Schottdorf, Tom J. Zajdel, Jonny L. Saunders, Sophie Dvali, Chase Marshall, Lorenzo Seirup, Ibrahim Sammour, Robert L. Chatburn, Daniel A. Notterman, and Daniel J. Cohen. PVP1 The People's Ventilator Project: A fully open, low-cost, pressure-controlled ventilator research platform compatible with adult and pediatric uses. *PLOS ONE*, 17(5):e0266810, May 2022.

[12] *PneuViewÂ®3.3 Software Manual*. Michigan Instruments, rev: 2021-01 edition, 2021.

[13] Juergen Hahn and Thomas F. Edgar. Process control systems. In *Encyclopedia of Physical Science and Technology*, pages 111–126. Elsevier, 2003.

[14] H. Shibata and N. Mitsukawa. Comparison of robustness between adaptive control and PID control. In *Proceedings of TENCON '93. IEEE Region 10 International Conference on Computers, Communications and Automation*. IEEE.

[15] J. Doyle. Robust and optimal control. In *Proceedings of 35th IEEE Conference on Decision and Control*. IEEE.

[16] *Learning-Based Adaptive Control*. Elsevier, 2016.

[17] Ying Tan, Xiang Chen, Youying Hua, and Qingyuan Tan. Model-guided extremum seeking case studies. *International Journal of Adaptive Control and Signal Processing*, 36(3):708–728, December 2021.

[18] Iso 80601-2-12:2020. *International Organization for Standardization*, 2021.

[19] *Datasheet SFM3300*. Sensirion, version 1.4 edition, March 2022.

[20] *TruStabilityÂ® Board Mount Pressure Sensors: SSC Series*. Honeywell, 1985 Douglas Drive NorthGolden Valley, MN 55422, October 2021.

[21] *Series AP directly operated proportional valves*. Camozzi Automation, May 2022.

[22] McMaster-Carr. Air Directional Control Valve 6124k402.

[23] McMaster-Carr. Compact Threaded Solenoid Diverting Valve 2565n15.

[24] *Raspberry Pi 4 Computer Model B*. Raspberry Pi Trading Ltd., January 2021.

[25] Adafruit HUZZAH32 ESP32 Feather Board.

[26] *Use and Maintenance Manual: Series AP Directly Operated Proportional Valves*. Camozzi Automation.

[27] Kartik B. Ariyur and Miroslav Krsti&cacute. *Real-Time Optimization by Extremum-Seeking Control*. Wiley-Interscience, 2003.

[28] Alexander Skafte. An Introduction to Extremum-Seeking Control. 2017.

[29] Miroslav Krstić and Hsin-Hsiung Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595–601, April 2000.

[30] *RS485 Sensor Cable: Sensirion-HDLC Command Set*. Sensirion The Sensor Company, 7 edition, June 2018.

[31] *Training & Test Lung Training & Test Lung Operation Manual.* Michigan Instruments, January 2021.

Chapter 3

MODEL-GUIDED EXTREMUM SEEKING CONTROL FOR MECHANICAL VENTILATORS PART 2: NUMERICAL IMPACTS OF GEOMETRY CHANGES ON PERFORMANCE

3.1 Introduction

At the beginning of the COVID-19 pandemic, academia and industry came together to support and create "open-source" ventilator designs. Publications stemming from most of these projects contained little information on the ventilator control systems. To address this gap, the current authors developed an model-guided extremum-seeking control (MGESC) approach [1] and applied it to the Mechanical Ventilator

Milano (MVM) hardware design [2]. MGESC was used since Tan et al. [3] found that for ESCs, the use of a model to provide an initial guess of the control input significantly reduces convergence times, which is desirable for a medical device such as a ventilator. In ref. [1], the model is purely based on the main proportional control valve which controls the inspiratory flow rate; changes to the specifics of other components may have little impact on performance since the flow is choked at the valve. The aim of this paper is to assess the robustness of the MGESC scheme used in ref. [1] to changes in hardware details. This is relevant due to the fact that components sourced for the prototype ventilator in ref. [1] will not necessarily be available in the future and/or in some geographic regions. Further, if the prototype design is implemented into a medical-grade product, presumably the details of many components would change and it is useful to know if the control scheme from ref. [1] would still be applicable.

In this paper, we employ numerical simulations to explore the effect of changing component details on ventilator performance. A 1-D flow network model is employed to easily vary component details. The key finding is that the numerical model is able to reproduce the experimental results with moderate accuracy. While investigating various ventilator parameters it was discovered that the main piping diameter throughout the ventilator and length of the piping in the expiratory section of the ventilator are the only two geometric parameters that significantly affect the ventilator's performance.

The remainder of this paper is organized as follows. In Section 3.2, a brief review of MGESC and the details of the control scheme used here, from ref. [1], is presented. In Section 3.3, the details of the flow model as implemented in the Amesim code by Siemens and is introduced. This is followed by assessment of the model against experimental data in Section 3.4. The parameters varied to assess robustness are then presented in Section 3.5. Finally, the results of the numerical investigation are detailed in Section 3.6.

3.2 Model-Guided Extremum Seeking Control for the MVM

ESC is a type of adaptive control in which the controller adjusts to changes in the system without a need for a system model. ESC uses an objective function, that is created as a function of the system's measurements and user targets, where the ESC will attempt to locate the global minima or maxima of said objective function. The process of seeking this extremum point is typically slow; however, with the introduction of a simple model, the ESC can locate the global extremum point faster than without one [1] [3]. The real-time model-guided extremum seeking controller design used in ref. [1] is to be created in the numerical model with the layout and parameters as illustrated in Figure 3-1.

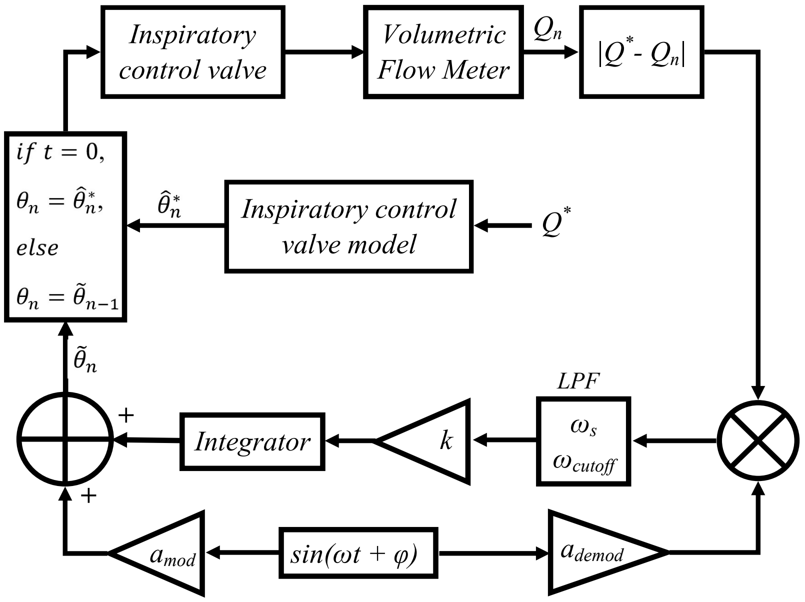

Figure 3-1: Extremum seeking controller design

The simple ESC depicted takes measurements from the ventilator system, in real-time, to create an objective function with an output of y. The objective function output is then multiplied by the perturbation signal with an amplitude of a_{demod}, a frequency of ω, and a phase shift of φ. The product of the multiplication goes through a low-pass filter (LPF) with a sampling frequency of ω_s and a cutoff frequency of ω_{cutoff}. The outputted signal from the LPF then undergoes discrete integration following the trapezoidal rule with an integration gain of k. Following integration, the perturbation signal is added to the signal with a different amplitude of a_{mod}, and a different phase shift of φ, with the sum going back into the system as the control input.

The ESC scheme used in ref. [1] implements a model that is solely based on the proportional valve in the inspiratory pathway of the ventilator to provide the ESC with an initial guess for the control input. The same model is implemented for the ESC scheme used in this study making it a model-guided extremum seeking control (MGESC) scheme. The model uses a measurement of the supply pressure (p_{sup}) to estimate the control input ($\hat{\theta}^*$) that would result in the target flow rate (Q^*) for the given supply pressure using the valve's characteristic defined in it's specification manual [4]. The equation for the model is:

$$\hat{\theta}^* = \frac{Q^* - 33.896p_{sup} + 6.7395}{-0.1105p_{sup}^2 + 0.8058p_{sup} + 0.5309} + 12. \tag{3.1}$$

As shown, the model relies on empirical data for the proportional valve given in it's documentation; so, in the case that a proportional valve with different properties is sourced, these values can easily be updated to reflect the new valve's characteristic. The same MGESC parameters are used in this study than the ones given in [1]. These parameters being: $a_{mod} = 0.0001$, $a_{demod} = 0.01$, $k = 1$, $\varphi = 0°$, $\omega_{cutoff} = 0.4\pi$ rad/s, and $\omega_s = 40$ Hz . Then the perturbation frequency ω is a function of the breaths per minute (BPM) value such that:

$$\omega = \begin{cases} 2BPM & BPM \leq 14. \\ BPM & BPM > 14. \end{cases}$$

3.3 Complete Model of the Ventilation System

To enable rapid exploration of controller robustness in the face of changes to the hardware, a numerical model is employed. The numerical model uses a one-dimensional compressible flow network implemented in the Siemens code Amesim version 2020.2 [5]. A test lung model and the control scheme are also implemented within the Amesim model.

3.3.1 Ventilator Model

Using the Amesim pneumatic library, the ventilator's air pathways are created. All components take into consideration compressibility and frictional effects. The ventilator design used in the Amesim model is the same as the physical design in ref. [1], which the hardware design in that paper is based off of ref. [2]. The pneumatic components of the Amesim model, which are the ventilator components, are displayed in Figure 3-2, alongside the control signal components depicted in red.

Very few different pneumatic components were needed to create the ventilator in Amesim as there are many repeat components. This was done to stay true to the ventilator design in ref. [1], each individual physical component in the actual ventilator was modelled as an individual, and sometimes more than one, component in Amesim. A list of the ventilator components in the Amesim model is found in Table 3-1 along with a part description, their Amesim part name, and the quantity used in the model. A more detailed description of each individual component in the Amesim model is available in Appendix B.

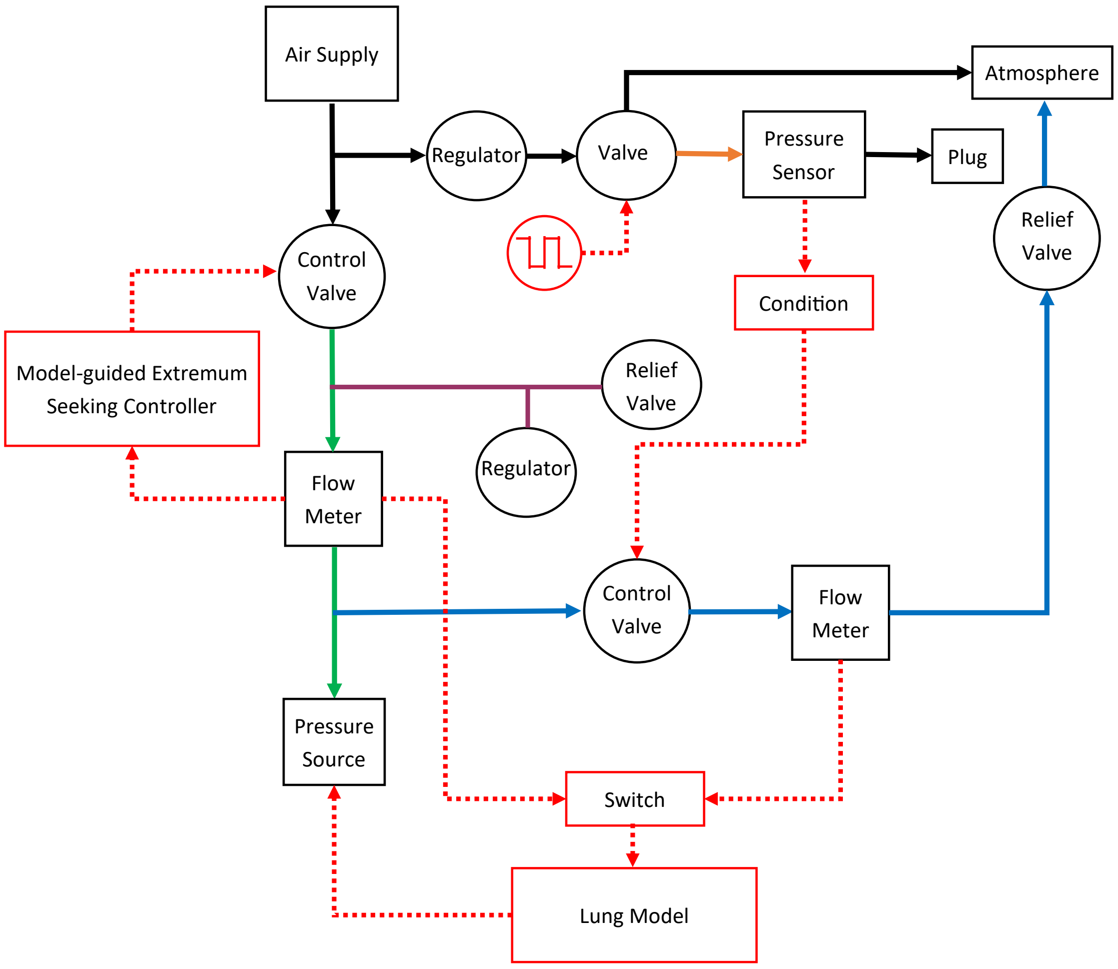

Figure 3-2: Schematic of ventilator, lung, and control system model in AmeSim

Table 3-1: Amesim pneumatic components

Part No.	Part description	Amesim part name(s)	Quantity
1	Pipe section	PNPC0	7
		PNPC1	2
		PNPC2	9
		PNPC4	3
2	Pipe elbow	PNBP001	1
3	Pipe t-joint	PN3P000	3
		TPTE001	3
4	2-port proportional valve	PNTV01	4
5	3-port proportional valve	PNPV001	1
6	Pressure regulator	PNPR12	2
7	Pressure relief valve	PNRV00	2
8	Supply pressure	PNCS001	1
9	Flow sensor	PNQS02	2
10	Pressure sensor	PNPS001	1
11	Lung model feedback	PNVS001	1
12	Solenoid valve	PNSV231_04	1
13	Atmosphere	PNAS001	4
14	Plug	PNPL01	1

It should be noted that all pneumatic components in Amesim take into consideration both compressibility and frictional effects. For parts 1 and 3, the pipe section and pipe t-joint, there are different Amesim part names for the components that refer to the specific sub-model used for each individual section of pipe. Different sub-models needed to be used depending on the sub-models of the surrounding components in order for the model to properly compile in Amesim. The remainder of this section focuses on the modeling details of the proportional valve (PV), control system, and lung model within Amesim.

3.3.2 Proportional Valve

Available PV models in Amesim have either 1st or 2nd order responses. The measurements discussed in ref. [1] showed that the valve response is complex, as depicted in their open-loop test results displayed in Figure 3-3. It can be seen how the response consists of an initial rapid rise to $\sim 70\%$ of the final value, followed by a slow response which eventually reaches a steady-state value. The experimental data is the ensemble average of the flow rate across 30 breaths during inspiration; the standard deviation across breaths is also shown to give a sense of the variability of the valve. Importantly, it was shown in ref. [1] that this behavior is inherent to the valve dynamic response and occurs in both open-loop and controlled tests.

To determine how best to model this behavior in Amesim, an analytical approximation of the response is developed. The flow rate appears to have characteristics of two first-order responses, one fast and one slow, that blend together at an average of 0.1285 seconds into the inspiration period, and with a standard deviation of 0.0002 seconds across all tests performed. Hereafter this transition is referred to as the "kink." Merging two such first order responses with the fraction of total rise and kink time fixed leaves only a single degree of freedom, however, which was inadequate to reproduce the experimentally-produced response. It was found that a combination of two first order exponential decay functions before the kink and a different combination of two first order exponential decay functions after the kink matched the valve's flow rate behavior well. This combining of functions is written out in Equation 3.2 where t is time in seconds, τ_1 is the slow time constant that stays the same before and after the kink, and τ_2 is the fast time constant that is different before and after the kink.

$$f_{combined} = \frac{(1 - e^{\frac{-t}{\tau_1}}) + (1 - e^{\frac{-t}{\tau_2}})}{2} \tag{3.2}$$

To create the combined function seen in Figure 3-3: $\tau_1 = 0.01$ seconds, $\tau_{2,before} = 0.4$ seconds, $\tau_{2,after} = 0.8$ seconds, and a delay of 0.025 seconds is introduced at the beginning of each inspiration cycle as that is the time lag associated with the experimental results from ref. [1].

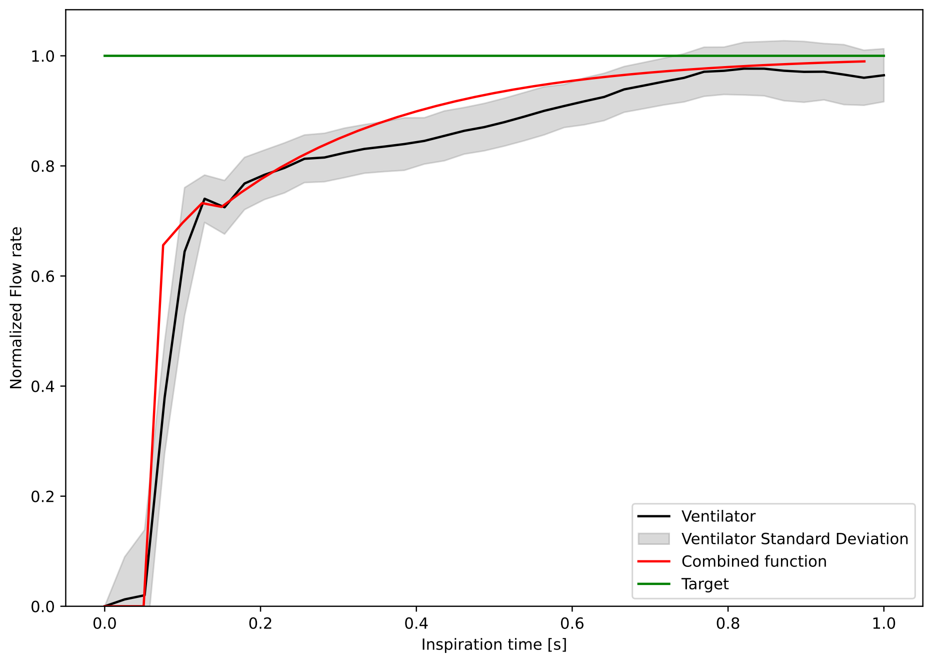

Figure 3-3: Experimental valve behaviour

It is clear that the function is not a perfect match to the experimental results; but, overall it captures the valve response accurately. The root-mean-squared (RMS) error for the analytical model over the full 1 second of response is 0.0533 L/min.

To capture this behavior in Amesim, a system of four valves is used. Three of the valves are used as the three unique first order responses, and the fourth valve is used to merge the paths of the two slow time constant valves that switch at the kink point. This is illustrated in Figure 3-4.

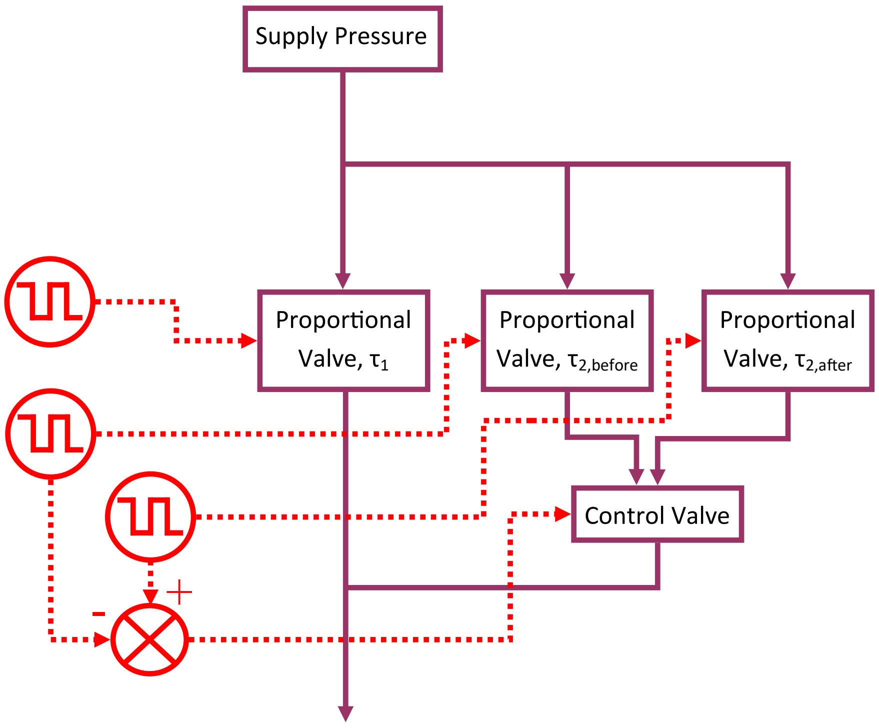
Supply Pressure
Proportional Valve, τ_1
Proportional Valve, $\tau_{2,before}$
Proportional Valve, $\tau_{2,after}$
Control Valve
+
-

Figure 3-4: Schematic of proportional valve model implemented in Amesim

Referring to Table 3-1, the three 2-port valves represent the three different time constants used, then the single 3-port valve acts as a switch which changes the flow from the middle valve, to the right valve at the specified kink point at 0.1282 seconds for all tests. The control signals for the left 2-port valve has a time constant of 0.01 seconds that receives flow for the entire duration after the 0.025 second delay mentioned above. The middle valve receives a control signal from 0.025 to 0.125 seconds, and the right valve receives a signal from 0.125 to 1 second. The control signals for the valves come from the ESC and the timing is controlled by a switch and a square wave for each valve. The layout of this is more clearly shown in Figure 3-5, where the control signal components are explained.

3.3.3 Control System

The MGESC described in Section 3.2 is implemented in Amesim using elementary control signal components. Using the signal output from the volumetric flow sensor at

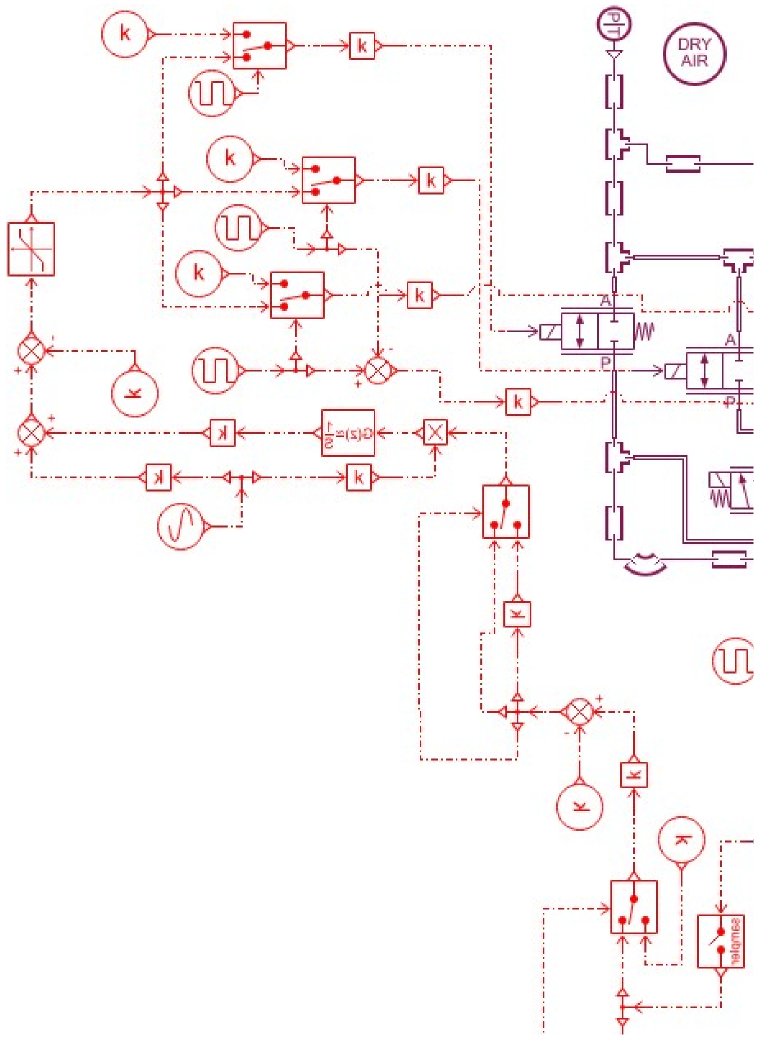

Figure 3-5: Control signal Amesim components

the end of the inspiratory pathway (see Figure 3-2), the objective function is created as the target flow rate is a user-supplied constant for a given test. The measurements from the sensor are sampled every 25 ms, matching the sampling rate achieved experimentally in ref. [1]. Figure 3-5 illustrates the control system implementation.

The dither signal is created using a single sinusoidal function block the phase angle is constant for both demodulation and modulation, but the amplitudes are different, so the signal is split into two with each section having its own gain a and b. After demodulation, the product undergoes discrete integration using the trapezoidal method, which is the same method used in ref. [1]. The control signal after modulation is subtracted by a constant and then saturated to be within the boundaries of 0 and 1 to correspond with 0% and 87.5% full scale (%FS), which are the limitations on the control input explained in ref. [1]. The subtraction is done on the control signal before the saturation, to keep the valve's characteristic nature in relation to the supply pressure is preserved in the model. This is because the valve characteristic requires a minimum input before the valve opens, and this minimum input is a function of supply pressure; the details can be found in ref. [1]. Both the initial guess and the subtracted constant are a function of the supply pressure. After saturation, the control input is then split up to go to all three of the valves. However, each valve is only active during the inspiratory part of each breath cycle, and none of the valves are active for the entire inspiration period. To control the timings of the valves, a square-wave function is used in pairing with a switch for each valve's control signal as explained in Section 3.3.2.

3.3.4 Lung Model

To model the patient or test lung in Amesim, a simple piston-cylinder device with a spring and damper is used with inspiration coming from [6]. The simple system is depicted in Figure 3-6.

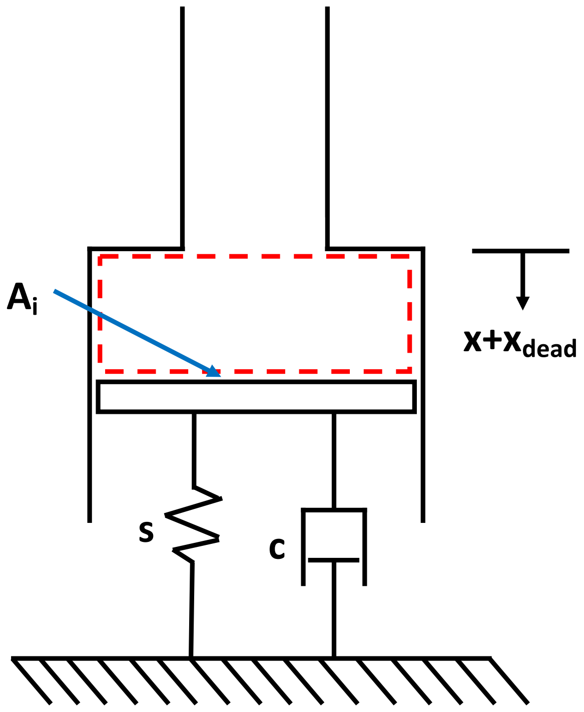

Figure 3-6: Simplified lung model

In the illustration of the model, the lung's volume (Vol) can be found using $Vol = A_i x$, where A_i is the piston interface area in units of [m^2], x is the piston's displacement which represents the lung's displacement in meters. It should be noted that in the figure the piston's displacement has a constant x_{dead} added to it to represent the dead volume of the lung. For testing this value is kept at 0 m. For the lung's properties in the model, the spring's stiffness coefficient s represents the lung's compliance in units of [mL/hPa], and the dash-pot's damper coefficient c represents the lung's linear resistance in units of [hPa/L/s]. From the application of Newton's 2nd law to the piston, an equation relating the gauge lung pressure in the lung (p_{lung}) to the lung's resistance and compliance is derived as $c\dot{x} + sx = p_{lung}A_i$. Another equation relating to the volumetric flow rate in and out of the lung, in terms of the lung displacement x in meters, is derived as $A_i\dot{x} = Q$.

To connect the lung model to the rest of the ventilator in Amesim, it is necessary that the volumetric flow rate be an input to the system and the pressure in the lung be an output of the system. For monitoring purposes, both the flow rate and volume of the lung need to be outputs of the system as well. Using this information and the two equations derived above, a one-state state-space representation (SSR) is created that models the lung that has a state equation of:

$$\dot{x} = [0]\, x + \left[\frac{1}{A_i}\right] Q \tag{3.3}$$

and an output equation of:

$$\begin{bmatrix} Q \\ Vol \\ p_{lung} \end{bmatrix} = \begin{bmatrix} 0 \\ A_i \\ \frac{s}{A_i} \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ \frac{c}{A_i^2} \end{bmatrix} Q. \tag{3.4}$$

To integrate this lung model and state-space representation in Amesim, the output of flow sensors are sampled every 0.025 seconds and feed into a discrete state-space block. Only the p_{lung} output of the SSR is needed to feedback into the pneumatic ventilator portion of the model such that the pressure increases during inspiration, and decreases during expiration. How these components are combined is displayed in Figure 3-7.

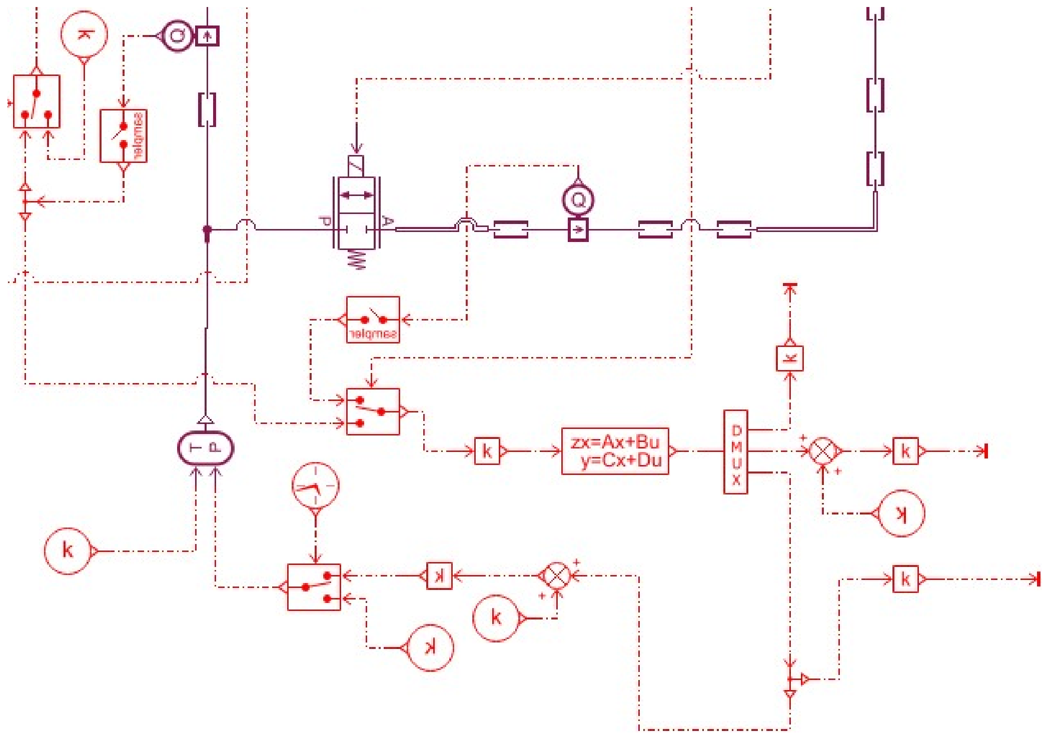

Figure 3-7: Lung model Amesim integration

3.4 Model Verification

The experiments in ref. [1] were carried out for seven standard tests from ISO 80601-2-12:2020(en), part 2-12, section 201.12.1.101 for testing a ventilator in volume-controlled ventilation modes [7]. To determine the accuracy of the Amesim model, the same seven ISO tests are simulated. In the ISO tests, the lung's resistance and compliance are varied, as well as the ventilator's BPM, bypass airway pressure (BAP) setting, and the target tidal volume. For all of the tests, the inspiration duration stays constant at 1 second. To analyze the experimental and numerical results from the ISO tests, both a quantitative and qualitative approach was taken. Quantitatively, the average error of the tidal volume and PEEP values and their respective standard deviations are compared. In Table 3-2, said comparison is performed by taking the difference of the Amesim measurement to the ventilator measurement and dividing the difference by ventilator measurement and displaying as a percentage.

Table 3-2: ISO test results

ISO Test	Average Tidal Error [%]	Standard Deviation of Tidal Error [%]	Average PEEP Error [%]	Standard Deviation of PEEP Error [%]
1	-4.8	3.7	-26.4	2
2	-12.9	4.8	-4.7	0.1
3	-7.0	3.1	-23.9	3.1
4	-15.9	5.7	13.7	5.1
5	-18.3	7.0	-17.5	8.3
6	-19.6	6.6	-6.1	2.3
7	-27.8	4.8	0.5	5.2

Reviewing the tidal volume average error, it is clear that for all tests the Amesim measurements are lower than the ventilator measurements. The largest average error is seen in ISO test 7 at -28%, and the smallest error is seen in ISO test 1 at -5%. ISO tests 1 and 3 have the smallest average errors, which these two tests exclusively share the same lung resistance setting of 5 hPa/L/s. As the lung resistance setting is increased, the average tidal volume error increases, with ISO test 6 and 7 having the worst performance at a lung resistance setting of 50 hPa/L/s. With this trend in mind, reviewing ISO test 2 and 6 with the other tests with the same lung resistance, both tests have the lowest error in their respective lung resistance groups and these two tests are the only tests with a BPM of 12. The difference in the tidal volume calculations essentially stem from the average flow rate during inspiration for each breath and by plotting the ensemble average of the flow rate during inspiration the difference in the Amesim and ventilator results during the course of inspiration for ISO test 2 and 7 is illustrated in Figure 3-8.

Reviewing the tidal volume errors, tests 1 through 4 perform significantly better than tests 5 through 7, not only by having a lower average error, but by having a lower standard deviation as well. In Figure 3-8, the ensemble inspiratory flow rate during inspiration for ISO test two and seven are plotted to compare the results qualitatively. It is clear that the Amesim model does not accurately capture the more rapid rise time associated with the lower flow rate target.

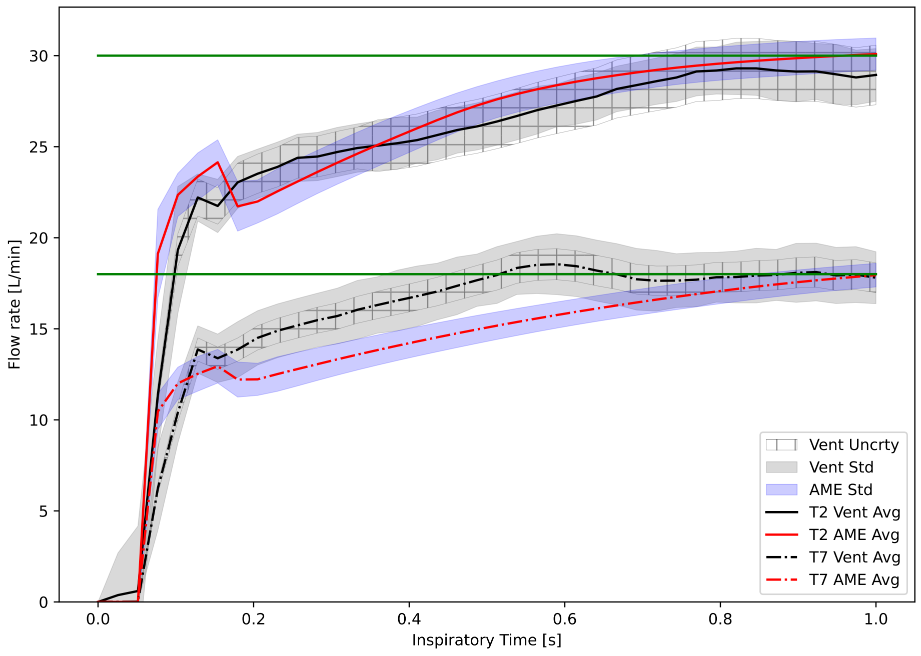

Figure 3-8: Inspiratory glow rate ensembled average for ISO test 2 and 7

Only ISO tests 2 and 7 were shown because ISO test 2 has the least tidal volume error out of the tests with a lung resistance setting of 20 hPa/L/s, and has the least average PEEP calculation out of the test with a 30 L/min target flow rate. Test 7 was shown as it has the largest tidal volume error and the least PEEP calculation error out of the tests with a target flow rate of 18 L/min. Both in Figure 3-2 and Figure 3-8, it is clear that the Amesim model does not accurately capture the more rapid rise time associated with the lower flow rate target after the kink point. This occurs because the slow valve's time constant after the kink point is kept constant for all target flow rates, with the values stated in Section 3.3.2. The physical valve in the experiment demonstrates that the slow time constant after the kink point ($\tau_{2,after}$) is inversely proportional to the targeted control input. Thus, to achieve better performance in the Amesim model, $\tau_{2,after}$ should be a function of the control input, and not constant at 0.8 seconds.

For the difference in PEEP values between the experimental and numerical results, there is one major problem that is due to the hardware used in the experiment. In ref. [1], it is explained how the expiratory pressure release valve (EPRV) that controls the baseline airway pressure (BAP) could not be set to a BAP of 5 hPa as it could only go down to around 5.4 hPa intermittently. This means that the ISO tests with a BAP setting of 5 hPa (ISO test 1, 3, and 5) will inherently have larger average errors as the BAP setting is different from the numerical model to experiment. The average PEEP error for the tests with a BAP of 10 hPa are low, with the largest error being ISO test 4 14% with a standard deviation of 5%, and that of the four tests, two have the Amesim calculations larger on average, and the other two have the Amesim calculations smaller.

In summary, the Amesim model is most accurate for tests targeting the higher flow rate of 30 L/min, a lower lung resistance setting, and with a BAP setting of 10 hPa. This leaves ISO test 2 with the with the greatest simulation accuracy out of the 7 tests, so moving forward, ISO test 2 is used for assessment of the impact of changing geometric system parameters.

3.5 Changing Parameters

In this section, physical ventilator parameters that are impractical to change in the experimental setup are changed in the Amesim model. Each parameter is varied individually, keeping all other parameters at their nominal values. There are ten parameters considered that could change in the ventilators design: the length of tubing in the branch off of the main inspiratory path (L_{branch}), the diameter for the majority of piping (D), the length of the expiratory path (L_{exp}), the length of the main inspiratory path (L_{insp}), the length of tubing from the three-way solenoid valve to the normally-open pneumatic valve that controls the flow in the expiratory path (L_{tubing}), the relative roughness of the piping components (ϵ), the supply pressure (p_{sup}), the time constant of the slow valve before the kink point ($\tau_{2,before}$), and the time constant of the slow valve after the kink point ($\tau_{2,after}$).

Each parameter was chosen as they could change if different components are used in the ventilator, but after initial investigation it was determined that most of the parameters will not have a significant affect on performance. These parameters are: L_{insp}, L_{branch}, L_{tubing}, and ϵ. The reasoning for L_{insp}, and L_{branch} is that in the inspiratory pathway, flow is controlled by the proportional valve which has a time constant of 0.01 seconds [4] and it is known that the flow after the valve will always be choked, incompressible, and laminar in practice. The pressure ratio across the inspiratory pressure valve will be 0.0228, from the upstream pressure supply at 50 psi and the maximum pressure allowed in the lung being 1.14 psi, so the flow will be choked since the critical pressure ratio for air is 0.528. To ensure that the flow is laminar in this section the Reynolds number can be calculated from

$$Re = \frac{\rho D(\frac{4Q}{\pi D^2})}{\mu}. \tag{3.5}$$

Using the properties for ideal air at 20°C, the density (ρ) can be estimate as 1.204 kg/m^3, and the dynamic viscosity (μ) can be estimated as 1.825×10^{-5} kg/m-s. Taking the nominal piping diameter (D) for this section used as 0.01905 m (0.75 in), and the maximum flow rate in the section (Q) to be 0.0005 m^3/s (30 L/min), the Reynolds number can be estimated as 2205 which puts the flow in the laminar regime. For L_{tubing}, the flow will be incompressible through this section. The control valve NOV has a time constant of 0.01 seconds, and operates with a 248 kPa pressure difference from full to empty; so, assuming the local speed of sound of the air as 343 m/s, the length of this section would need to be 1.72 meters for a pressure wave to travel the length of the tube and bounce back. Finally, the relative roughness of the internal surface of the main piping in the ventilator will not matter, as the flow will be in the laminar regime and thus the friction loss in the piping is not a function of roughness.

The remaining parameters should have some significant effect on the ventilator's performance due to their definition, location, and operating conditions. L_{exp} does not have the same behavior as the inspiratory path because it operates at a significantly smaller pressure difference. The pressure difference between the inflated lung and the BAP setting is never larger than 1 psi meaning the flow across the expiratory control valve will not be choked. So, changing the length of the expiratory pathway while the diameter and pressure difference remains constant will: reduce the energy loss of the air as it flows through the pipe, increase the initial velocity throughout the section, and thus lower the residence time of the air. Then by changing the diameter of the piping in both the inspiratory and expiratory paths will change the Reynolds number, and increase the velocity of the fluid through the pipes as the flow rate is held constant during inspiration. Due to the proportional valve's characteristic, varying the supply pressure will change the control input that results in the target flow rate, and thus the system may take longer or shorter to reach the target flow rate based on how good the model's initial guess is in Equation 3.1. Note that the minimum supply pressure tested of 4 bar is still sufficient enough to maintain choked flow through the inspiratory pathway. Lastly the slow time constant τ_2 will evidently have an affect on the valve's transient behavior during inspiration. When $\tau_{2,before}$ is increased, the maximum flow rate that is reached before the kink point should decrease. Then by is increasing $\tau_{2,after}$, the time it takes to settle on the target flow rate will increase.

The range of values tested for each significant parameter is listed in Table 3-3. For each parameter, the nominal value is defined to be the value used in the experimental setup defined in ref. [1] and is the value used in the model verification in Section 3.4.

Table 3-3: Test matrix

Parameter	Unit	Run 1	Run 2	Run 3	Run 4 (nominal value)	Run 5	Run 6	Run 7
D	[cm]	0.3175	0.635	1.27	1.905	2.54	2.8575	3.81
L_{exp}	[cm]	25	50	75	107	125	150	175
p_{sup}	[bara]	4	4.2	4.4	4.5	4.6	4.8	5
$\tau_{2,before}$	[s]	0.1	0.2	0.3	0.4	0.5	0.6	0.7
$\tau_{2,after}$	[s]	0.5	0.6	0.7	0.8	0.9	1	1.1

The limits of the range of values for each parameter were chosen as practical limits for each component. For the expiratory path length, L_{exp}, the minimum and maximum values were set to be extreme conditions where the path is unrealistically long or short. For the piping diameters (D), each value corresponds to NPT piping sizes that could be purchased from the typical hardware store or generic supplier. The minimum and maximum values of this range were set for practicality as one should be able to source components between this large range from 0.3175 to 3.81 cm. For the supply pressure range, it was said that a hospital's air supply typically fluctuates by ± 0.345 bar, so the minimum and maximum ranges tested are ± 0.5 bar. Finally, the ranges for the slow time constant τ_2 were set by incrementally increasing and decreasing by 0.1 seconds from the nominal value.

3.6 Results and Discussion

In this section, the results presented focus on the parameters that are predicted to have a significant effect on the ventilator's performance as discussed in Section 3.5. All of the parameters that were not expected to not affect the ventilator's performance were verified to have their range of values not change the results in practice.

For the diameter of the piping sections, it was found that the change in flow rate during inspiration was insignificant (as expected); however, the peak inspiratory pressure was increased as the diameter was decreased, and the PEEP increased as the diameter was decreased. In Figure 3-9, the average peak inspiratory pressure (PIP) and average PEEP across all breaths against the varying diameter is plotted on a dual-y axis.

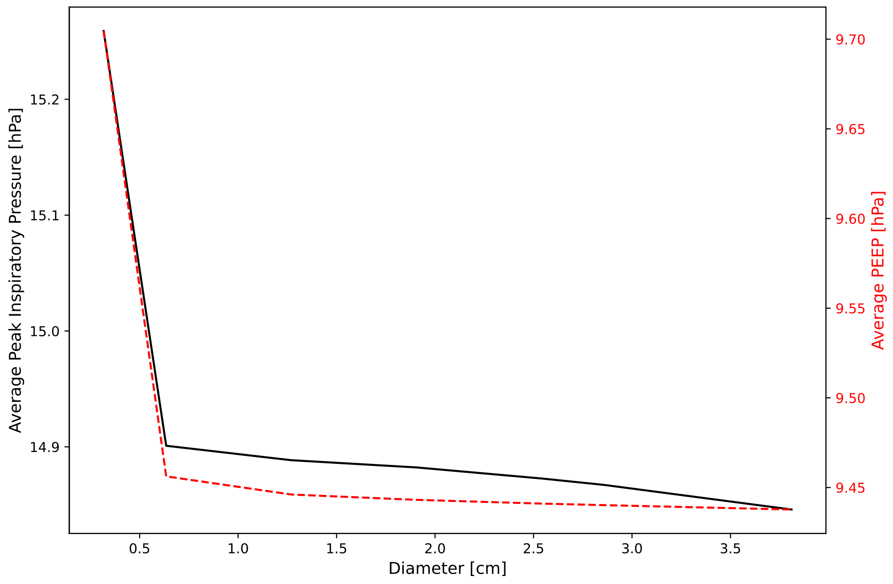

Figure 3-9: Diameter changed pressure waveform

Immediately it is seen that both the average peak pressure and average PEEP share a similar relationship with the piping diameter. The PEEP and average pressure increase at a slow linear pace until a diameter of less than 0.6 cm is achieved in which the slope is increased. In the inspiratory section the friction loss can be defined using the Darcy-Weisbach equation for head loss:

$$\frac{\Delta p}{L} = \frac{8 f_D \rho Q^2}{\pi^2 D^5}. \tag{3.6}$$

Where $\frac{\Delta p}{L}$ is the pressure loss per unit length in $[Pa/m]$, ρ is the density of the fluid in $[kg/m^3]$, Q is the volumetric flow rate of the fluid in $[m^3/s]$, f_D is the Darcy friction factor, and D is the diameter of the piping in $[m]$. The Darcy friction factor depends on the Reynolds number of the flow and since the diameter is changing, the flow regime also changes. Using 64 over the Reynolds number for laminar flows, and the Moody Diagram for transitional and turbulent flows, in Table 3-4, the flow regime and friction factor for each diameter is listed.

Table 3-4: Darcy friction factor for each piping diameter

Diameter [cm]	Flow regime	f_D
0.3175	Turbulent	0.029
0.635	Turbulent	0.036
1.27	Transitional	0.019
1.905	Laminar	0.029
2.54	Laminar	0.039
2.8575	Laminar	0.044
3.815	Laminar	0.058

While the flow is laminar, the friction factor decreases with diameter until the flow starts to transition from laminar to turbulent. As this transition occurs the friction factor increases by a factor of 1.9. As the diameter is further decreased, the flow becomes completely turbulent and the friction factor again decreases. Now, this trend is not seen in any of the results and does not follow the trend shown in Figure 3-9 meaning that the friction factor in the pipes has an insignificant affect on the ventilator's performance. Knowing that the changes in f_D are insignificant, we can look back to Equation 3.6 and know that the diameter is the only other variable that changes in the inspiratory path and that as the diameter is decreased, the frictional losses increase and scale with $1/D^5$. This results in the total pressure in the inspiratory piping to increase and the velocity through the inspiratory section to increase. Both the pressure loss from after the valve to the patient, and the velocity of the fluid within this section scales with $1/D^2$. So, because the total pressure at the inspiratory line inlet, after the valve, is increased, the outlet pressure into the lung will increase as well. Thus, resulting in the average PIP to be increasing as the diameter is decreased. The PEEP follows the same trend as the PIP to keep the pressure difference across the expiratory pathway constant at around 5.5 hPa. Even with the increase in slope in Figure 3-9, the average PIP only increases by 0.3 hPa which will be insignificant to the patient, and the PEEP actually increases to be closer to the BAP setting. Thus, changing the diameter does not have a significant affect on the pressure measurements during ventilation. To understand the effect of changing the PIP and PEEP has on the performance during expiration, the ensemble average of the flow rate during expiration is plotted for the smallest and largest diameter tested, shown in Figure 3-10.

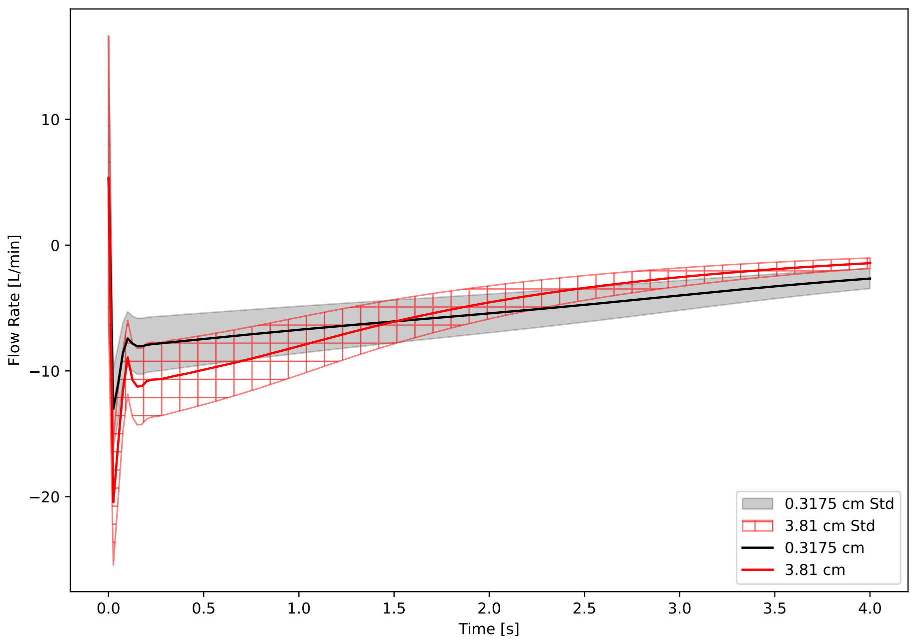

Figure 3-10: Diameter changed expiratory flow rate ensemble average

In the figure, it was found that for all the tests except for the smallest diameter, the curves were essentially identical which coincides with the results in Figure 3-9 how the slope stays the same and then increases at the smallest diameter. The reason a similar trend occurs in the flow rate during expiration is because in the previous paragraph it is explained how decreasing the diameter but keeping the same pressure difference across the section will result in an increase in velocity in the pipes due to the increase in frictional losses. Then it is this increase in frictional losses as the diameter decreases that causes the expiratory flow rates to be smaller for the smallest diameter case as this scales approximately with $1/D^5$, as previously mentioned.

Next we focus on the other geometric parameter which impacts performance, the length of the expiratory path. During expiration, it was found that when the length reached a certain minimum threshold of less than 50 cm, pressure fluctuations, and in turn flow rate fluctuations, start to occur. This phenomenon is illustrated in Figure 3-11 where the ensemble average of flow rate during the expiration period for the smallest length of 25 cm, shows large oscillations at the beginning of expiration and then smaller oscillations towards the end. The physical mechanism behind this behaviour is suspected to be the pressure waves within the tubing section. Approximating the local speed of sound of the air as 343 m/s, when the length of the expiratory section is 50 cm, the round-trip travel time of the pressure wave originating from the lung is 0.0015 seconds, and when the length is 25 cm, the round-trip travel time is 0.00073 seconds. Since the NOV has a time constant of 0.01 seconds, this explains why all of the tests have this initial oscillation from the air filling the entire volume after the valve completely opens. By decreasing the length of the section, it decreases the length the pressure wave has to travel, and thus its round-trip travel time, which appears to increase the magnitude and duration of the initial oscillation. The oscillations begin to occur again in the flow rate measurement 1.5 seconds into expiration which are also presumably caused by the acoustic resonances of the pressure waves within the expiratory section. To further understand why they occur a more detailed acoustic analysis needs to be performed at this minimum length; however, that is outside the scope of this work. For our purposes, it is known that undesirable fluctuations in the volumetric flow rate occur once the length of the expiratory section is decreased to less than 50 cm. A length which is unrealistic in a practical setting as the expiratory length includes all piping components from the patient's mask to the expiratory pressure relief valve which will almost always be greater than 50 cm, as in our case it is 107 cm where 70 cm of that length is the generic hospital tubing from the patient's mask to the ventilator.

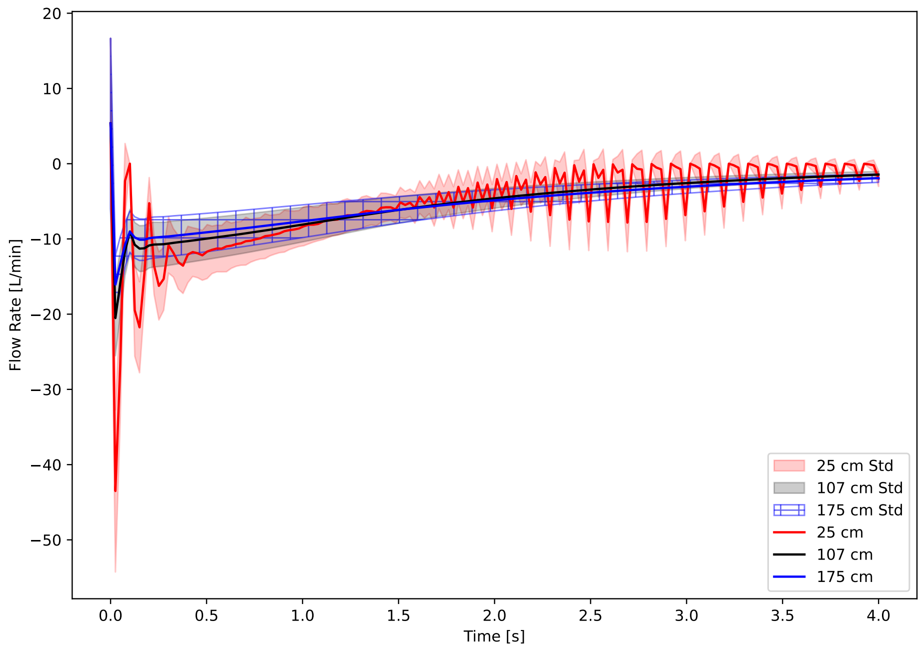

Figure 3-11: eLength expiratory flow rate ensemble average

Plotting the ensemble average of the flow rate during expiration also demonstrates that as the length of the expiratory section is increased, the peak expiratory flow rate is decreased. This coheres with the understanding of incompressible un-choked flow through a pipe. When the expiratory control valve is first opened, since the flow is incompressible, the pressure difference between the lung and the expiratory pressure release valve is at its maximum. Thus, when it first opens the velocity of the flow is at its maximum and with a shorter length, the flow has to travel less thus resulting in a higher peak flow rate.

Moving onto the non-geometric parameters that impacted the ventilator's performance, the changing of the supply pressure significantly affected the flow rate during inspiration, indicating that the model for the valve flow rate vs. open fraction was not being correctly adjusted for the supply pressure. Using the ensemble average of flow rates during inspiration in Figure 3-12, it is seen how as the supply pressure is increased above nominal, the flow rate is consistently increased.

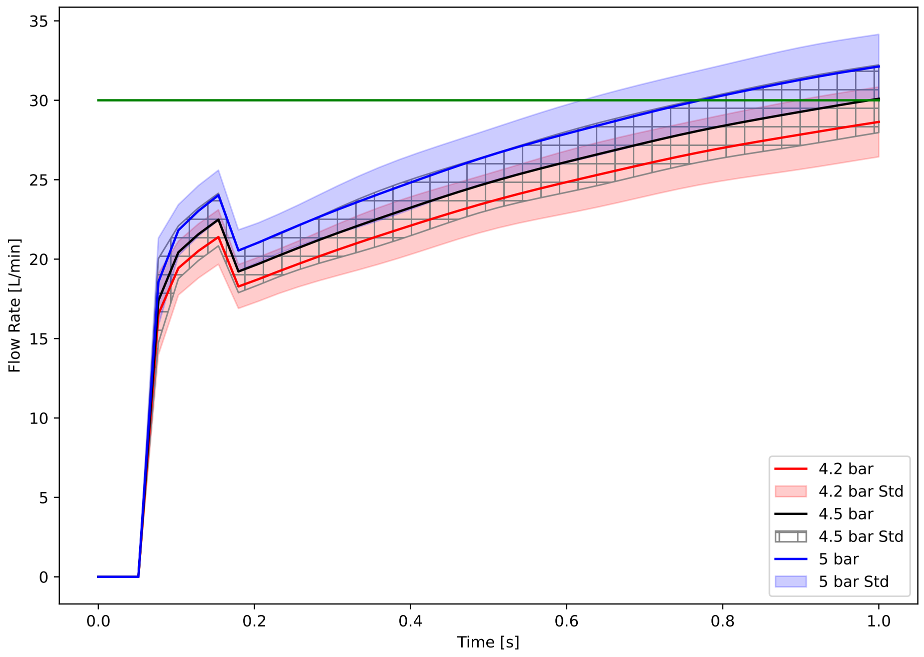

Figure 3-12: Supply pressure changed inspiratory flow rate ensemble average

The shaded regions in the figure shows the standard deviation of the ensemble average at that moment in time during inspiration, which are all large, for the three tests illustrated. Note that the smallest supply pressure tested, as listed in Table 3-3, of 4 bar is not included as the target volumetric flow rate of 30 L/min could not be achieved with a control input less than or equal to the maximum control input of 87.5%FS. Thus, the lowest supply pressure tested that yielded usable results was 4.2 bar. For the range of pressure supply from 4.2 to 5 bar, it is illustrated how the MGESC does it's job by converging the flow rate to the target because all of the standard deviation ranges include the target flow rate. When the tidal volume for each breath is plotted for the same three tests, shown in Figure 3-13, it supports this result as the tidal volumes converge for all the tests converge as more breaths are simulated.

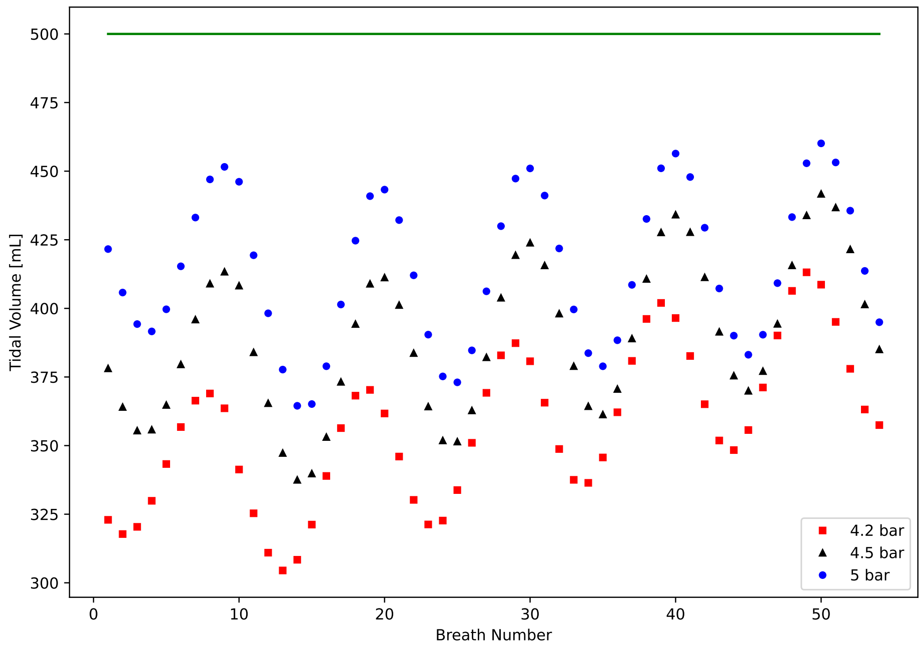

Figure 3-13: Supply pressure changed tidal volumes

This behavior is desirable as it shows the control scheme's ability to eventually converge onto the same results despite large changes in supply pressure from nominal. However, this really only proves that the ESC is working correctly, not the model that provides the initial guess. When the average control input during inspiration per breath is plotted for the same three tests, it becomes clear that the model is not adequately adapting to changing supply pressure. This is illustrated in Figure 3-14, with the y-axis error bars being the standard deviation of the average control input during inspiration for each breath.

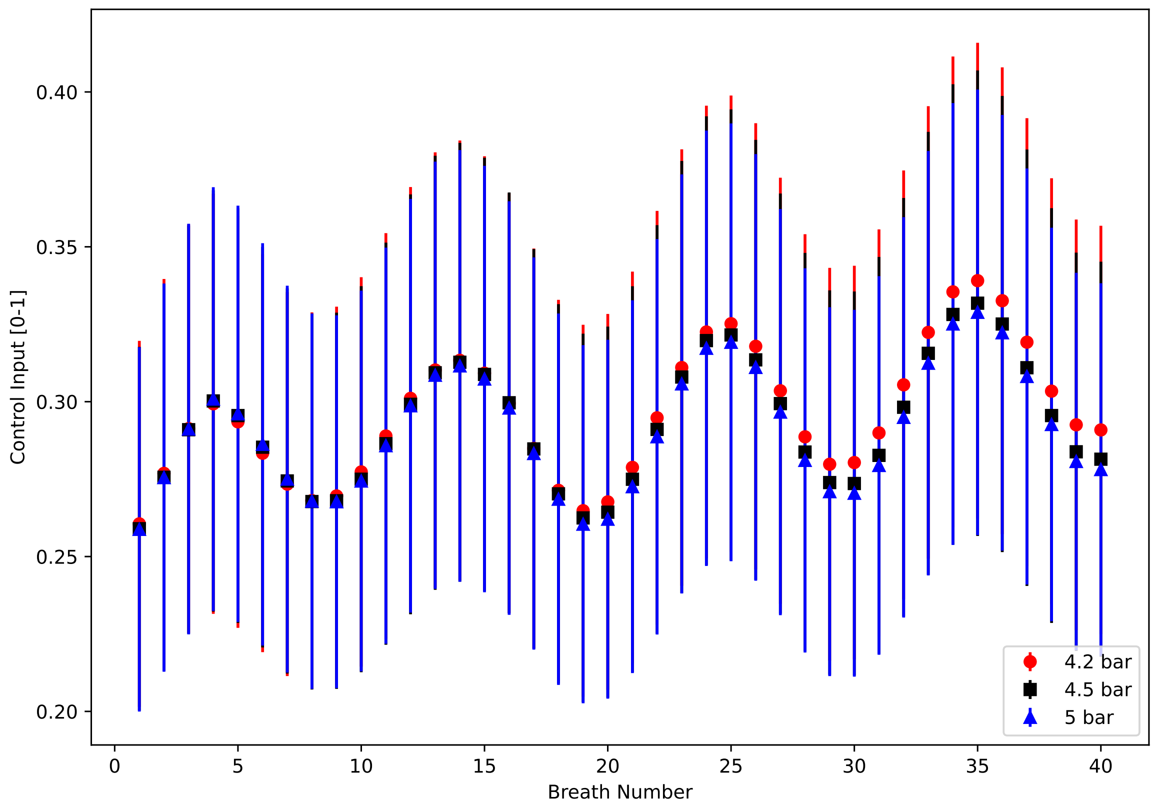

Figure 3-14: Supply pressure changed average control input

In the figure it is clear that the model that provides the initial control input guess to the ESC is not sensitive enough to the supply pressure, as it is seen to vary a little – but not enough. The ESC is working as lower the supply pressure evidently results in a larger control input needed as shown by the points diverging after 20 breaths.

The last two non-geometric parameters that affected the ventilators performance were the slow time constant before and after the kink point, $\tau_{2,before}$ and $\tau_{2,after}$. Both parameters significantly changed the behaviour of the flow rate during inspiration; however, both parameters did not have a significant impact on the tidal volume. Graphing the ensemble average of the flow rate during inspiration for $\tau_{2,before}$ in Figure 3-15, it is clear only one thing changes.

Investigating the change of the slow valve's time constant before the kink drastically changed the behavior of the flow rate before the kink point during inspiration. Figure 3-15 shows the ensembled average of the flow rate during inspiration and it is evident that as the time constant is decreased, the peak flow rate before the kink point is increased.

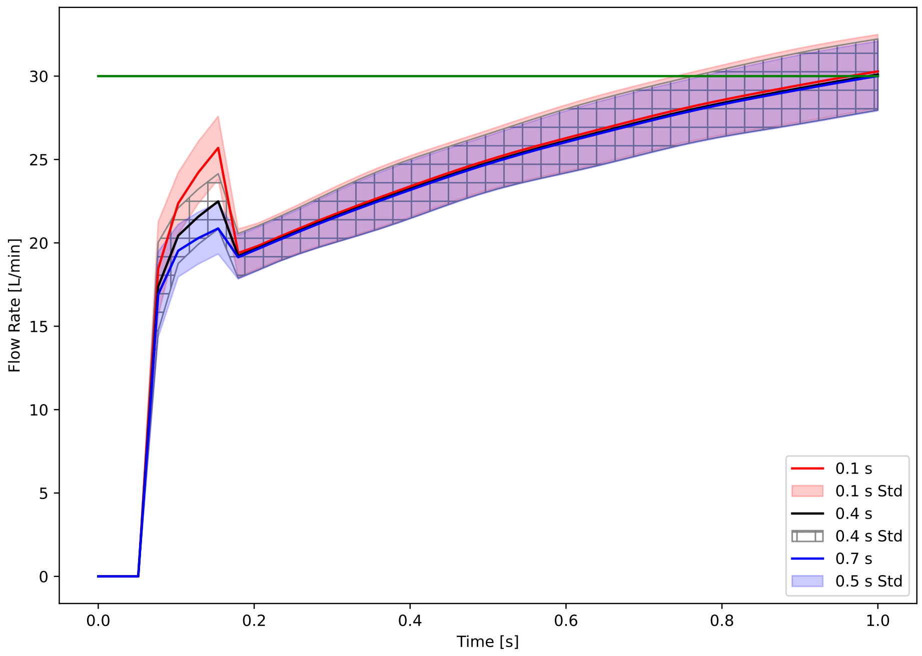

Figure 3-15: $\tau_{2,before}$ changed inspiratory flow rate ensemble average

The one thing that changes is the peak flow rate before the kink point which intuitively increases as the time constant decreases. Due to how the proportional valve is modelled in Amesim, the flow rate at the kink point remains unchanged. This increase in peak flow rate does increase the tidal volume per breath; however, the increase is small and insignificant. Similarly for $\tau_{2,after}$, Figure 3-16 is created in the same manner as Figure 3-15.

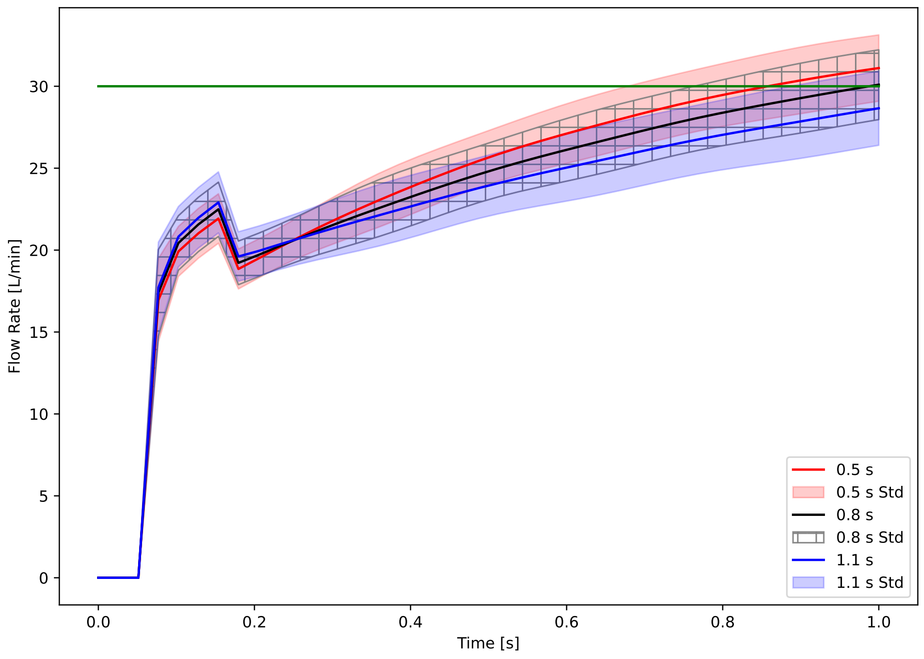

Figure 3-16: $\tau_{2,after}$ changed inspiratory flow rate ensemble average

From the figure it is clear that $\tau_{2,after}$ had a large impact on the flow rate during inspiration after the kink, and a small impact before the kink point. As expected, when the time constant is decreased, the flow after the kink point is increased. But, it is also shown how as the time constant is decreased, the peak flow rate before the kink slightly decreases and the flow rate at the kink point is decreased. Now, the change in flow rates appear to be larger for $\tau_{2,after}$ compared to $\tau_{2,before}$; but, to demonstrate this the average tidal volume as a function of the time constants are shown in Figure 3-17.

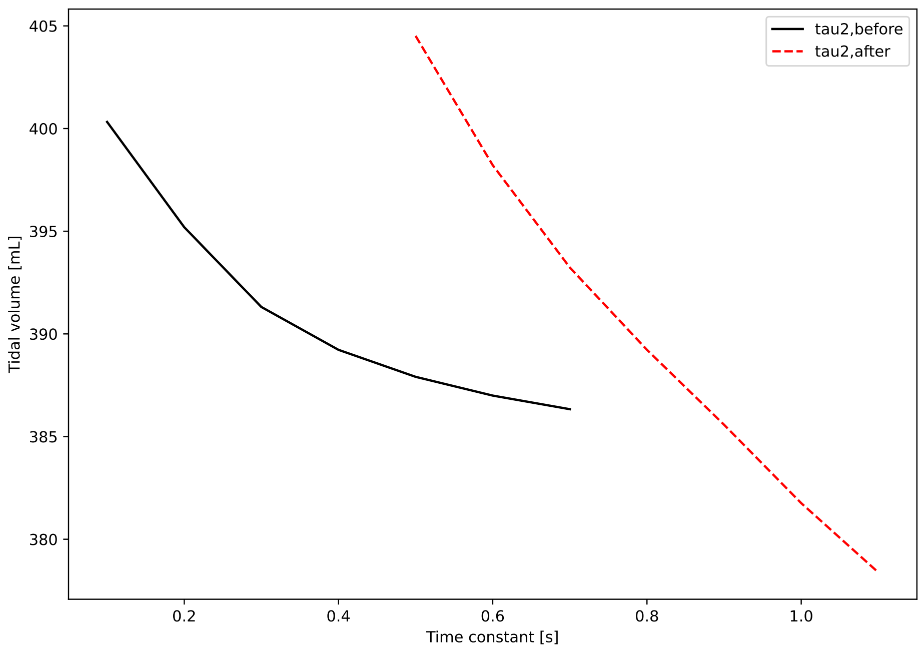

Figure 3-17: $\tau_{2,before}$ and $\tau_{2,after}$ affect on average tidal volume

For $\tau_{2,before}$, the difference in tidal volume for the smallest and largest time constant is roughly 14 mL; and for $\tau_{2,after}$, the largest difference being roughly 27mL. This results in the time constant after the kink point having twice as large of an affect on the average tidal volume achieved than for the time constant before the kink point. The inspiration duration is constant at 1 second; so, this result is likely due to the time after the kink point being significantly longer than the time before the kink point. Overall, these conclusions imply that for better performance the valve should have the fastest, or smallest, time constant both before and after the kink point.

The other parameters listed in Section 3.5 that were said to not affect the ventilators performance, as their value is varied, were also tested using Amesim as to verify our initial claim. The results from the simulations confirmed that varying their value did not affect the ventilators performance.

3.7 Conclusion

Using the proposed ventilator design and model-guided extremum seeking controller design from ref. [1], a numerical model of the system was created. For the numerical model we employed a a 1-D flow network model in Amesim to explore the effect of changing component details on ventilator performance. The entire Amesim model included the ventilator, control system, and a model of a patient's lung such that any parameter could be investigated within the system. Using 7 ISO standard tests the Amesim model results were compared to the experimental results given in ref. [1]. It was determined that an increase in error between the numerical and experimental tidal volume calculations when the lung's resistance setting is increased and when the target tidal volume is decreased. Also that the Amesim model does not accurately capture the more rapid rise time associated with the lower flow rate target because τ_2's value before and after the kink point are kept constant although their behavior suggests they are a function of the target volumetric flow rate. Thus, to achieve better performance $\tau_{2,before}$ and $\tau_{2,after}$ should be a function of the target flow rate. In ref. [1], it is stated that the EPRV that controls the BAP could only go as low as 5.4 hPa; but, the EPRV in the numerical model follows the ISO test BAP setting of 5 hPa. This resulting in the difference in PEEP calculations between the numerical and experimental results for ISO tests 1, 3, and 5, to be abnormally large. Overall, it was determined that ISO test 2 had the greatest simulation accuracy out of the seven tests; so, only settings listed in ISO test 2 were used while varying the ventilator's parameters. Out of all of the parameters investigated, only two geometric parameters had an effect on the ventilator's performance: D, and L_{exp}. A decrease in D resulted in a slight increase in the average peak inspiratory pressure, PEEP, and expiratory flow rate, until a large increase was seen in these values for the smallest diameter tested of 0.3175 cm. This sudden increase is a result of the cross-sectional area being proportional to the square of the piping diameter. During expiration, it was found that when L_{exp} reached a minimum threshold of 25 cm, pressure fluctuations, and in turn flow rate fluctuations, start to occur. For the non-geometric parameters, varying the supply pressure demonstrated that the ESC works; but, it made it clear that the model that provides the initial control input guess to the ESC is not sensitive enough to the supply pressure. By varying the slow time constant of the proportional valve before and after the kink, $\tau_{2,before}$ and $\tau_{2,after}$, it was determined that by decreasing either time constant, the average tidal volume across all breaths would increase. More specifically, the difference in tidal volume for the smallest and largest time constant tested for $\tau_{2,before}$ was roughly 14 mL, and for $\tau_{2,after}$ it was determined to be roughly 27 mL. Deeming $\tau_{2,after}$ twice as significant as $\tau_{2,before}$ on the ventilator's performance.

For all of the parameters investigated, the ESC scheme was effective at converging to the target flow rate during inspiration. A major design flaw in the numerical model was that the slow time constant associated with the proportional valve in the inspiratory path before and after the kink point $\tau_{2,before}$ and $\tau_{2,after}$ were kept constant when they need to be a function of the target volumetric flow rate. Besides this design error and the faulty EPRV, the numerical model was able to achieve tidal volumes within 16% difference and PEEP values within 14%.

3.8 Bibliography

[1] R. Wardell, P. Saini, and J. Defoe. Model-guided extremum seeking control for mechanical ventilators part 1: Measured performance. *TBD*, 2023.

[2] A. Abba et al. The novel mechanical ventilator milano for the COVID-19 pandemic. *Physics of Fluids*, 33(3):037122, mar 2021.

[3] Ying Tan, Xiang Chen, Youying Hua, and Qingyuan Tan. Model-guided extremum seeking case studies. *International Journal of Adaptive Control and Signal Processing*, 36(3):708–728, dec 2021.

[4] *Use and Maintenance Manual: Series AP Directly Operated Proportional Valves.* Camozzi Automation.

[5] Siemens. *Siemens Simcenter Amesim 2020.2*, 2020.

[6] Sunder Neelakantan, Yi Xin, Donald P. Gaver, Maurizio Cereda, Rahim Rizi, Bradford J. Smith, and Reza Avazmohammadi. Computational lung modelling in respiratory medicine. *Journal of The Royal Society Interface*, 19(191), jun 2022.

[7] Iso 80601-2-12:2020. *International Organization for Standardization*, 2021.

Chapter 4

SUMMARY, CONTRIBUTIONS, AND FUTURE WORK

In this thesis, a model-guided extremum seeking control scheme is created for a mechanical ventilator with the intent of the control scheme's performance not being significantly affected by changes in patient parameters as well as major changes in the ventilator's design. In this chapter, the two works shown in this thesis are outlined with a description of how they relate to one another. Subsequently, the key contributions and recommendations for future work are discussed.

4.1 Summary

The testing of model-guided extremum seeking controllers on various applications has been sparsely investigated, and the use of adaptive control techniques for mechanical ventilators has been studied by many. The combination of these two fields has not been investigated. This clear gap in literature, as well as its practical uses, are the motivation behind the current work. Due to its possible benefits, a model-guided extremum seeking controller was created, that requires minimal knowledge on the ventilator's design, to be able to adapt to large patient variations as well as variations in the design of the mechanical ventilator.

In Chapter 2, it was found that ISO test 7 had a 34% difference in it tidal volume calculation between the ventilator and lung measurement, and ISO test 3 had a 17.3% difference in the PEEP calculation. But, by removing these two tests from the results leads to the tidal volume calculations being within 21% and the PEEP calculations being within 13%. The introduction of a model to provide the ESC with an initial control input reduced convergence time by a minimum of 29 breaths, and this result was achieved even though it was determined that the model used consistently under-predicts the control input needed to achieve the target volumetric flow rate. The settling time for the flow rate during inspiration was unaffected by the type of control used, and that the settling time was a function of the control input. But, most importantly, the MGESC was able to converge upon the specified target inspiratory volumetric flow rate for all tests; and it was determined that a standard deviation in tidal volume measurement of 7.7% of the intended tidal volume was the average contribution from varying the patient parameters. Solidifying that the control scheme is able to perform, and is robust, across the range of patient parameters defined by the ISO standard.

In Chapter 3, it was determined that an increase in error between the numerical and experimental tidal volume calculations occurred when the lung's resistance setting is increased and when the target tidal volume is decreased. A part of the reason for this was because the Amesim model does not accurately capture the more rapid rise time associated with the lower flow rate target because the modelling of the proportional valve is kept constant although it's behavior suggests that it is function of the target volumetric flow rate. From Chapter 2, it is known that the experimental expiratory pressure release valve that controls the baseline airway pressure could only go as low as 5.4 hPa; but, the valve in the numerical model follows the ISO test BAP setting of 5 hPa, resulting in the error in PEEP calculations between the experimental and numerical results being inflated for tests with a BAP of 5 hPa. Without these sources of error, the numerical model was able to achieve tidal volumes within 16% difference and PEEP values within 14% of the experimental results. Out of the ISO tests performed, ISO test 2 had the best performance so its parameters were used for testing the varying ventilator parameters in the model. Out of the geometric parameters investigated only two were found to have a significant affect on the ventilator's performance: the main diameter of piping, and the length of the expiratory piping section. Now, the MGESC was able to adapt to all of the ranges of parameters tested and was always able to converge upon the target flow rate. However, undesirable fluctuations in measurements occurred when the diameter of piping was less than 0.635 cm, or when the length of the expiratory section was less than 50 cm. Similar to the results in Chapter 2, it was determined that the model that provides the initial control input guess to the ESC is not sensitive enough to changes in the supply pressure.

These works both relate to each other by demonstrating the performance of the proposed model-guided extremum seeking control scheme. Chapter 2 shows the control scheme's performance on a physical experimental setup with changing lung parameters. Then, Chapter 3 uses a numerical model of the experimental setup to test the control scheme's performance by varying different parameters within the ventilator's design. Together these two works clearly demonstrate the suitability of the use of a model-guided extremum seeking controller for a mechanical ventilator operating in volume-controlled ventilation modes.

4.2 Contributions

This thesis makes several contributions. An open-source ventilator hardware and software design is created using all commercially available components and open-source software. The ventilator hardware measurements were tested against the measurements from a test lung apparatus to demonstrate the accuracy of the sensors in Chapter 2. An open-source numerical model of the open-source ventilator design is created using a common industrial software in Chapter 3. Within the numerical model it is explained how to represent the behaviour the proportional valve in the open-source ventilator design, and how to model a lung using a simple state-space representation. A model-guided extremum seeking controller was created using only the volumetric flow rate measurement, and the model used for the controller is specific to the main inflow controlling proportional valve. The robustness of the MGESC scheme to patient variation is tested experimentally, in Chapter 2, using the open-source ventilator design and by following the proper ISO standard procedures. The robustness of the MGESC scheme to changes in the ventilators geometry is tested, in Chapter 3, using a 1-D flow network model to perform the numerical simulations. In both the experimental results and numerical simulations, it is shown that the MGESC scheme performs throughout all of the variations in patient and ventilator geometry. This proves that a MGESC is a viable choice for mechanical ventilators operating in volume-controlled ventilation modes.

Most importantly, the works in this thesis introduce extremum seeking control techniques to the application of mechanical ventilators. The MGESC, open-source ventilator design, and 1-D flow network model created in this thesis could be improved upon and used as a starting point to conduct further research involving extremum seeking controllers for use on mechanical ventilators.

4.3 Future Recommendations

This section contains recommendations for future work that would significantly improve both the ventilator and model-guided extremum seeking controllers performance based on the observations made in this thesis. To improve the ventilator's results, the pressure measurements from the patient's mask need better filtering as there is significant noise which led to large inaccuracies in PEEP calculations. A new EPRV valve should be sourced that is capable of reaching 5 hPa, and the ventilator should be more thoroughly investigated for leakage as it was quite significant.

To improve the MGESC, the model needs to be tuned to be more accurate, and to be more sensitive to changes in the supply pressure. Furthermore, the objective function for the ESC should be created by using the projected tidal volume during inspiration and comparing it to the target tidal volume so that the ventilator can better achieve the target tidal volumes.

Appendices

Appendix A

Ventilator Hardware

Only a high-level description and illustration of the hardware of the ventilator was given in the main body as that is all that is needed for the purpose of the paper as more focus is on the design of the control scheme for the ventilator. However, it is still important that the detailed information on the ventilator's hardware is included to make all resources open-source for this project. A more in depth image of the ventilator layout is given in Figure A-1 below, which shows all of the individual piping components between the major components of the ventilator.

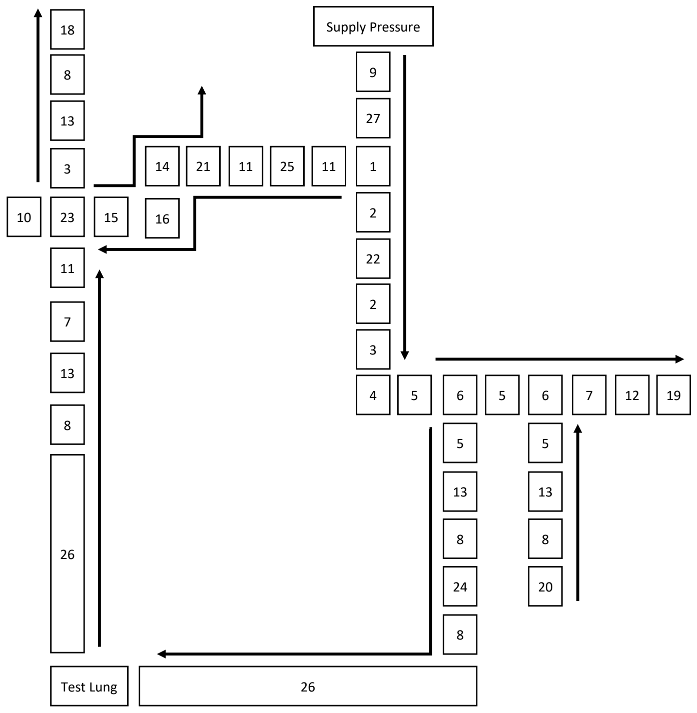

Figure A-1: Ventilator component layout

The entire ventilator was built in-house and instructions on how to assemble the ventilator from start to finish as well as the full bill of materials are included in the sections below.

A.1 Assembly Instructions

1) Using the detailed ventilator layout in Figure A-1, layout all of the hardware components in the correct orientation and sequence.

2) Using the pipe joint tape, wrap the male threads of all of the pipe components then apply a small amount of the pipe joint compound on top of the tape.

3) Make all of the male to female pipe connections in which you prepped the male

end in step 2.

4) For the connections between part 8 and 13, which there are 4 of them, the 3/4 NPTF X 1/4 NPTF fitting must be drilled out on the 1/4 NPTF side. The threads must be drilled out to allow for the 15mm side of part 8 to fit snugly in. Drill the pieces out and attach them permanently with an epoxy sealant that works with both metal and plastic. Let sit before installation following the chosen epoxy's instructions.

5) For the pressure transducer upstream of the proportional valve, a pressure tap is needed in part 2 directly upstream of the proportional valve. The outer diameter of the tubulation is 1.59mm and the tubulation must sit flush with the internal diameter of the part. Drill the hole, place the tubulation flush with the ID, seal the tubulation with epoxy. Allow for the epoxy to set before pressurization.

6) Connect the remaining ventilator components together.

7) Following the wiring diagram in Figure A-2 below, make all of the necessary connections with caution. Note that breadboard's were used in assistance to assembly.

8) With all of the connections in place, set up the Raspberry Pi and install all of the necessary programs.

9) Using the Arduino IDE and the ESP32 code provided in A.3.2, flash the Huzzah32 with the new program.

10) In the command console, the python script, in A.3.1, will now be able to be executed.

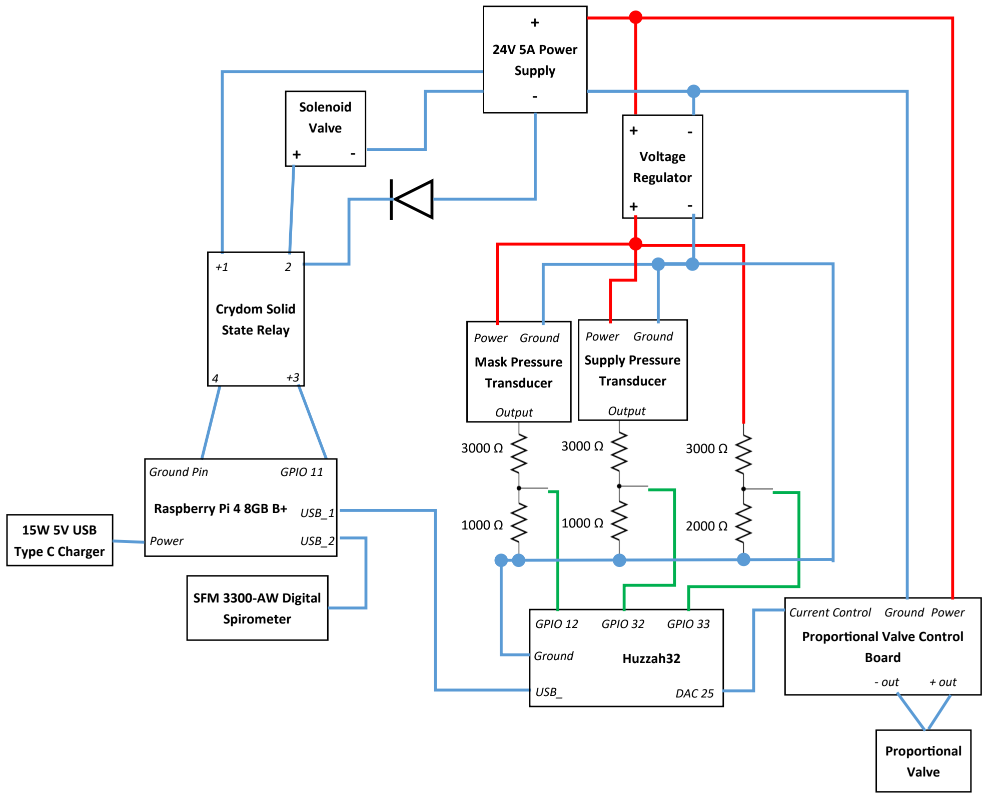

Figure A-2: Ventilator wiring diagram

A.2 Bill of Materials

The ventilator built consists of mainly of commercial-off-the-shelf components. Five pieces needed to undergo minor modifications in order for the ventilator to be completely assembled and reading to start testing. Four of the pieces are a piece of piping where the internal threads were drilled out so that a piece of plastic mating tubing could connect the metal parts of the ventilator with the plastic parts. The fifth piece that was altered was the metal connection pipe directly upstream of the proportional valve. A tiny hole was drilled in this piece for the pressure tap to be glued into place, flush with the internal diameter of the pipe. In Table A-1 below, all of the components used in the ventilator as well as components used in the manufacturing process are included.

Table A-1: Ventilator Bill of Materials (BOM)

Part No.	Item Name	Quantity	Vendor Part No.	Vendor	Cost/ 1 Item
1	Right-Angle Tee Adapter 1/4 NPTF x 1/4 NPTM	1	48805K571	McMaster-Carr	37.45
2	Adapter, 1/8 BSPP M X 1/4 NPT F, SST	2	4822T337	McMaster-Carr	39.81
3	Fitting, 3/4 NPTM X 1/4 NPTM, SST	2	48805K828	McMaster-Carr	16.83
4	Elbow, 3/4 NPTF X 3/4 NPTF, SST	1	4452K415	McMaster-Carr	12.80
5	Fitting, 3/4 NPTM X 3/4 NPTM X 3 in Long	4	4830K195	McMaster-Carr	5.82
6	T-Junction, 3/4 NPTF X 3/4 NPTF X 3/4 NPTF	2	4464K52	McMaster-Carr	15.80
7	Bushing, 3/4 NPTM X 1/4 NPTF	2	4452K168	McMaster-Carr	6.62
8	ISO 5356 Equal M22/F15 - M22/F15	5	CA001824	Medical e-Shop	0.4

9	Tubing, 22mm OD, 15mm IDX 40mm Long, Blue	1	N/A	Medical e-Shop	1.35
10	Cap, 1/4 NPTM	1	5232T229	McMaster-Carr	7.13
11	Fitting, 1/4 NPTM X 1/4 NPTM	3	4830K132	McMaster-Carr	3.00
12	Fitting, 1/8 NPTM X 1/4 NPTM	1	48805K86	McMaster-Carr	9.99
13	Fitting, 3/4 NPTF X 1/4 NPTF	4	48805K816	McMaster-Carr	27.07
14	Swivel Adapter, 1/4 Barbed ID X 1/4 NPTM	1	91465K91	McMaster-Carr	9.32
15	Swivel Adapter, 1/4 Barbed ID X 1/8 NPTM	1	91465K11	McMaster-Carr	15.66
16	Tubing, 1/4 ID	1	52375K12	McMaster-Carr	7.90
17	Silicone Tubing 1/8 ID x 3/16 OD 16ft	1	N/A	Amazon	8.49
18	PEEP Valve	1	00-118	Medical e-Shop	7.88
19	Airway Pressure Limiting Valve	1	99045K11	McMaster-Carr	36.88

20	Negative Pressure Relief Valve	1	00-1800	Medical e-Shop	1.04
21	Solenoid Valve	1	2565N15	McMaster-Carr	133.15
22	Proportional Valve	1	AP-7211-QW2-U711-OX2	Cowper Inc.	190.24
23	Pneumatically Actuated Valve	1	6124K401	McMaster-Carr	76.11
24	Flow Meter	1	SFM3300-AW	Mouser	263.37
	Mask Pressure Transducer	1	SSCDRRN 001PDAA5	Digikey	95.86
	Supply Pressure Transducer	1	SSCDANN 100PAAA5	Mouser	56.39
25	Pressure Regulator	1	6763K13	McMaster-Carr	48.11
	Proportional Valve Control Board	1	130-222	Cowper Inc.	152.00
	Huzzah32	1	N/A	Adafruit	21.95
	Raspberry Pi 4 B+ 8GB	1	N/A	Adafruit	104.95
	Solid State Relay	1	DC60S3	Mouser	38.49
	Voltage Step-down	1	LM2596	Amazon	7.16
	Main Power Supply	1	1470-3098-ND	Digikey	35.42
	Jumper Cables M/M	1	1956	Adafruit	1.95
	Jumper Cables F/M	1	1953	Adafruit	1.95

	Jumper Cables F/F	1	1951	Adafruit	1.95
	Bread Board	2	EL-CP-003	Amazon	5.00
	Waterproof DC Power Cable Set - 5.5/2.1mm	1	743	Adafruit	2.50
	Micro-USB to Regular USB	1	592	Adafruit	2.95
	DIN Connector for PV	1	122-800EX	Cowper Inc.	10.97
	Connector for Flow Meter	1	EK-F3x-CAP	Mouser	132.27
	Pipe Joint Tape	1	5495-85	Home Hardware	2.00
	Pipe Joint Compound	1	3210-016	Home Hardware	23.99
	Bulged Stainless Steel Tubulation	1	TUBN-063-2"	Scanivalve	0.00
	1/8 NPTM Plug with Hex Head	1	5232T449	McMaster-Carr	4.90
26	Inspiratory and Expiratory tubing with check valves	1	N/A	St. Clair College	0.00
27	Tubing quick connect into 1/4 NPTM	1	52115K203	McMaster-Carr	17.65

A.3 Software

Since most projects reviewed in the literature review did not include the software to go along with the hardware, in the below sections are the programming scripts used in the operation of the ventilator testing. There is a main python script that includes the main functionalities of the ventilator, there is the main C++ script for the Huzzah32, and there is an emergency python script that closes PV in the case the main script failed. The packages used in the main python script are displayed in Table A-2 below, and the programming scripts are in the sections below.

Table A-2: Python packages used

Name	Version	Use
pyserial	3.5	Serial communication
numpy	1.21.2	Array handling
time	N/A	General use
pandas	1.4.1	Reading .csv data
math	N/A	General use
matplotlib.pyplot	3.5.0	Generating plots
csv	N/A	Saving data to csv
RPi.GPIO	0.7.1	Raspberry Pi GPIO control
scipy.signal	1.7.3	Mask pressure filtering

A.3.1 Main Python Script

```
#
    ##----------------------------------------------------------------------------
     # Name: Final_Ventilator_Scipt.py # Purpose: Main script
    to operate the ventilator on the raspberry pi # # Author:
    Ryan Wardell # Created: 2022-12-01
```

```
#-----------------------------------------------------------------
 # Imported packages import serial import numpy as np
import time import pandas as pd import math import
matplotlib.pyplot as plt import csv import RPi.GPIO as GPIO
 from scipy import signal
# Global variables for the two serial connections global ser #
 For SP-1 global ser2 # For Huzzah32
#
   ---------------------------------------------------------------
 # Ventilator Settings BPM = 20 iTime = 1 itime_step =
0.025 etime_step = 0.025 numBreaths = 5 yref = 12 # [L/min]
 BAP = 5.5 # 5 cmH2O to psi
eTime = (60/BPM) - iTime iIter = math.floor(iTime/itime_step)
   eIter = math.floor(eTime/etime_step)
# For calculating flow during expiratory cycle rho = 1.225 #
   gas density mu = 1.83e-5 # kinematic viscosity L_exp = 1.27
    # expiratory piping length D_exp = 0.03 # expiratory
   piping diameter 10mm tubing from mask exp_area = (math.pi*(
   D_exp**2))/4 # expiratory flow area used in flow rate
   calculation
# ESC parameters phase = 0 # zero K = 1 # integration gain k_a
    = 0.1; k_b = 0.001; amp = 0.1 omega = 40
# Low-pass filter for ESC samplingFreq = (1/itime_step)/(2*np.
   pi) # 1/inspiratory time step /2 pi to convert to Hz w0 =
   2*np.pi*0.2 # pole frequency (rad/s) AKA cutoff frequency
   num = w0 # transfer function numerator coefficients den =
   [1,w0] # transfer function denominator coefficients lowPass
    = signal.TransferFunction(num,den) # Transfer function dt
   = 1.0/samplingFreq discreteLowPass = lowPass.to_discrete(dt
   ,method='gbt',alpha=0.5) b = discreteLowPass.num a = -
   discreteLowPass.den
# WMA for pressure signal wmin = 0.1 nw = 5 Z = -np.log(wmin)
   /(nw-1) w = np.zeros(nw) for i in range(0,5): w[i] = np.exp
   (i*np.log(wmin)/(nw-1)) w = w / np.sum(w)
   #---------------------------------------------------------------------
    # Initialize Serial Ports
# Open Serial Port for SP-1 ser = serial.Serial( port = '/dev/
   ttyUSB1', #Device name baudrate=115200, #baudrate bytesize=
   serial.EIGHTBITS, #number of databits parity=serial.
   PARITY_NONE, #enable parity checking stopbits=serial.
   STOPBITS_ONE, #number of stopbits timeout=1, #set a timeout
    value (example only because reset #takes longer) xonxoff
   =0, #disable software flow control rtscts=0, #disable RTS/
   CTS flow control )
# Specify the address of the RS485 adapter cable ADDRESS = 0
# Set up the serial port to the ESP32 ser2 = serial.Serial(
   port = '/dev/ttyUSB0', #Device name baudrate=9600, #
   baudrate bytesize=serial.EIGHTBITS, #number of databits
   parity=serial.PARITY_NONE, #enable parity checking stopbits
   =serial.STOPBITS_ONE, #number of stopbits timeout=1, #set a
    timeout value (example only because reset #takes longer)
   xonxoff=False, #disable software flow control rtscts=False,
    #disable RTS/CTS flow control )
#
   ----------------------------------------------------------------------
    # GPIO Setup GPIO.setmode(GPIO.BOARD) # Use the board
   numbering system for GPIO pinNum = 11 # What GPIO pin are
   we using GPIO.setup(pinNum,GPIO.OUT) # Initialize the pin
   to be an output
#
   ----------------------------------------------------------------------
    # SP-1 Functions
def compute_SHDLC_checksum(listofbytes): # sum up all bytes
   tmpchecksum = sum(listofbytes) # take least significant
   byte tmpchecksum = tmpchecksum & 0xff # invert (bit-wise
   XOR with 0xff) tmpchecksum = 0xff ^ tmpchecksum return
   tmpchecksum
def byte_stuff(listofbytes): i=0 while i<len(listofbytes): if
   listofbytes[i]==0x7e: listofbytes[i]=0x7d listofbytes.
   insert(i+1,0x5e) i+=1 elif listofbytes[i]==0x7d:
   listofbytes[i]=0x7d listofbytes.insert(i+1,0x5d) i+=1 elif
   listofbytes[i]==0x11: listofbytes[i]=0x7d listofbytes.
   insert(i+1,0x31) i+=1 elif listofbytes[i]==0x13:
   listofbytes[i]=0x7d listofbytes.insert(i+1,0x33) i+=1 i+=1
   return listofbytes
def make_and_send_SHDLC_command(address, commandID, data):
   datalength = len(data) # compose command command = [address
   , commandID, datalength] + data # compute checksum command.
   append(compute_SHDLC_checksum(command)) # do byte stuffing
   command = byte_stuff(command) # add start byte and stop
   byte command = [0x7e] + command + [0x7e] # convert list of
   numbers to bytearray command = bytearray(command) # send
   command to the device ser.write(command) def
   read_SHDLC_response(): response = np.zeros(15) res = ''
   count = 0 # Iterate read until res is empty or stop byte
   received firstbyte=True while True: res = ser.read(1) if
   not res: break elif firstbyte or (ord(res) != 0x7e):
```

```
firstbyte = False response[count] = ord(res) count+=1 else:
 response[count] = ord(res) count+=1 break # remove first
element (the start byte) from the response response[0] = 0
# remove the last element (the stop byte) form the response
 response[-1] = 0 # Check for bytes that are stuffed i=0 #
loop through response list while i<len(response): if
response[i] == 0x7D: # 0x7d marks stuffed bytes. see SHDLC
documentation if response[i+1] == 0x5E: response[i] = 0x7E
elif response[i+1] == 0x5D: response[i] = 0x7D elif
response[i+1] == 0x31: response[i] = 0x11 elif response[i
+1] == 0x33: response[i] = 0x13 i+=1 # return only the '
data' portion of the response return response[4:7]
#
------------------------------------------------------------------------
 # Reading Flow Meter def read_spiro(): while True: data =
[] while len(data)<2: make_and_send_SHDLC_command(ADDRESS,
0x35, []) data = read_SHDLC_response() if data[0] > 0:
break # Combine the first two data bytes into one 16bit
data value value_from_sensor = data[1]*256 + data[2]
# Compute two's complement (handle negative numbers!) if
value_from_sensor >= 2**15: # 2**15 = 32768
value_from_sensor = value_from_sensor-2**16 # 2**16 = 65536
 actual_value = (value_from_sensor+32768)/120 # Note this
value is in sl/min return actual_value
#-----------------------------------------------------------------------
 # Reading Pressure Transducer Functions def read_esp32():
esp_in = '0000' # Read data from the ESP32 twice while True
: if len(esp_in)<17 and len(esp_in)>13: if esp_in[0] == '!'
 and esp_in[-1] == '!': break else: esp_in = '0000' # end
```

```
of nested-if else: esp_in = ser2.readline().decode('ascii')
.rstrip() # get rid of start and stop byte size = len(
esp_in) esp_in = esp_in[1:size-1] # intialize local
variables ps1_str = '' ps0_str = '' psup_str = '' character
 = '' place = 1 count = 0 while count < len(esp_in):
character = esp_in[count] if (character == ":"): place =
place + 1 count = count + 1 else: if place==1: ps1_str =
ps1_str + character count = count + 1 elif place==2:
ps0_str = ps0_str + character count = count + 1 else:
psup_str = psup_str + character count = count + 1 # end of
nested if-elif-else # end of if-else # end of while loop
ps1_in = int(ps1_str) ps0_in = int(ps0_str) psup_in = int(
psup_str) # Format 0-4095 to 0-3.3V ps1_pin = (ps1_in/4095)
*3.3 ps0_pin = (ps0_in/4095)*3.3 psup_pin = (psup_in/4095)
*3.3 #Correct voltages for esp32 pin offsets ps1_pinCorr =
0.8878*ps1_pin + 0.321 # Pin 12 ps0_pinCorr = 0.8955*
ps0_pin + 0.39 # Pin 32 psup_pinCorr = 0.8948*psup_pin +
0.22 # Pin 33 # Correct for the voltage division used
ps1_sensor = ps1_pinCorr/0.7538 ps0_sensor = ps0_pinCorr
/0.7292 psup = psup_pinCorr/0.6059 #print(psup) # Convert
the voltage to pressure in psi -- equation from pressure
transducer manual ps1_psi = (ps1_sensor - 0.1*psup) *
((2.0)/(0.8*psup)) - 1.0 ps0_psi = ((ps0_sensor-(0.1*psup))
*((100.0)/(0.8*psup))) - 0.0 # Convert ps1 from psi to
cmH20 ps1_cmh20 = 70.31*ps1_psi return [ps1_cmh20,ps0_psi]
#
 ------------------------------------------------------------------------
 # Convert unew to esp32in def createInput(input_wanted):
terminate_term = '\n' # Termination byte for serial
```

```
    handling board_resistance = 185.3 current_wanted = ((20-4)*
    input_wanted) + 4 voltage_wanted = (current_wanted/1000)*
    board_resistance voltage_percent = voltage_wanted/3.3
    input_to_esp32 = round(voltage_percent*255)
# Saturate the input to be between 0 and 255 if input_to_esp32
     < 0: input_to_esp32 = 0 elif input_to_esp32 > 255:
    input_to_esp32 = 255
to_esp32 = str(input_to_esp32) + terminate_term return
    to_esp32
#
    ----------------------------------------------------------------------
     # Find the initial guess for ESC using the valve
    characteristic's and supply pressure
# READ OUT 20 VALUES AND AVERAGE THEM FOR PRESSURE READING z =
     0 z_count = 50 sup_p_measurements = np.zeros(z_count)
    while z < z_count: esp32_data = read_esp32()
    sup_p_measurements[z] = esp32_data[1] # in units of psi z
    +=1
sup_p_mean = np.mean(sup_p_measurements) print('mean␣supply␣
    pressure␣=␣',sup_p_mean) supply_pressure_barg = (sup_p_mean
    -14.69595)/14.503773773
# Y axis is Flow rate in [Normal Litres per minute] # X axis
    is %FS of valve opening 0-100 = 4-20mA slope = -0.1105*(
    supply_pressure_barg**2) + 0.8058*(supply_pressure_barg) +
    0.5309 x_intercept = -14.176*(supply_pressure_barg) +
    91.638 y_intercept = -slope*x_intercept
just_closed = (x_intercept - 0)/100 # this will need adjusting
initial_guess_x = (yref-y_intercept)/slope +12 # This gives us
     the intial %FS from 0-100 or 4-20mA print('initial valve
    guess',initial_guess_x)
# Initial guessing stuff #just_closed = 0.42 unew =
    initial_guess_x/100 # intial k input yout = 0 # initial
    flow rate output (zero since initial input k is zero) ycost
     = abs(yout - yref) # Initial cost function output uhat =
    unew
#
    ------------------------------------------------------------------
     # Start-up Commands for Flow Meter
# Set resolution make_and_send_SHDLC_command(ADDRESS, 0x41,
    [14]) read_SHDLC_response()
# Clear measurement buffer make_and_send_SHDLC_command(ADDRESS
    , 0x36, [2]) read_SHDLC_response()
# Start continuous measurement make_and_send_SHDLC_command(
    ADDRESS, 0x33, [0,0]) read_SHDLC_response()
#
    ------------------------------------------------------------------
     # Initialize For Main Loop and LPFs i = 0 volume = 0 xi =
    0 LPF = 0 p_mask_read = 0 # current p_mask_1 = 0 # n-1
    p_mask_2 = 0 # n-2 p_mask_3 = 0 # n-3 p_mask_4 = 0 # n-4
    tidal_vol = np.zeros(numBreaths) minute_vol = np.zeros(
    numBreaths) PEEP = BAP
#
    ------------------------------------------------------------------
     # Send command to the valve just so the valve is closed
    to_esp32 = createInput(just_closed) ser2.write(to_esp32.
    encode()) time.sleep(2) # sleep for 2 seconds
#
    ------------------------------------------------------------------
```

```
 # Open the valve to the initial guess to_esp32 =
createInput(unew) ser2.write(to_esp32.encode())
#
-----------------------------------------------------------------------

slow_ESC = False # Main Loop with open('ISO_TEST_8.csv','w',
newline='') as csvfile: fieldnames = ['Time','Flow Rate','
unew','Mask Pressure','Time Difference','Volume','Raw Mask
Pressure','PEEP'] writer = csv.DictWriter(csvfile,
fieldnames = fieldnames) writer.writeheader() while i<
numBreaths: # Run for a specific number of breaths ser2.
reset_input_buffer() ser2.reset_output_buffer() j = 0 #
inspiratory counter time_diff = 0 GPIO.output(pinNum,0) #
Close expiratory line time.sleep(0.001) is_filter = False
vol_max = -100 # reset max insp volume to large negative
number yout = 0 # Starting inspiratory Volume
start_insp_vol = volume while j<iIter: # Run loop for
specific number of iterations #Start timer start = time.
time_ns()
#----------------------------------------------------------
 # Read Huzzah32 outputs p_mask_4 = p_mask_3 p_mask_3 =
p_mask_2 p_mask_2 = p_mask_1 p_mask_1 = p_mask_read [
p_mask_read,p_sup] = read_esp32() if j>4: p_mask_5MA = (
p_mask_read + p_mask_1 + p_mask_2 + p_mask_3 + p_mask_4)/5
# Moving average p_mask_5WMA = (p_mask_read*w[0] + p_mask_1
*w[1] + p_mask_2*w[2] + p_mask_3*w[3] + p_mask_4*w[4]) #
Weighted moving average else: p_mask_5MA = p_mask_read
p_mask_5WMA = p_mask_read # end of if-else yout_old = yout
# Read flow rate y_read = read_spiro() # unit is in sl/min
```

```
if y_read>100: yout = yout else: yout = y_read print('Flow
rate (sl/min)',yout) # Create cost function ycost = abs(
yout- yref) # Create cost function that uses [L/min] #print
('y cost',ycost)
# Time t = time.time() # Assign old values of LPF and xi for
the Low Pass Filtering xiOLD = xi LPFOLD = LPF # ESC Dither
 Multiplication xi = ycost*k_a*amp*math.sin(omega*t + phase
) # Apply LPF LPF = a[1]*LPFOLD + b[0]*xi + b[1]*xiOLD #
Initial value for integration if j == 0: uhat = uhat else:
uhat = uhat + LPF*K*itime_step # Add Dither Signal unew =
uhat + k_b*amp*math.sin(omega*t + phase) #unew = 0.70 #
Send new input to valve to_esp32 = createInput(unew) ser2.
write(to_esp32.encode()) # Tracking first and last values
if j==0: print('First value',unew) elif j==(iIter-1): print
('Last value',unew) # end of if # Calculate volume using
trapezoidal rule del_vol = itime_step*0.5*(yout-yout_old)
volume = volume + del_vol if volume>vol_max: # keeps the
highest volume achieved as the tidal volume vol_max =
volume # end of if # Write to CSV file writer.writerow({'
Time':t,'Flow Rate':yout,'unew':unew,'Mask Pressure':
p_mask_5WMA,'Time Difference':time_diff,'Volume':volume,'
Raw Mask Pressure':p_mask_read,'PEEP':PEEP})
#--------------------------------------------------------------------
 # End timer and sleep finish = time.time_ns() time_diff =
(1e-9*(finish - start)) if (itime_step-time_diff) < 0:
time_diff = itime_step time.sleep(itime_step-time_diff) j
+=1 # End of inspiratory while loop # Close the inspiratory
 valve to_esp32 = createInput(just_closed) ser2.write(
to_esp32.encode()) time.sleep(0.005) tidal_vol[i] = vol_max
```

```
 - start_insp_vol # Calculate Tidal Volume minute_vol[i] =
tidal_vol[i]*BPM k = 0 # expiratory counter time_diff = 0
peep_1 = 0 flow = 0 GPIO.output(pinNum,1) # Open expiratory
 line while k<eIter: # Run loop for specific number of
iterations start = time.time_ns()
#-----------------------------------------------------------------
 # Read Huzzah32 outputs p_mask_4 = p_mask_3 p_mask_3 =
p_mask_2 p_mask_2 = p_mask_1 p_mask_1 = p_mask_read [
p_mask_read,p_sup] = read_esp32() if j>4: p_mask_5WMA = (
p_mask_read*w[0] + p_mask_1*w[1] + p_mask_2*w[2] + p_mask_3
*w[3] + p_mask_4*w[4]) # Weighted moving average else:
p_mask_5WMA = p_mask_read # end of if-else if k>(eIter-3):
peep_2 = peep_1 peep_1 = p_mask_5WMA # Calculate flow and
volume from pressure difference flow_old = flow temp =
((32*mu*L_exp)/(D_exp**2)) temp2 = ((-temp + math.sqrt((
temp**2)+(2*rho*((p_mask_5WMA-PEEP)/70.307))))/rho) flow =
-exp_area*temp2 flow = 60000*flow # units from m^3/s to L/
min print('flow rate [sl/min]',flow) t = time.time() #
Calculate volume using trapezoidal rule del_vol =
etime_step*0.5*(flow-flow_old) volume = volume + del_vol if
 volume<0: volume = 0 # Write to CSV file writer.writerow
({'Time':t,'Flow Rate':flow,'unew':just_closed,'Mask
Pressure':p_mask_5WMA,'Time Difference':time_diff,'Volume':
volume,'Raw Mask Pressure':p_mask_read,'PEEP':PEEP})
#-----------------------------------------------------------------
 finish = time.time_ns() time_diff = (1e-9*(finish - start)
) if (etime_step-time_diff) < 0: time_diff = etime_step
time.sleep(etime_step-time_diff) k+=1 # End of expiratory
while loop PEEP = (peep_1+peep_2)/2 i+=1 # Increase breath
```

```
    counter by 1 # End of number of breaths while loop
    #------------------------------------------------------------
     # Closing Commands for Flow Meter and Huzzah32
# stop continuous measurement make_and_send_SHDLC_command(
    ADDRESS, 0x34, []) read_SHDLC_response()
closestr = '0\n' ser2.write(closestr.encode()) # close the
    valve
    #------------------------------------------------------------
     # Post Processing Attached
    #------------------------------------------------------------
     # Read using Pandas data = pd.read_csv('/home/pi/Downloads
    /ISO_TEST_8.csv', sep = ',')
data_array = data.to_numpy()
Time = data_array[:,0] Flow_Rate = data_array[:,1] unew =
    data_array[:,2] Mask_Pressure = data_array[:,3]
    Time_Difference = data_array[:,4] lung_volume = data_array
    [:,5] Raw_Mask_Pressure = data_array[:,6]
#
    -------------------------------------------------------------
     # Post Processing
# Create an array that has the actual time size = np.shape(
    Time)[0] actual_time = np.zeros(size,float) time_step = np.
    zeros(size,float) for i in range(size): if i == 0:
    actual_time[i] = 0.0 time_step[i] = 0.0 else: time_step[i]
    = (Time[i]-Time[i-1]) actual_time[i] = actual_time[i-1] +
    time_step[i]
# What is the maximum time difference? print('Slowest time
    step is: ',Time_Difference.max())
```

```
# What is the tidal volume and minute volume for each breath?
    print('The theoretical tidal volume of each breath was:',(
    yref/60)*iTime,'L') print('The theoretical minute volume of
     each breath was:',(yref/60)*iTime*BPM,'L/min') for i in
    range(numBreaths): print('Tidal volume of breath #',i+1,'
    was ',round(tidal_vol[i],4),'L') print('Minute volume of
    breath #',i+1,' was ',round(minute_vol[i],4),'L/min')
# Plot unew with respect to actual time plt.figure(1) plt.plot
    (actual_time,unew) plt.xlabel('Time [s]') plt.ylabel('unew
    [0 to 1]') plt.title('unew versus time')
# Plot Flow rate with respect to actual time plt.figure(2) plt
    .plot(actual_time,Flow_Rate) plt.xlabel('Time [s]') plt.
    ylabel('Flow rate [sL/min]') plt.title('Flow rate versus
    Time')
# Plot Volume with respect to actual time plt.figure(3) plt.
    plot(actual_time,lung_volume) plt.xlabel('Time [s]') plt.
    ylabel('Lung Volume [L]') plt.title('Lung Volume Waveform')
# Plot Mask Pressure with respect to actual time plt.figure(4)
     plt.plot(actual_time,Raw_Mask_Pressure,label='Raw',color =
     'k') plt.plot(actual_time,Mask_Pressure,label='Filtered',
    color = 'r') plt.xlabel('Time [s]') plt.ylabel('Pressure [
    cmH2O]') plt.title('Lung Pressure Waveform') plt.legend()
```

A.3.2 Huzzah32 Script

```
/* Arduino IDE Code for ESP32 esp32_conmplete.cpp Ventilator
    Project Ryan Wardell */
#define DAC1 25 #include <string>
int voltVal = 0; // 0-255 corresponding to 0-3.3V const
```

```
    unsigned int MAX_MESSAGE_LENGTH = 12; // Maximum length
    able to be read of a single message
const int ps1Pin = 12; // PS-1 ADC input pin const int ps0Pin
    = 32; // PS-0 ADC input pin const int powerPin = 33; //
    Power Supply ADC input pin
int ps1Val = 0; int ps0Val = 0; int pSupVal = 0; String
    out_string = "0";
// Setup code void setup() { Serial.begin(9600); // // Opens
    the serial port and sets data rate to 9600 bps
}
// Main code void loop() { // Read PV-1 Input Command from Pi
    while (Serial.available() > 0) { //Create a place to hold
    the incoming message static char message[MAX_MESSAGE_LENGTH
    ]; static unsigned int message_pos = 0;
//Read the next available byte in the serial receive buffer
    char inByte = Serial.read();
//Message coming in (check not terminating character) and
    guard for over message size if ( inByte != ’\n’ && (
    message_pos < MAX_MESSAGE_LENGTH - 1) ) { //Add the
    incoming byte to our message message[message_pos] = inByte;
     message_pos++; } //Full message received... else { //Add
    null character to string message[message_pos] = ’\0’;
//Convert to integer int voltVal = atoi(message);
dacWrite(DAC1,voltVal); // Output a voltage to channel 1
//Reset for the next message message_pos = 0;
} } // Read PS0, PS1, and pressure transducer’s power supply
    voltage ps1Val = analogRead(ps1Pin); // Read patient mask
    pressure ps0Val = analogRead(ps0Pin); // Read upstream
    supply pressure pSupVal = analogRead(powerPin); // Read
    transducer power supply voltage out_string = '!' + String(
    ps1Val) + ':' + String(ps0Val) + ':' + String(pSupVal) +
    '!'; // Format output to Pi Serial.println(out_string); //
    Print out whole pressure value as a line to the Pi }
```

A.3.3 Emergency Valve Close Script

```
#!/usr/bin/env python3 # -*- coding: utf-8 -*- """ Created on
    Tue Oct 25 23:17:12 2022
@author: pi """ import serial global ser2 # For Huzzah32 # Set
     up the serial port to the ESP32 ser2 = serial.Serial( port
     = '/dev/ttyUSB1', #Device name baudrate=9600, #baudrate
    bytesize=serial.EIGHTBITS, #number of databits parity=
    serial.PARITY_NONE, #enable parity checking stopbits=serial
    .STOPBITS_ONE, #number of stopbits timeout=1, #set a
    timeout value (example only because reset #takes longer)
    xonxoff=0, #disable software flow control rtscts=0, #
    disable RTS/CTS flow control )
closestr = '0\n' ser2.write(closestr.encode()) # close the
    valve print('valve is closed') ser2.close()
```

Appendix B

Complete Amesim model

The complete Amesim model with a break-down of components and their location within the model is displayed. There are only 32 individual Amesim components needed to create the entire ventilator, control scheme, and lung model. A list of these components can be found in Table B-1 where each part is listed with their respective submodels and the quantity of that part used in the model.

As explained in Section 3.3.1, for the piping sections and piping joints there are multiple submodels used due to complilation issues in Amesim. All of the pneumatic component submodels account for compressibility and frictional effects. In the table each part has a given part number from 1 to 32 such that their location in the model can be illustrated in an image of the entire Amesim model as depicted in Figure B-1.

Using the information from within this thesis, one could easily recreate this model in Amesim.

Table B-1: All Amesim components

Part No.	Part description	Amesim part name(s)	Quantity
1	Pipe section	PNPC0, PNPC1, PNPC2, PNPC4	7, 2 ,9 ,3
2	Pipe elbow	PNBP001	1
3	Pipe t-joint	PN3P000, TPTE001	3, 3
4	2-port proportional valve	PNTV01	4
5	3-port proportional valve	PNPV001	1
6	Pressure regulator	PNPR12	2
7	Pressure relief valve	PNRV00	2
8	Supply pressure	PNCS001	1
9	Flow sensor	PNQS02	2
10	Pressure sensor	PNPS001	1
11	Lung model feedback	PNVS001	1
12	Solenoid valve	PNSV231_04	1
13	Atmosphere	PNAS001	4
14	Plug	PNPL01	1
15	Sampler	ZOH00	2
16	Switch	SWITCH01	8
17	Gain	GA00	16
18	Discrete state-space	SIGDISCSSP	1
19	Demux	DYNDMUX2	1
20	Sink	SSINK	6
21	Summation	JUN3P	3
22	Constant	CONS00	14
23	2-way signal split	SPLT0	7
24	Clock	CLOC	1
25	Square wave	SQW00	4
26	Subtraction	JUN3M	4
27	4-way signal split	SPLT1	2
28	Mulitplication	MUL00	1
29	Discrete integration	DTI02	1
30	Sin wave	SIN0	1
31	Saturation	SAT0	1
32	Signal comparator	SIGCOMP0	1

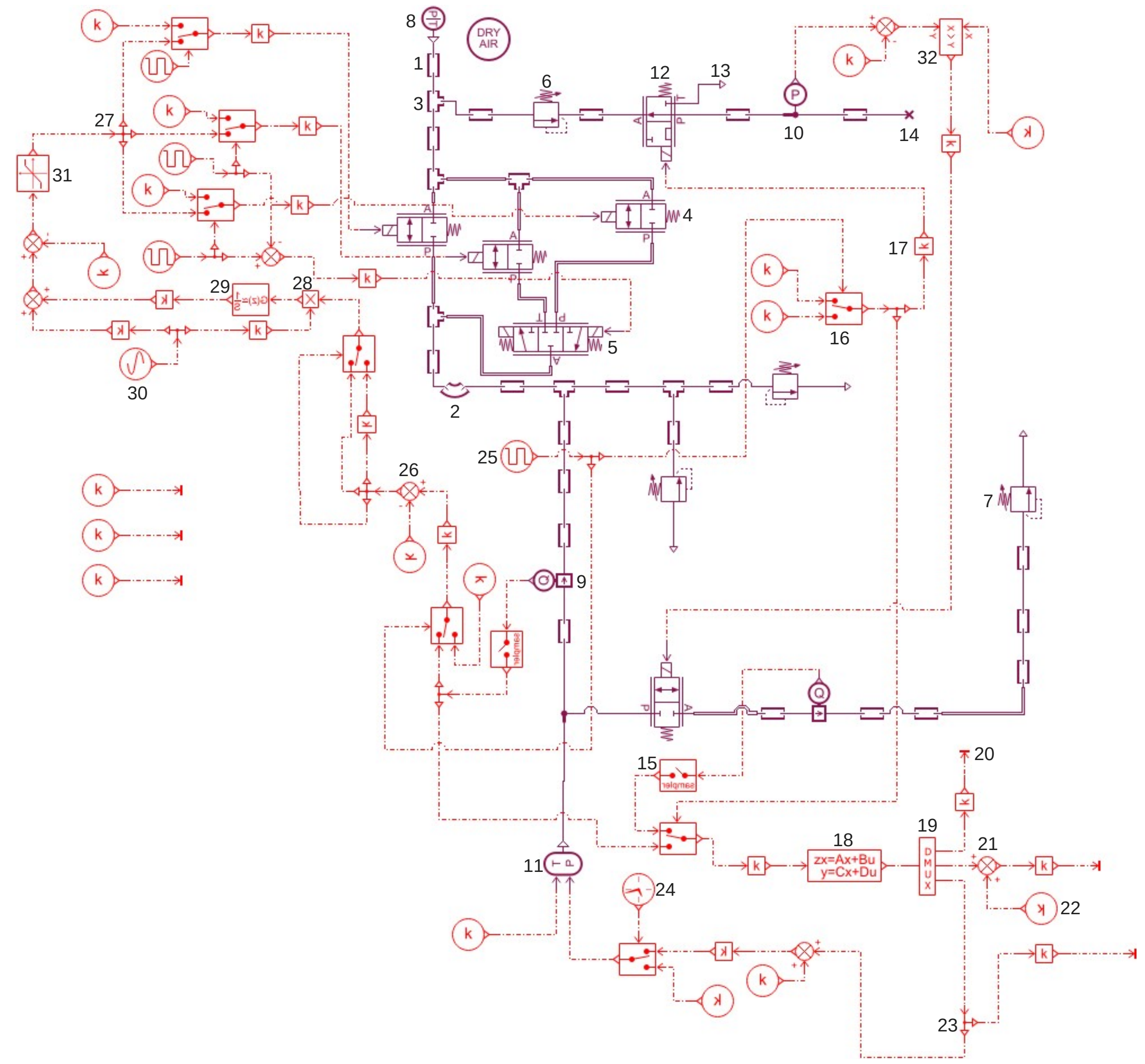

Figure B-1: Complete Amesim model

Vita Auctoris

Name:	Ryan K. Wardell
Place of Birth:	Windsor, Ontario, Canada
Year of Birth:	1998
Education:	M.A.Sc. in Mechanical Engineering University of Windsor, 2021-2023
	B.A.Sc. in Mechanical Engineering with Aerospace Option University of Windsor, 2016-2020