Universidad de Cuenca

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones.

Autores :

Paúl Gustavo Carrión Portilla — C.I. 010483800-8
(pcportilla@gmail.com)

David Andrés Montalván Astudillo — C.I. 010427313-1
(dmontast@gmail.com)

Director :

Ing. Luis Ismael Minchala Ávila, PhD — C.I. 030145348-6

Co-Director :

Ing. Darwin Fabián Astudillo Salinas, PhD — C.I. 010390703-6

Cuenca-Ecuador

5 de Noviembre de 2021

Resumen

A mediados de diciembre de 2019, en la ciudad de Wuhan, China, se reportan los primeros casos del virus respiratorio infeccioso COVID-19. El COVID- 19, según la Organización Mundial de la Salud (WHO), es una enfermedad respiratoria infecciosa causada por un virus perteneciente a la gran familia CoV (Coronavirus). Los síntomas varían su intensidad para cada ser humano, desde un cuadro clínico leve que no requiere cuidados específicos, hasta síntomas severos que pudiesen conllevar una serie de enfermedades graves que necesiten tratamiento respiratorio intensivo, y en el peor de los casos la muerte.

Toda esta situación obliga a que el personal médico tenga que decidir si los pacientes son aptos para acceder a los distintos tratamientos de soporte vital de acuerdo a la "gravedad e intensidad de los recursos", entre ellos el uso de ventiladores mecánicos.

No obstante, la creación de nuevos dispositivos médicos para aliviar la saturación, es posible, es por esto que la construcción de ventiladores portátiles de bajo costo se presenta como una solución importante y factible.

En este trabajo, se presenta el proceso para la implementación de un modelo de ventilador mecánico asistido. El modelo fue desarrollado con base en la investigación recopilada durante la pandemia por médicos italianos y el MIT. Este ventilador permite que pacientes con síntomas menos severos de Covid-19 puedan ser atendidos por especialistas de la salud menos experimentados. Todo esto ayuda mientras el resto de recursos es usado en pacientes con cuadros clínicos severos.

En la implementación se obtuvo un funcionamiento satisfactorio del dispositivo además de haber logrado el abaratamiento de costos en su construcción. Se logró la compresión de la bolsa de resucitación manual mediante la variación del volumen tidal, la cual está diseñada para variar en un rango entre 100 y 800 ml. Así mismo, el modo asistido tiene mayor precisión para valores de volumen tidal entre 400 y 500 ml y para un RR de 1.2 Brpm.

Palabras clave : Covid-19. Ventilador. Mecánico. Asistido

Abstract

In mid-December 2019, in the city of Wuhan, China, the first cases of the infectious respiratory virus COVID-19 were reported. COVID-19, according to the World Health Organization (WHO), is an infectious respiratory disease caused by a virus belonging to the large CoV family (Coronavirus). The symptoms vary in intensity for each human being, from a mild clinical picture that does not require specific care, to severe symptoms that could lead to a series of serious diseases that require intensive respiratory treatment or in the worst cases, death.

This situation forces the medical staff to decide if patients are suitable to access the different life support treatments according to the "severity and intensity of the resources", including the use of mechanical ventilators.

However, the creation of new medical devices to alleviate this saturation is possible, which is why the construction of low-cost portable ventilators is presented as an important and feasible solution.

This document shows the process followed in order to build an assisted mechanical ventilator model. The model was developed based on the research collected during the pandemic by Italian physicians and MIT. This ventilator allows patients with less severe Covid-19 symptoms to be cared for by less experienced healthcare specialists. All this helps while the rest of the resources are used in patients with severe clinical case.

In the implementation, a satisfactory operation of the device was obtained in addition to having reduced costs in its construction. Compression of the manual resuscitation bag was achieved by varying the tidal volume, which is designed to vary in a range between 100 and 800 ml. Likewise, the assisted mode has greater precision for tidal volume values between 400 and 500 ml and a RR of 1.2 Brpm.

Keywords : Covid-19. Ventilator. Mechanic. Assisted

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Yo, Paúl Gustavo Carrión Portilla, autor del trabajo de titulación "Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 5 de noviembre de 2021

Paúl Gustavo Carrión Portilla
010483800-8

Cláusula de Propiedad Intelectual

Yo, David Andrés Montalván Astudillo, autor del trabajo de titulación "Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 5 de noviembre de 2021

David Andrés Montalván Astudillo
010427313-1

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Paúl Gustavo Carrión Portilla en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 5 de noviembre de 2021

Paúl Gustavo Carrión Portilla
010483800-8

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, David Andrés Montalván Astudillo en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 5 de noviembre de 2021

David Andrés Montalván Astudillo
010427313-1

Certifico

Que el presente proyecto de tesis: Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida, fue dirigido y revisado por mi persona.

Digitally signed by LUIS ISMAEL MINCHALA AVILA
Date: 2021.10.29 15:37:37 -05'00'

Ing. Luis Ismael Minchala Ávila, PhD
Director

Certifico

Que el presente proyecto de tesis: Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida, fue dirigido y revisado por mi persona.

Firmado digitalmente por DARWIN FABIAN ASTUDILLO SALINAS
Fecha: 2021.10.29 18:21:19 -05'00'

Ing. Darwin Fabián Astudillo Salinas, PhD
Co-director

Dedicatoria

Charles Bukowski en su primer novela escribió ”Este libro no va dedicado a nadie”. Snoop Dog en el discurso de aceptación de su estrella en el Paseo de la Fama en Hollywood se dedicó todo su éxito a sí mismo. Yo tan solo espero que esto no se pierda como los libros de la biblioteca de Alejandría, porque esto no es una novela y tampoco tengo éxito.

A todo aquel que lea, sea empático, buena persona y procure aportar a la humanidad. Ojalá y lo haga basado en el conocimiento que aquí se comparte.

Paul Gustavo Carrión Portilla

Dedicatoria

La vida universitaria me ha brindado una gran cantidad de conocimientos, me ha enseñado el valor del esfuerzo, la dedicación, la superación, pero sobre todo levantarme de los tropiezos y seguir adelante. Durante esta experiencia he tenido a mi lado de manera incondicional a lo más grande que tengo en el mundo, mi familia, especialmente mis papás y hermanas, que han moldeado mi ser, me han tomado de la mano y me han llevado al lugar en donde me encuentro ahora, solo puedo decirles que Dios les pague y que les quiero con toda mi alma.

De igual manera, he recibido la ayuda incondicional de mis abuelos, primos y tíos, sin ellos el camino hubiera sido más cruento, gracias por estar a mi lado y por todo lo que me han dado.

De manera especial dedico este trabajo a mi papá, Marco Montalván, que desde el cielo me cuida, gracias por haber sido mi héroe y el pilar de mi vida. Con el mismo cariño dedico este trabajo a mi mamá, Sonia Astudillo, que está a mi lado, gracias a su amor y cuidados he llegado tan lejos. Todos los frutos de este largo esfuerzo son para ustedes dos, que Dios les bendiga, nunca se separen de mi lado.

David Andrés Montalván Astudillo

Agradecimientos

Agradecemos a nuestros padres por dedicarnos su tiempo y cariño. A nuestros hermanos y hermanas por compartir vivencias en todos estos años. A nuestros abuelos por darnos su apoyo y comprensión. A nuestros tíos y primos por estar cuando los necesitamos, a la familia entera, muchas gracias.

Agradecemos a la Universidad de Cuenca, por darnos el lugar para perseguir nuestros sueños, a nuestros profesores por darnos las herramientas para forjar un futuro y a los amigos de aula por el apoyo mutuo, a ustedes, muchas gracias.

Finalmente, queremos agradecer a nuestros directores de tesis, gracias por guiar nuestros pasos durante este trabajo de titulación, por la ayuda en momentos de incertidumbre y por su experiencia, a ustedes, muchas gracias.

Paul Gustavo Carrión Portilla
David Andrés Montalvan Astudillo

Capítulo 1

Introducción

Este capítulo presenta la identificación del problema, justificación y los objetivos del presente trabajo de titulación.

1.1. Identificación del problema

A mediados de diciembre de 2019, en la ciudad de Wuhan, China, se reportan los primeros casos del virus respiratorio infeccioso COVID-19. El COVID- 19, según la World Heal Organization (WHO), es una enfermedad respiratoria infecciosa causada por un virus perteneciente a la gran familia CoV (Coronavirus). Los síntomas varían su intensidad para cada ser humano, desde un cuadro clínico leve que no requiere cuidados específicos, hasta síntomas severos que pudiesen conllevar una serie de enfermedades graves que necesiten tratamiento respiratorio intensivo, y en el peor de los casos la muerte.

Según la WHO [10], a la fecha, globalmente existen 3.7 millones de personas contagiadas, 1.24 millones de personas recuperadas y 263 mil muertes. Considerando dichas cifras, es obvio que los sistemas de salud públicos y privados a nivel mundial, se están enfrentando a una crisis sin precedentes en tiempos modernos. Todo esto ha desatado una gran demanda en infraestructura, personal de la salud y en el uso de dispositivos médicos. Pero, debido a la incapacidad de satisfacer dicha demanda, la saturación en hospitales es una nueva realidad.

Toda esta situación obliga a que el personal médico tenga que decidir si los pacientes son aptos para acceder a los distintos tratamientos de soporte vital de acuerdo a la "gravedad e intensidad de los recursos" [11], entre ellos el uso de ventiladores mecánicos. No obstante, la creación de nuevos dispositivos médicos para aliviar la saturación, es posible, es por esto que la construcción de ventiladores portátiles de bajo costo se presenta como una solución importante y factible.

1.2. Justificación

Debido a la falta de recursos hospitalarios alrededor del mundo, el acceso a ventiladores para respiración asistida es reducido. Es por esto que la construcción de ventiladores de bajo costo ha sido una de las áreas en las cuales la comunidad científica se ha enfocado. El proyecto aquí presentado ha sido desarrollado para mitigar dicha carencia mediante la aplicación de patentes liberadas tales como el modelo PB-560 de la empresa India, Medtronic [12] [13] y el modelo de ventilación mecánica desarrollado por el MIT [14]. Se ha construido un ventilador con recursos de desarrollo disponibles en el medio, mejorando el costo sin la reducción de sus propiedades aplicativas.

1.3. Alcance

Este trabajo de tesis usa el modelo de ventilador diseñado, construido y compartido por el Instituto Tecnológico de Massachussets (MIT) como base de desarrollo [14]. Haciendo uso de los materiales disponibles en el medio, reduciendo el costo y sin disminuir sus configuraciones aplicativas, el ventilador ha sido construido solamente con el enfoque mecánico y electrónico.

Debido a nuestra formación académica, aspectos específicamente relacionados con el campo de la medicina serán obviados, centrándonos en su construcción y configuración técnica solamente. Las métricas usadas para su implementación y calibración se realizaron de acuerdo a los parámetros definidos por la guía CR501:2020 de la AAMI (Association for the Advancement of Medical Instrumentation) con variación experimental al COVID-19 [15].

1.4. Objetivos

1.4.1. Objetivo general

Construcción de un ventilador mecánico portátil de bajo costo, que permita la asistencia respiratoria a pacientes diagnosticados con COVID-19.

1.4.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Comprender del ámbito médico mediante la literatura correspondiente y usar ese conocimiento para la correcta construcción del ventilador.
- Manejar los diseños previos obtenidos en los proyectos de investigación antes mencionados para el posterior aporte de nuevas innovaciones al diseño.
- Implementar y calibrar el prototipo de acuerdo a los parámetros definidos por la guía CR501:2020 de la AAMI (Association for the Advancement of Medical Instrumentation) con variación experimental al COVID-19.
- *Benchmarking* del prototipo realizado con respecto a modelos anteriores pertenecientes al proyecto desarrollado por el Instituto Tecnólogico de Massachussets (MIT).

Capítulo 2

Fundamentos teóricos

2.1. Introducción

Este capítulo presenta una visión general sobre los antecedentes médicos, mecánicos y eléctricos relacionados con la respiración asistida. Así mismo, se presenta el estado del arte en cuanto a la tecnología disponible para la ventilación mecánica asistida.

2.2. Antecedentes médicos

La enfermedad respiratoria COVID-19 es parte de una extensa familia de virus denominados coronavirus, estos virus son conocidos por causar infecciones respiratorias que pueden tener sintomatología similar a la del resfriado común hasta problemas más graves atribuidos al síndrome respiratorio de Oriente Medio (MERS) o el síndrome respiratorio agudo severo (SARS).

De forma general, el período de incubación del virus se estima al rededor 12,5 días desde la exposición para mayoría de los casos. El cuadro clínico típico posee signos y síntomas importantes tales como fiebre, tos seca, disnea, mialgia o fatiga y linfopenia; como ya se ha mencionado existen síntomas muy variados que oscilan entre leves y muy graves, inclusive pueden existir pacientes asintomáticos [16].

El presente trabajo tiene la intención de centrar su estudio en los pacientes que necesitan intervención intensiva debido a sus síntomas graves, es decir, aquellos que necesitan respiración asistida. Los médicos se encuentran en un predicamento, debido a que no se ha determinado con certeza el tiempo preciso en donde un paciente necesite recibir respiración asistida, intubar un paciente muy temprano o muy tarde puede tener repercusiones graves sobre la salud a largo plazo o en casos extremos la muerte. Cabe mencionar que en los momentos actuales la falta de instrumentación médica que solvente dicho problema tiene un impacto significativo sobre el dilema antes presentado.

2.3. Escenarios clínicos anticipados

El proyecto presentado tiene como finalidad su despliegue en ambientes en donde todo tipo de soporte respiratorio invasivo convencional haya sido agotado y deberá ser usado por profesionales de la salud entrenados.

En el caso específico del COVID-19, se han anticipado los siguientes escenarios de emergencia en donde el presente proyecto pueda ser usado de forma segura [17]:

- Un paciente con COVID-19 que se encuentre en un estado de deterioro, que presente dificultad para respirar o hipoxia, es decir que su respiración no pueda de forma adecuada oxigenar su sangre.
- Empeoramiento del cuadro clínico de un paciente cuando desarrolle Síndrome de Distrés Respiratorio Agudo (SDRA).
- Pacientes que deban salir de la Unidad de Cuidados Intensivos (UCI), por algún tipo de examen externo al mismo o que se haya determinado que se deba brindar soporte fuera de la UCI por algún motivo distinto.

Todos estos pacientes deberán ser intubados o alternativamente se podrá realizar una traqueostomía, por tanto deberán permanecer sedados y paralizados, impidiendo de esta forma que exista un des-sincronismo entre el paciente y el ventilador, si es que se diera el caso en donde el control de asistencia no se encuentre disponible.

Como ya se ha mencionado, es de extrema importancia que un equipo multidisciplinario (médico, enfermera de cuidado crítico y terapeuta respiratorio) se encuentre monitorizando el estado del paciente en todo momento, adicionalmente se recomienda que el laboratorio clínico sea capaz de monitorizar los gases presentes en la sangre y otros exámenes correspondientes a la UCI de forma constante para que el equipo clínico pueda realizar ajustes y decisiones apropiadas.

2.3.1. Síndrome de distrés respiratorio agudo

Pacientes con SDRA deberían, preferiblemente, usar ventilación asistida por ventiladores estándar para UCIs, un ventilador mecánico está destinado al caso específico en donde ventiladores tradicionales se encuentren agotados, o para pacientes cuyos problemas pulmonares no precisen de máquinas con mecanismos demasiado sofisticados.

Comúnmente, la alteración del funcionamiento normal del sistema pulmonar humano por un mecanismo externo a este puede causar dolencias agudas o crónicas. En condiciones de SDRA, fluidos se encuentran presentes en el alveolo o en el espacio intersticial (entre el alveolo y un capilar sanguíneo), lo que resulta en cambios en la distribución de gases entre el alveolo y los vasos sanguíneos [18], otras condiciones incluyen:

- Cualquier patología que cause acumulación de fluidos en el pulmón (pulmón húmedo), causado por infecciones, inflamaciones, factores mecánico-pulmonares o factores hidrostáticos
- Cualquier patología que cause fibrosis de la estructura pulmonar.

El límite de seguridad de la terapia de ventilación no ha sido determinado todavía, sin embargo, en la situación actual que se encuentran enfrentando los profesionales de la salud, este proyecto intentará dar a los pacientes que no puedan obtener un lugar en las UCIs o hasta que un ventilador de mayores prestaciones se encuentre disponibles, una alternativa que pudiese salvar sus vidas.

2.4. Tecnologías de respiración asistida

La publicación del documento "Recomendaciones para la expansión de capacidades de atención clínica y despliegue de equipos médicos de emergencia" [19] de la OPS (Organización Panamericana de la Salud) tuvo el objetivo de ofrecer recomendaciones para facilitar la expansión de capacidades de atención clínica. Incluye recomendaciones para el despliegue de equipos médicos de emergencia, esto para asegurar una respuesta positiva ante un elevado número de pacientes que pudiera exceder los límites de la red integrada de servicios de salud de una comunidad o área afectada por COVID-19.

En este documento[19], basado en la mayor cohorte de pacientes con COVID-19, ellos dictaminan que:

- El 40 % de los pacientes pueden tener una enfermedad leve donde el tratamiento será principalmente sintomático y no necesitará atención hospitalaria.
- Alrededor de un 40 % tendrá una enfermedad moderada que podría requerir atención hospitalaria.
- Un 15 % tendrá una enfermedad grave que requerirá oxigenoterapia entre otras intervenciones hospitalarias.
- Alrededor de un 5 % tendrá una enfermedad crítica que requerirá de ventilación mecánica

2.4.1. Ventilación no invasiva

2.4.1.1. Modalidades de ventilación mecánica no invasiva en el apoyo a pacientes con SDRA

Existe la presión positiva continua en las vías respiratorias (CPAP) y la presión positiva de dos niveles en las vías respiratorias (BiPAP), que son modos de soporte de ventilación no invasiva. Es importante señalar que las soluciones a base de mascarillas no son suficientes para los pacientes que experimentan insuficiencia respiratoria hipóxica. En el mejor de los casos, se trata de soluciones temporales. Ambas soluciones corren el riesgo de aerosolizar COVID-19 en el aire expirado que, por lo general, no se filtra.

La CPAP se utiliza principalmente para ayudar con la oxigenación mediante la aplicación de una presión positiva constante durante todo el ciclo respiratorio del paciente. Se usa comúnmente para pacientes con obstrucción de las vías respiratorias o problemas de oxigenación. Los pacientes tienen insuficiencia respiratoria hipóxica, pero aún no tienen insuficiencia respiratoria.

BiPAP detecta los esfuerzos inspiratorios y espiratorios del paciente y proporciona una presión diferencial positiva en las vías respiratorias para ayudar a facilitar el trabajo respiratorio. Los pacientes que se cansan y demuestran insuficiencia respiratoria hipoxémica se benefician del método BiPAP en lugar del uso del método CPAP. Es importante saber identificar fallas y detectar cuando el paciente

necesita ser intubado.

La intubación es necesaria para los pacientes que experimentan insuficiencia respiratoria porque se elimina el esfuerzo de respirar. El paciente puede ser sedado y paralizado, si es necesario, y se usa una presión más alta para mantener el intercambio de gases en un pulmón enfermo. Algunos pacientes pueden recuperarse con CPAP y BiPAP, por lo que se debe usar la discreción clínica antes de la intubación.

En los pacientes COVID-19 positivos, las modalidades de ventilación mecánica no invasiva ciertamente pueden probarse, pero se debe diagnosticar sin demora una monitorización cuidadosa del empeoramiento del paciente hacia la insuficiencia respiratoria para poder iniciar la terapia ventilatoria invasiva. La demora y / o la falta de asistencia ventilatoria de presión positiva invasiva oportuna puede provocar una lesión pulmonar autoinfligida por el paciente [20]. En este punto, la única solución es la ventilación invasiva.

2.4.2. Ventilación mecánica clínica

A grandes rasgos, un ventilador mecánico moderno posee tres parámetros importantes:

1. Volumen tidal o corriente.
2. Inicio de la fase de inspiración o evento desencadenante.
3. Inicio de la fase de espiración o evento cíclico.

Estos parámetros mencionados son determinados en un inicio por un profesional de la salud y son configurados en el ventilador, además, dependiendo de los resultados de laboratorio y monitorización del paciente, ajustes en tiempo real son necesarios para optimizar su estado clínico. En términos técnicos, el paciente se convierte en un ”sensor” que determinará el funcionamiento de la máquina.

En un ventilador mecánico, se puede especificar un volumen en mililitros o una presión inspiratoria para el volumen tidal V_T, el cual es generalmente discutido y pensado como un valor en centímetros cúbicos por kilogramo de peso ideal de una persona, así como se muestra en la ecuación (2.1). Cuando un paciente sufre de SDRA, el volumen tidal de la máquina se suele mantener idealmente entre 4 a 8 cc/kg [21].

$$\begin{aligned} Peso\ Corporal\ Hombre_{Kg} &= 50 + [0{,}91\,(altura_{cm} - 152{,}4)] \\ Peso\ Corporal\ Mujer_{Kg} &= 45{,}5 + [0{,}91\,(altura_{cm} - 152{,}4)] \end{aligned} \tag{2.1}$$

Un profesional de la salud puede definir el volumen tidal, como se puede observar en la Figura 2.1, de esta manera la máquina intentará entregar dicho volumen seleccionado en un flujo inspiratorio de proporción uniforme sobre un tiempo inspiratorio especificado, este procedimiento es realizado sin tener en cuenta la presión que se acumula en los pulmones, la cual se denomina “Presión Inspiratoria Pico o PIP". Ventiladores modernos poseen funciones de protección ante daño a los pulmones por presión PIP dañina, dicho trauma a los pulmones se conoce como Barotrauma. Los ventiladores para respiración asistida poseen la capacidad de realizar una “retención al final de la inspiración", que es programar una duración sobre la cual la presión en el circuito es medida, está presión se denomina

"Presión Plateau o P_{plat}". Un ciclo controlado de respiración en conjunto con una retención al final de la inspiración se ilustra en la Figura 2.1.

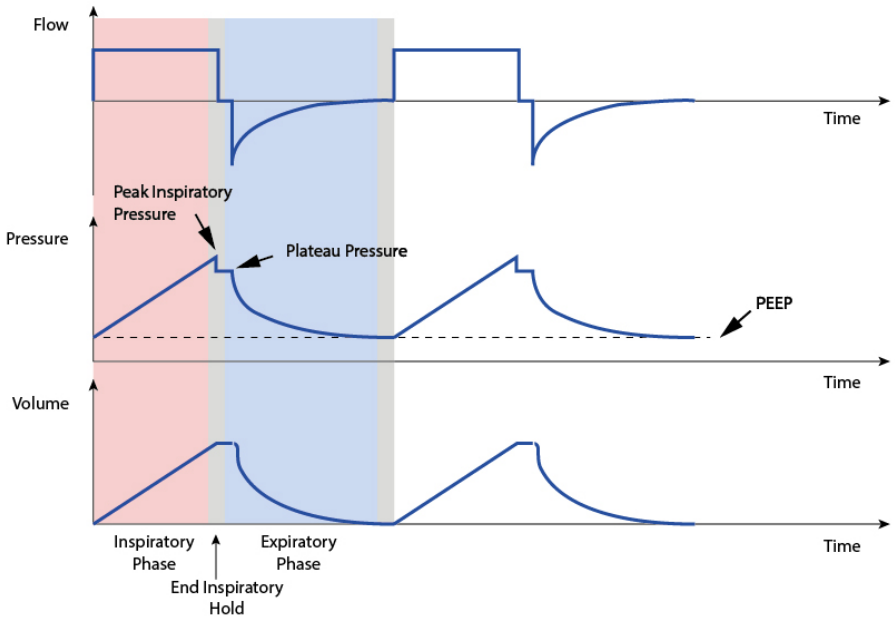


Figura 2.1: Perfiles de flujo, presión y volumen para ventilación controlada por volumen en dos ciclos de respiración [1]

El modo de control de presión, usa la presión brindada por el ventilador y la compliancia pulmonar del paciente en conjunto con el tiempo de inspiración para determinar el volumen de gas entregado (volumen tidal) a los pulmones [5], como puede observarse en la Figura 2.2. En el caso de los pacientes con COVID-19, existe un cuadro clínico desconocido, por lo que el proceso seguido en orden de realizar una ventilación mecánica se ha dado conforme a los mecanismos seguidos en pacientes con SDRA. Por lo que, se ha deducido que pacientes con COVID-19 poseen una compliancia pulmonar que cambia según el avance de la enfermedad en el sistema respiratorio. Es por esto que el volumen tidal deberá cambiar en el transcurso del tiempo en el caso de encontrarnos con un cuadro clínico que requiera ventilación mecánica a largo plazo.

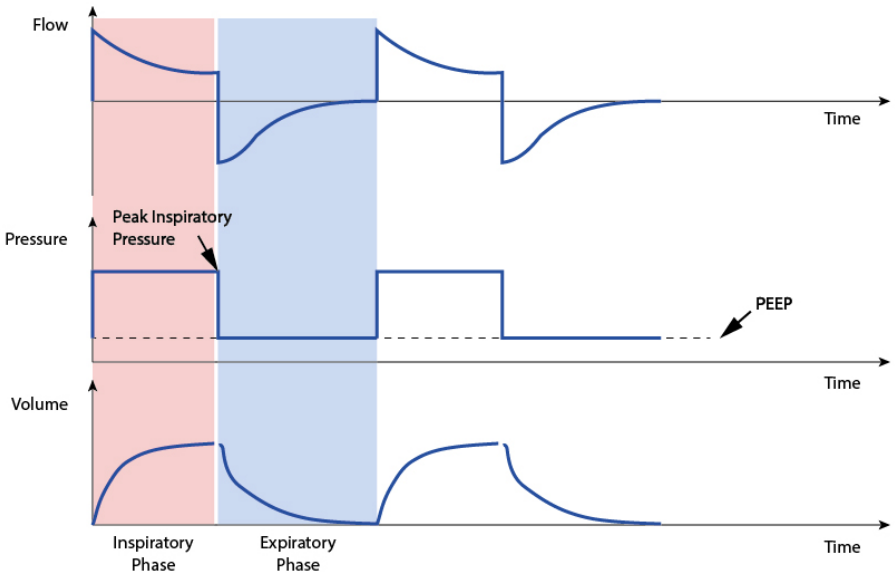


Figura 2.2: Perfiles de flujo, presión y volumen para ventilación controlada por control de presión en dos ciclos de respiración [1].

Lo mencionado en anterioridad en cuanto a compliancia pulmonar puede ser dividida según se estudie las vías respiratorias superiores o inferiores, como puede observarse en la Figura 2.3. Las vías respiratorias superiores consisten en las estructuras que serán atravesadas por el tubo endotraqueal, sean estas, boca, nariz, orofaringe y traquea. Las vías respiratorias inferiores se componen de los bronquios (izquierdo y derecho, los cuales se dividen en bronquios secundarios y terciarios, bronquiolos y alveolos). Como se ha mencionado anteriormente, la compliancia pulmonar varía dependiendo de la enfermedad, pudiendo ser de tipo restrictivos u obstructivos, ambos que pueden ser divididos en intrínsecos y extrínsecos. Pacientes con COVID-19 que desarrollan SDRA, desarrollan enfermedades intrínsecas restrictivas [22], las cuales necesitan una presión de referencia adicional para mantener abiertos los alveolos para que puedan seguir realizando el intercambio de gases. Esto se consigue mediante la presión positiva espiratoria o PEEP.

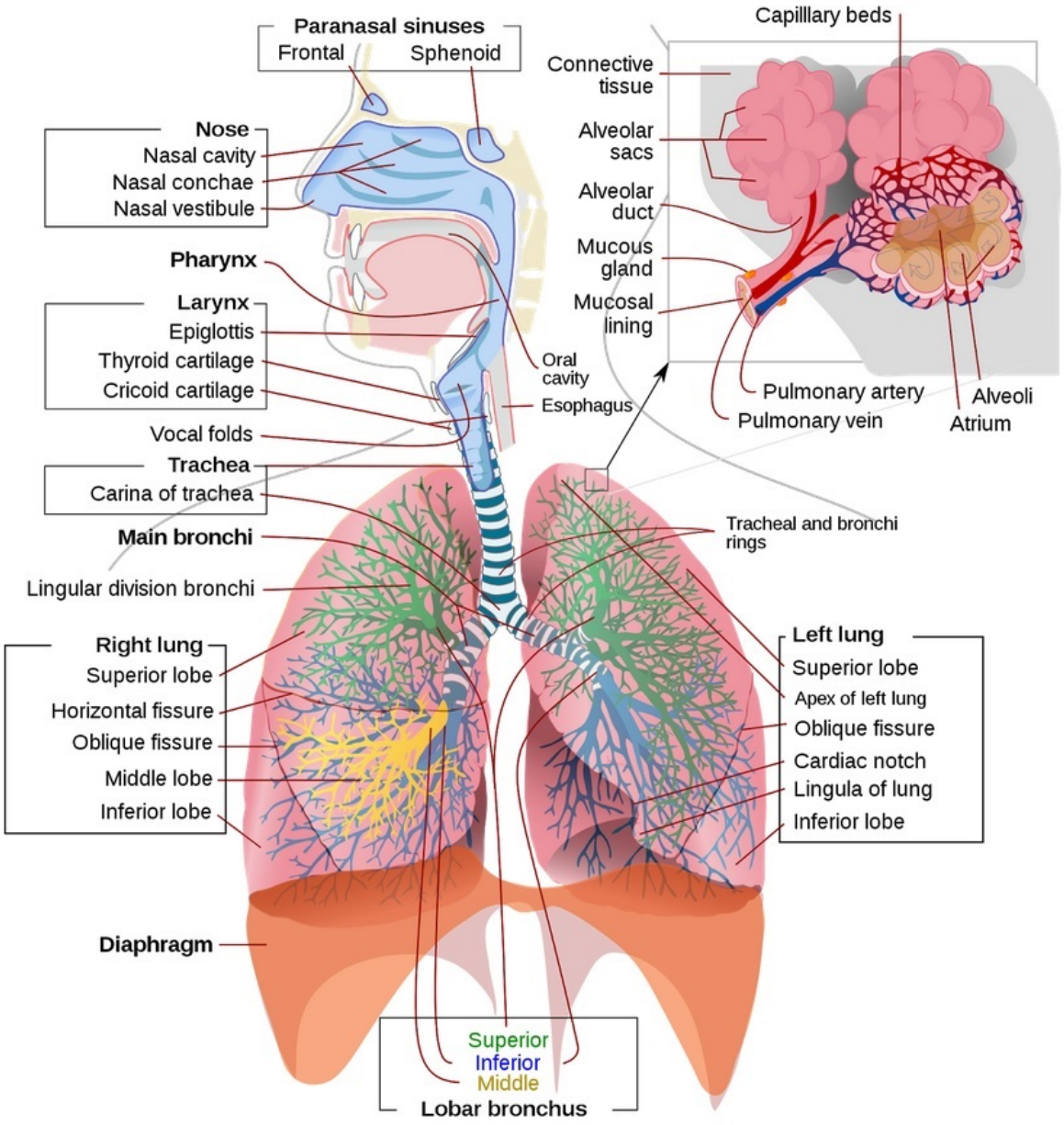


Figura 2.3: Anatomía de las vías respiratorias humanas [1].

2.4.3. Inicio de la fase inspiratoria: tiempo, presión y activación de flujo

La fase de inspiración puede ser ajustada para comenzar en un intervalo regular, ya sea escogiendo una tasa respiratoria constante o permitiendo que el ventilador mida el esfuerzo respiratorio del paciente, y ajustando el inicio de la fase inspiratoria del ventilador.

Ventiladores mecánicos modernos pueden ser configurados basándose en límites de flujo o presión para iniciar la respiración asistida, éstos son inherentes a la forma de construcción del ventilador o pueden ser configurados manualmente por un experto en salud. Por tanto, es imperativo mencionar la diferencia entre ventiladores diseñados para ambientes de cuidado intensivo, cuyo uso generalmente se

da por días o semanas, y ventiladores usados para pacientes con cuadros más leves que necesiten estar conectados por minutos o unas cuantas horas.

2.4.4. Inicio de la fase espiratoria: tiempo, volumen y ciclos de presión

El inicio de la fase espiratoria puede ser determinado mediante varias variables: tiempo, volumen, flujo y presión. La duración de la fase inspiratoria puede ser programada y la espiración comienza luego de que el tiempo de inspiración haya concluido, esto se denomina ciclo de tiempo. En control de volumen, la inspiración se detiene luego de que el volumen inspiratorio escogido ha sido entregado, esto se denomina ciclo de volumen. El ventilador puede pasar de su fase de entrega de gas en inspiración a espiración cuando el pico de flujo inspiratorio haya alcanzado un 10 o 25 % de flujo de inspiración [1], este proceso se llama ciclo de flujo.

Así mismo, una inspiración puede ser cambiada a espiración cuando un límite de presión sea alcanzado, en instancia, si un paciente presenta algún evento en donde sé desincronice con el ventilador, la presión de aire aumentará dramáticamente, lo que puede llegar a ser peligroso para el paciente, ya que la ventilación no es efectiva cuando el paciente lucha en contra de la ventilación, en este estado el ventilador cambia de fase de inspiración a fase de espiración y activa una alarma de presión alta, este proceso se denomina ciclo de presión.

Adicionalmente, un único respiro del paciente (inspiración y espiración) se denomina ciclo de respiración, el tiempo de inspiración y espiración es importante debido a que se debe asignar más tiempo a la espiración que a la inspiración, ya que se puede entregar una cantidad de aire superior a la soportada por los pulmones. Este factor puede ser modificado mediante el ajuste de la tasa de inspiración-espiración, Tasa I:E cuando una tasa respiratoria bpm (*breaths per minute*) sea usada.

La presión de final de espiración positiva o *Positive end-expiratory pressure, PEEP*, es aplicada en orden de mantener al pulmón “abierto", de esta manera impidiendo colapso alveolar y por tanto mejorando el intercambio de gases y reduciendo el atelectrauma (lesión pulmonar inducida por la ventilación mecánica). En adición, debido a la no homogeneidad en los tejidos pulmonares, la ventilación de presión positiva puede llevar a la sobre distensión de los alveolos (barotrauma), lo que puede impedir el intercambio de gases y empeorar el cuadro clínico. Las diferencias regionales en la compliancia pulmonar son dinámicas y cambian significativamente a través de la estadía del paciente en el hospital. Dependiendo si el paciente o la máquina determinan cada uno de los parámetros mencionados anteriormente, se crean diferentes modos de funcionamiento para el ventilador.

2.4.5. Conjunto de parámetros mínimos

Un ventilador mecánico automatizado debería inicialmente operar en modo de control de volumen, con una tasa inicial, ajustando la ventilación por minuto mientras la homeostasis del paciente es optimizada estudiando sus signos vitales y resultados de laboratorio. Comúnmente se proponen dos formas de funcionamiento para este tipo de ventiladores:

1. **Control de volumen:** entrega de un volumen tidal determinado funcionando en lazo cerrado,

usando medición de presión por motivos de seguridad.

2. **Control asistido:** el sistema medirá fluctuaciones de presión en el aire entregado al paciente, apoyando las respiraciones iniciadas con el esfuerzo del paciente y consiguientemente reconociendo y permitiendo la exhalación.

En la implementación clínica más sencilla, el sistema es configurado usando observación clínica directa y estudios de laboratorio, por tanto el dispositivo puede servir como un dispositivo transitorio a un dispositivo más avanzado, o como un ventilador de uso a largo plazo cuando la demanda en ventiladores haya sido saturada. Los suministros mínimos requeridos por el hospital a usar el ventilador desarrollado son los siguientes:

1. Bolsa resucitadora manual "Ambu": viene en diferentes configuraciones de mercado, se recomienda que posea una válvula de escape y una válvula PEEP.
2. Válvula PEEP (puede ser adicionada si es que no viene con la bolsa Ambu).
3. Tubos endotraqueales y/o de traqueostomía.
4. Circuito para respiración con mecanismo de válvula para el paciente que minimice el espacio muerto y la re-inhalación de CO_2.
5. Conector pequeño y flexible para conectar el final del circuito respiratorio al apartado de traqueostomía.
6. Oxígeno / mezclador de aire, si es posible con ajuste FiO_2.
7. Filtro HEPA para remover partículas del virus de los gases espirados (dependiendo si el paciente está aislado o no).

2.4.6. Parámetros de rendimiento mínimos

Los parámetros de rendimiento mínimos, así como sus valores recomendados pueden ser observados en la Tabla 2.1.

Tabla 2.1: Parámetros de rendimiento mínimos.

Parametro	Valor o Rango	Nota
Modos	Control de volumen Control asistido Seguridad ante fallo	Reconocer si un paciente para de respirar, cambiar a modo por defecto
Volumen tidal	200-800 mL 6 mL o menos / kg (peso ideal del paciente) como punto de partida	Debe ser ajustable
Tasa	8 - 40 o 10 - 40 respiraciones por minuto	Debe ser ajustable
PEEP	5 - 20 cm H_2O	Debe ser ajustable
Presión plateau	Límite: 40 - 60 cm H_2O; Conseguido con válvula	Generalmente fija dependiendo de la bolsa Ambu
Tasa I:E	1:2; rango de 1:1 - 1:4 Pacientes con COVID requieren generalmente 1:3 o superior	Debe ser ajustable
Filtración espirada	Filtro HEPA disponible como componente	Debe estar en el circuito respiratorio
Filtración inspirada	Filtro HEPA disponible como componente	Debe estar en el circuito respiratorio
Humidificación inspirada	Combinada con la saliente (auto humidificación)	Recomendada
Control de Asistencia (detección respiratoria o activador de sensibilidad)	Medición de presión de -1 a -5 cm H_2O	Recomendada - Requiere de un transductor de presión en el diseño
FiO_2	30 % - 100 %	Recomendada
Presión de inspiración pico PIP	Será determinada por el límite de la válvula de escape, si se usa un transductor de presión para medir continuamente la presión del canal de aire puede programarse para limitar el PIP	Fijo o ajustable - requiere de un transductor de presión en el diseño

2.4.7. Modalidad de funcionamiento de un ventilador mecánico

- Los pacientes son entubados o tendrán una traqueostomía.
- La presión PEEP es ajustada dependiendo del juicio del médico.
- La tasa de respiración es ajustada dependiendo de las necesidades médicas. Este parámetro será configurado de forma digital o mecánica, de no ser ajustado digitalmente el médico deberá hacerlo usando un reloj.
- La tasa I:E puede ser ajustada digitalmente o mecánicamente, de ser posible, además puede ser fija.

- Suministro de oxígeno debe estar conectado a la bolsa Ambu. La fracción de oxígeno inhalado puede ser ajustado con un mezclador de flujo de gas (mezclando 100 % de oxígeno y aire (21 % oxígeno) de acuerdo con el estado clínico del paciente).
- ABVV conectado al paciente, el paciente es observado y se ajustan sus características basándose en SpO2, evaluación clínica, etc.
- El volumen tidal es configurado observando la compresión de la bolsa Ambu de acuerdo al criterio del médico. Posteriormente el sistema puede ser calibrado para una marca de funda en particular y el volumen tidal inicialmente ajustado variarlo para estar lo más cerca posible a 6 cc/kg del peso corporal ideal.

2.4.8. Conocimientos clínicos adicionales

- La máquina más simple es capaz de operar en modo de control de volumen, que no comprende medición de respiración, esto es aplicado solamente a pacientes sedados y paralizados.
- En pacientes no paralizados, es improbable que toleren una máquina que no sea capaz de detectar inspiración. Es posible que los pacientes toleren si se agrega una válvula de vía única en inspiración.
- Cuando un paciente espontáneamente intenta respirar, aire debe fluir para evitar la generación de presión negativa, que en su caso puede conllevar a un edema pulmonar y empeorar la compliancia pulmonar y el intercambio de gases. Las bolsas Ambu generalmente tienen esta característica ya implementada.
- Una sobre presión PIP es requerida con una alerta activa basada en un transductor de presión, una solución aceptable es reducir el volumen inspirado, la mayoría de los pacientes deberían ser capaces de tolerar esto.
- El ajuste de la tasa I:E mejora la flexibilidad, pero no es esencial y sencillo de realizar con un control digital.
- SDRA: Basado en reportes de pacientes con COVID-19 que se encuentran entubados, estos poseen una compliancia pulmonar relativamente normal (dependiendo del avance de la enfermedad), y presiones de funcionamiento en respiración asistida normales pueden ser usadas (presión Plateau menos PEEP, intentando llegar a <15 cm H_2O). Sin embargo la compliancia pulmonar tiende a decrecer con un SDRA que empeora.

2.5. Bolsa de resucitación manual

El diseño mecánico desarrollado para este proyecto tiene base en el uso de una bolsa de resucitación Ambu. Ciertos requerimientos mecánicos deben cumplirse en orden de obtener un funcionamiento correcto en la respiración asistida mediante la bolsa Ambu.

Se debe tener cuidado al manipular la bolsa Ambu, considerando un uso de 7 días, con funcionamiento de 24 horas, cada 60 minutos, con un ciclo respiratorio que requiere comprimir la bolsa dos veces y considerando aproximadamente 30 respiraciones por minuto, equivalen a 604,800 ciclos de funcionamiento de la bolsa. Por tanto cualquier diseño de ventilador que use una bolsa ambu deberá ser lo suficientemente gentil con la bolsa para garantizar su duración, consiguientemente es recomendable que los brazos sean construidos de un material que reduzca la fatiga del material del que está hecha la

bolsa.

2.5.1. Modelo matemático de una bolsa de resucitación manual

Se asume que la forma de la bolsa Ambu será aquella parecida a un esferoide oblongo o prolato, es decir, una elipse rotada al rededor de su eje mayor. Su eje polar es más grande que su eje ecuatorial, que en el caso de una bolsa Ambu corresponde a un círculo perfecto antes de ser deformado por la presión de los brazos mecánicos [2]. La representación sobre los ejes puede observarse en la Figura 2.4

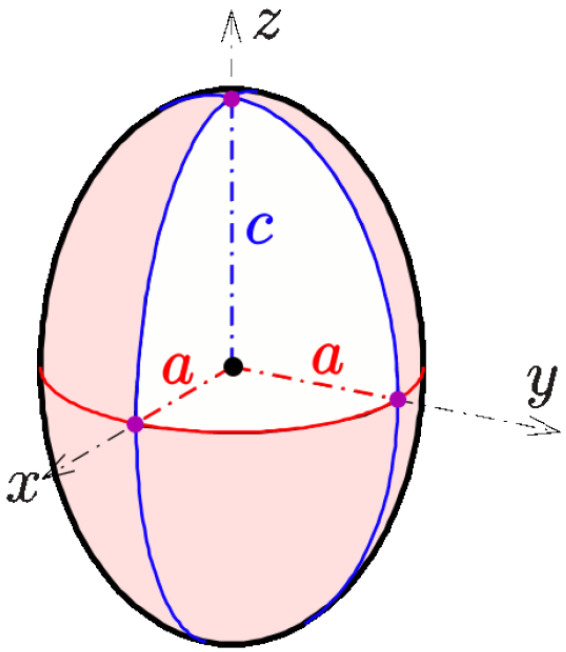


Figura 2.4: Representación de la bolsa Ambu [2].

El radio más grande, **c**, se acuesta sobre el eje **z**, el radio menor del círculo en el plano **x-y**, es denominado **a**. Partiendo de lo mencionado, el volumen esferoidal será el que se muestra en la ecuación (2.2).

$$V = \frac{4}{3}\pi c a^2 \tag{2.2}$$

El movimiento de un plano de presión viniendo desde el eje negativo **y** es asumido, de esta manera, el volumen diferencia en todo tiempo será:

$$dv = \pi x z dy \tag{2.3}$$

$$v = \pi \int_{-a}^{a} (xz)\, dy \tag{2.4}$$

La ecuación de la circunferencia en el plano **x-y** es:

$$\begin{gathered}\frac{x^2}{a^2}+\frac{y^2}{a^2}=1\\ x^2+y^2=a^2\end{gathered} \tag{2.5}$$

$$x=\sqrt{a^2-y^2} \tag{2.6}$$

La elipse en el plano **y-z** es:

$$\frac{y^2}{a^2}+\frac{z^2}{c^2}=1\rightarrow z^2=c^2\left(1-\frac{y^2}{a^2}\right)\rightarrow z=c\sqrt{1-\frac{y^2}{a^2}} \tag{2.7}$$

$$z=\frac{c}{a}\sqrt{a^2-y^2} \tag{2.8}$$

Reemplazando (2.6) y (2.8) en (2.4) se obtiene:

$$\begin{gathered}v\left(y\right)=\pi\int_{-a}^{a}\frac{c}{a}\left(a^2-y^2\right)dy=\pi\frac{c}{a}\int_{-a}^{a}\left(a^2-y^2\right)dy\\ v\left(y\right)=\pi\frac{c}{a}\left[a^2y-\frac{y^3}{3}\right]_{-a}^{a}\end{gathered} \tag{2.9}$$

Desde la ecuación (2.9), se integra hasta **h**, lo que representa el movimiento de los planos de presión que vienen desde el eje negativo **y**, de esta manera se obtiene:

$$v\left(y\right)=\pi\frac{c}{a}\left[a^2y-\frac{y^3}{3}\right]_{-a}^{h} \tag{2.10}$$

Evaluando los limites:

$$\begin{gathered}v\left(y\right)=\pi\frac{c}{a}\left[a^2h-\frac{h^3}{3}-a^2\left(-a\right)+\frac{\left(-a\right)^3}{3}\right]=\pi\frac{c}{a}\left[a^2h-\frac{h^3}{3}+a^3-\frac{a^3}{3}\right]\\ v\left(y\right)=\pi\frac{c}{a}\left[a^2h-\frac{h^3}{3}+\frac{2}{3}a^3\right]\end{gathered} \tag{2.11}$$

La ecuación (2.11), calcula el volumen como una función del movimiento de los planos de presión sobre la bolsa Ambu, si se obtiene la función inversa, esta representará cuanto los planos de presión deben moverse en orden de obtener un volumen de aire **x** definido por el usuario en el sistema. Desarrollando se obtiene lo siguiente:

$$\begin{gathered}\frac{va}{\pi c}=a^2h-\frac{h^3}{3}+\frac{2}{3}a^3\\ \frac{h^3}{3}-a^2h-\frac{2}{3}a^3+\frac{av}{\pi c}=0\\ h^3-3a^2h+3\frac{av}{\pi c}-2a^3=0\end{gathered} \tag{2.12}$$

La ecuación (2.12), es una ecuación de tercer grado reducida en **h** de la forma:

$$\begin{gathered}h^3+ph+q=0\\ p=-3a^2\\ con\\ q\left(v\right)=3\frac{av}{\pi c}-2a^3\end{gathered} \tag{2.13}$$

Si se grafica la ecuación (2.13), con **h** en el eje **x** y **v** en el eje **y**, para volúmenes de compresión **v** entre 0 y 500 cc y se asume un radio **a=5cm** y un radio superior **c=10cm**, se obtienen las curvas de solución mostradas en la Figura 2.5.

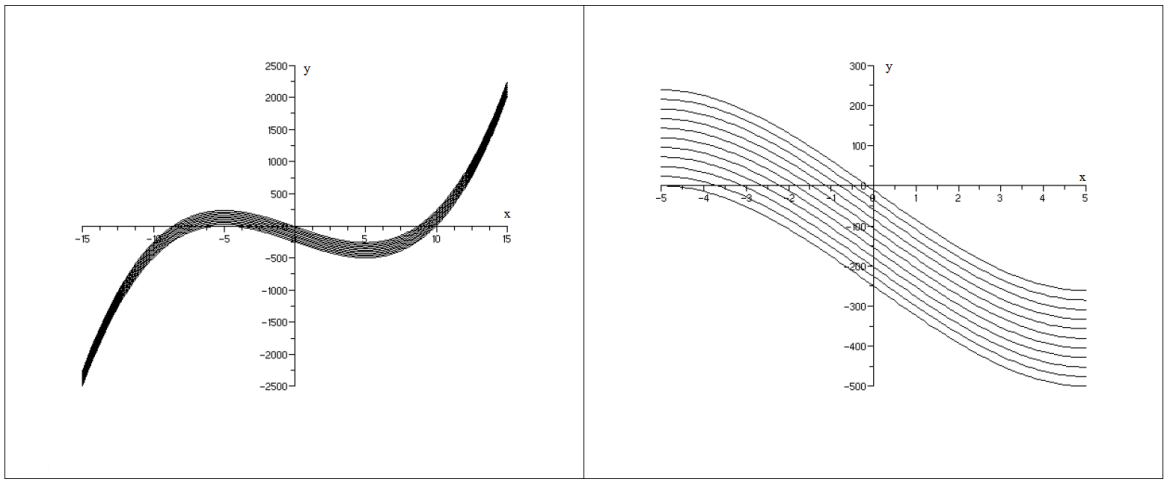


Figura 2.5: Curvas de solución para a=5cm, c=10cm y v de 0 a 500 cc.
Izquierda: Tres soluciones para el problema. Derecha: Zoom de las tres soluciones [2].

En la Figura 2.5, se observan tres soluciones para un valor **v** entre 0 y 500 cc, todas en el dominio real, ya que la ecuación (2.9), comienza en **a=-5** y tiene un máximo de 0, es decir, mitad de la bolsa, es posible reconocer cual de las tres soluciones en cada curva es útil para controlar la presión de los brazos mecánicos.

Existen varios métodos para encontrar las tres soluciones a la ecuación (2.13) , pero la más conveniente es la siguiente:

$$h = 2\sqrt{-\frac{p}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{3q}{2p}\sqrt{\frac{-3}{p}}\right) - \frac{2\pi k}{3}\right] \quad para\ k = 0,\ 1,\ 2 \tag{2.14}$$

Probando valores con la ecuación (2.14), se determina que una solución deseada ocurre cuando **k=1**, por tanto, la formula que relaciona el movimiento de los planos de presión como una función de volumen es:

$$h(v) = \sqrt{-\frac{p}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{3q}{2p}\sqrt{\frac{-3}{p}}\right) - \frac{2\pi}{3}\right] \quad p = -3a^2 \quad con \quad q(v) = 3\frac{av}{\pi c} - 2a^3 \tag{2.15}$$

Si se grafica la ecuación (2.15) entre 0 y 500 cc, se obtiene la Figura 2.6.

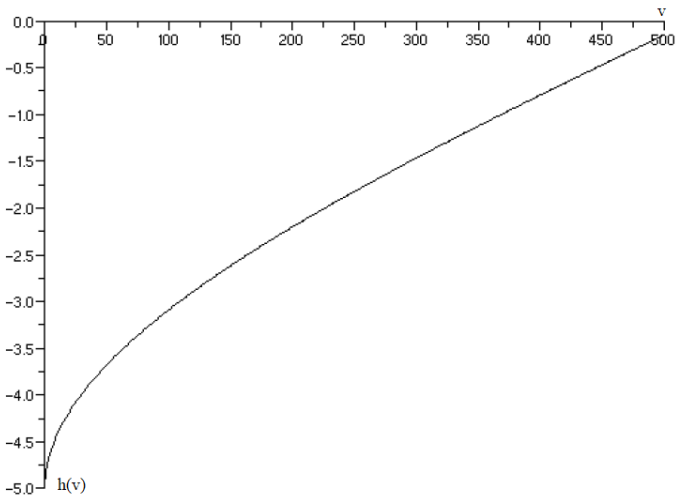


Figura 2.6: Curva de volumen para v entre 0 y 500 cc [2].

En este caso los valores de **h** son negativos, como se esperaba al mover **y** en -5 hacia el origen de las coordenadas. Para tener valores positivos se debe agregar una constante **a** como sigue:

$$h_2(v) = h(v) + a = 2\sqrt{-\frac{p}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{3q}{2p}\sqrt{\frac{-3}{p}}\right) - \frac{2\pi}{3}\right] + a \tag{2.16}$$

Lo que permite graficar valores positivos para h como puede observarse en la Figura 2.7.

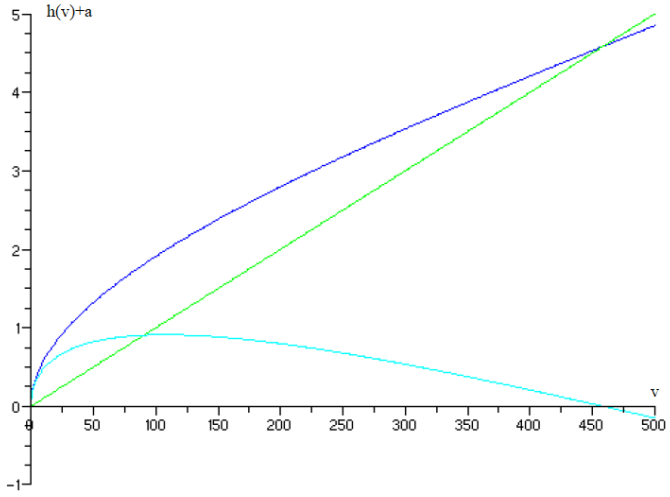


Figura 2.7: Gráfica de la ecuación (2.16) en azul, comportamiento lineal en verde y su diferencia en celeste [2].

Entonces la ecuación (2.16) recibe la entrada solicitada por el usuario y como resultado entrega la distancia que debe ser presionada en uno de los lados de la bolsa Ambu. Debe recordarse que los planos de presión deben ser planos que encajen con la ecuación y que el volumen real deberá ser el doble, ya que el otro lado de los brazos presionará la funda en el lado contrario con el mismo desplazamiento absoluto. Estas consideraciones pueden ser usadas para encontrar la forma final de la ecuación **h(v)**

como siguen:

Desde la ecuación (2.15):

$$\begin{gathered} h\left(v\right)=2\sqrt{-\frac{p}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{3q}{2p}\sqrt{\frac{-3}{p}}\right)-\frac{2\pi}{3}\right] \\ p=-3a^2 \\ reemplazando \\ q=\frac{3av}{\pi c}-2a^3 \\ se\ obtiene \\ h\left(v\right)=2\sqrt{\frac{3a^2}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{1}{2}\left(\frac{9av}{\pi c}-6a^3\right)\sqrt{\frac{-3}{(-3a^2)^3}}-\frac{2\pi}{3}\right)\right] \\ h\left(v\right)=2\sqrt{\frac{3a^2}{3}}\cos\left[\frac{1}{3}\arccos\left(\frac{1}{2}\left(\frac{9av}{\pi c}-6a^3\right)\frac{1}{\sqrt{(-3)^2a^6}}-\frac{2\pi}{3}\right)\right] \end{gathered} \tag{2.17}$$

Se preserva el signo negativo en signo de -3 en la raíz cuadrada, ya que la solución positiva no provee soluciones en el rango necesitado. Continuando:

$$\begin{gathered} h\left(v\right)=2a\cos\left[\frac{1}{3}\left(\arccos\left(\frac{9av-6a^3\pi c}{2\pi c}\frac{1}{-3a^3}\right)-2\pi\right)\right] \\ h\left(v\right)=2a\cos\left[\frac{1}{3}\left(\arccos\left(\frac{2\pi a^3c-3v}{2\pi a^2c}\right)-2\pi\right)\right] \\ h\left(v\right)=2a\cos\left[\frac{1}{3}\left(\arccos\left(1-\frac{3v}{2\pi a^2c}\right)-2\pi\right)\right] \end{gathered} \tag{2.18}$$

En donde **a**, corresponde al radio menor en cm y **c** el radio mayor en cm, de la bolsa Ambu.

En la ecuación (2.18), se tiene la fórmula general que deberá ser modificada para ser insertada en el código del microcontrolador teniendo las siguientes consideraciones:

- La ecuación (2.18) recibe una entrada de volumen requerido y retorna la distancia que debe ser presionada desde un lado de la bolsa, cuando se usa ambos brazos mecánicos, la fórmula debe ser modificada, ya que cada lado obtendrá solamente la mitad del volumen v_i requerido por la entrada del usuario.
- La integral ha sido evaluada desde **-a** hasta **h**, de forma que la fórmula responda a valores **[-a,0]**, es necesario convertirlas a **[0,h]**, esto se consigue agregando una constante **a**.
- El número de pasos *(ticks)*, provistos por el motor deben incluir su movimiento máximo, divididos para la distancia máxima al lado de la bolsa específico que será presionado por el plano de presión.

Si se aplican las consideraciones anteriormente mencionadas se obtiene la última ecuación:

$$\begin{gathered} P_c=\frac{P_{máx}}{a}\left\{2a\cos\left[\frac{1}{3}\left(\arccos\left(1-\frac{3(v_i/2)}{2\pi a^2c}\right)-2\pi\right)\right]+a\right\} \\ P_c=P_{máx}\left\{2\cos\left[\frac{1}{3}\left(\arccos\left(1-\frac{3v_i}{4\pi a^2c}\right)-2\pi\right)\right]+1\right\} \end{gathered} \tag{2.19}$$

La ecuación (2.19) es la fórmula que recibe la entrada de volumen requerida por el usuario y retorna el número de pasos *(ticks)*, que el motor debe realizar en orden de conseguir dicho volumen. Esto permite acomodar cualquier tipo de forma de onda respiratoria requerida.

2.6. Trabajos relacionados

Los ventiladores para respiración asistida generalmente son máquinas sofisticadas, desarrolladas bajo estándares internacionales que tratan de salvaguardar la seguridad de los pacientes, al mismo tiempo que garantizan el desempeño y correcto funcionamiento de los dispositivos en todo momento.

El ciclo de respiración humano es un proceso complejo, que varía de persona a persona y que puede cambiar bruscamente y en tiempos indefinidos estando bajo los efectos de ciertas patologías, dependiendo del tipo de ventilador, estos pueden tener principios de funcionamiento muy distintos entre sí y adicionalmente poseen modos de funcionamiento específicos que se intercambian dependiendo del tipo de enfermedad del paciente y bajo el criterio del médico tratante, además cada modo de funcionamiento tiene parámetros de configuración que sirven como base para su funcionamiento y monitorización. Toda la ingeniería aplicada en estos dispositivos lleva a que su costo tienda a elevarse enormemente, por lo que existen varios autores que han desarrollado prototipos de ventiladores de bajo costo, que aunque sean menos sofisticados y su uso sea específico para cierto modo de funcionamiento o para cierta patología específica, siguen siendo muy útiles en ambientes de alta saturación de pacientes o de pocos recursos, tal como ocurre en el estado de actual de la pandemia por COVID-19.

Dentro de la gran variedad de prototipos desarrollados, en general, para ventiladores mecánicos, se tiene como base de funcionamiento la bolsa Ambu, ya que esta se encuentra presente en todo ambiente médico y al ser desechable y de gran portabilidad resulta muy conveniente usarla. Lo que diferencia el funcionamiento de cada ventilador mecánico de bolsa Ambu, es el proceso que se sigue para obtener el volumen de aire requerido de la bolsa, lo más factible es presionar la bolsa para que de esta manera se entregue el volumen de aire al paciente, Díaz [23], usa una bolsa Ambu que es aplastada por un mecanismo de actuador de media luna, el cual mediante un desplazamiento angular de una leva permite una deformación uniforme y constante. Vasan y otros [24], usan así mismo una bolsa Ambu, esta vez usan un brazo de material plástico conectado con una pieza semicircular en su parte superior y que mediante un hilo de nailon hace un movimiento a manera de guillotina para presionar la bolsa Ambu.

Como se ha mencionado ya, el principio de funcionamiento de los distintos prototipos de ventiladores diseñados para esta emergencia sanitaria varía, Johar y Yadav [25], usan un compresor de aire, el cual envía la mezcla de gases necesarios a través reguladores y válvulas de aire, una válvula solenoide que se abre y cierra es la responsable de brindar la respiración al paciente, Fang y otros [26], desarrollaron un prototipo que tiene como principio de funcionamiento una bolsa Ambu, sin embargo, a diferencia de los métodos más tradicionales mencionados anteriormente, éste proyecto realiza el aplastamiento de la bolsa Ambu encerrándola en un espacio hermético y mediante el uso de una presión positiva enviada desde una válvula solenoide, aplasta la bolsa Ambu, cuyo volumen de aire de salida es el responsable de dar la respiración al paciente.

Generalmente, los ventiladores más sofisticados poseen distintos modos de ventilación (basados en presión, volumen, flujo, etc.), sin embargo todos se rigen bajo las normativas de los entes de regulación en cuanto a límites y parámetros de funcionamiento, por tanto, la mayoría de ventiladores funcionan bajo límites de funcionamiento muy similares, en la tabla 2.1, ya se ha desglosado los parámetros

de funcionamiento recomendados para un paciente que necesite respiración asistida debido al COVID-19.

2.7. Conclusiones

El COVID-19, al ser una enfermedad nueva, no posee un tratamiento definido, las complicaciones que se presentan por esta enfermedad, en los casos más graves llevan a la necesidad de intubar a los pacientes para darles respiración asistida, sin embargo, todavía no existe una guía específica de comportamiento dentro de la ventilación que determine el tiempo y los parámetros de ventilación para los pacientes bajo efectos de esta dolencia, por lo que los médicos se han basado en las enfermedades como el SDRA para dar tratamiento. La presente sección ha dado una muy breve introducción al COVID-19 y a la ventilación en general.

Dentro del ámbito de la respiración asistida, se ha visto que existen varios parámetros de control en la ventilación clínica, los principales modos de ventilación centran su funcionamiento en el control de volumen, presión, flujo de aire y tiempos de inspiración y espiración, y se ven extremadamente ligados al estado pulmonar del paciente, lo que hace que, durante el avance del tratamiento, tengan que ser modificados en tiempo real por los profesionales de la salud.

Finalmente se mencionaron los elementos clínicos necesarios para la operación del prototipo construido, así como los parámetros de rendimiento mínimos del ventilador, el funcionamiento como tal del ventilador estará a cargo del criterio del médico tratante.

Capítulo 3

Diseño e implementación mecánica

El presente capítulo describe el procedimiento empleado para enfrentar la problemática expuesta, mostrando las características de los dispositivos utilizados y el diseño del prototipo de ventilador en su parte mecánica. Así mismo, se describe los métodos utilizados para el abaratamiento de costos y la adaptación a la realidad de la industria ecuatoriana.

3.1. Introducción

Esta sección documenta el diseño mecánico del ventilador de emergencia. Cualquier diseño mecánico debe cumplir con las especificaciones claves de ventilación descritas en el anexo C.

El concepto básico consta de dos brazos que se cierran suavemente en sincronía para comprimir la bolsa. Esto debe ir acompañado de un sistema de control de circuito cerrado. Los principales requisitos de diseño mecánico que se ha considerado, incluyen:

- Se debe ser cuidadoso con la bolsa y sus mangueras. El diseño debe asegurar la bolsa, sujetarla y apretarla suavemente por ambos lados para reducir el riesgo de fatiga del material. Las pinzas deben ser lisas y tener una forma que maximice la expulsión de aire sin dañar la bolsa. La bolsa debe apoyarse con flexibilidad para permitir el movimiento durante la operación.

- Funcionamiento a prueba de fallos. Si la máquina falla, un médico debe poder apagar inmediatamente, abrir el dispositivo manualmente, retirar la bolsa y convertirlo en asistencia respiratoria manual.

- Múltiples posibilidades de detección y motor de accionamiento.

- Se ha pensado en este diseño como un modelo para cumplir con las capacidades del suministro local.

Las dimensiones generales y la operación están configuradas para ejecutar este diseño y ajustarlo a varios materiales y tecnologías de fabricación disponibles localmente. En este caso, el acero inoxidable se usó para los brazos con engranajes, perfiles de aluminio para los soportes del ventilador e impresión 3D en PLA para el resto de partes. En cuanto los puntos de movimiento se ha usado vástagos de acero, prisioneros y rodillos.

3.2. Diseño y construcción de los brazos con engranajes

Para la construcción del prototipo se comenzó por la parte mecánica, y lo más importante es la interacción entre los brazos mecánicos y el funcionamiento del motor. Es por esto que el primer diseño a tomar en cuenta fue el de los engranajes. Para esto se tomó los cálculos desarrollados por los ingenieros del MIT, mostrados en el apéndice F, y se procedió a replicar dichos cálculos. Para cumplir con las especificaciones se usó como guía los diseños mostrados en la figura 3.1. A continuación, se describe las medidas de los brazos, el piñon y los materiales usados.

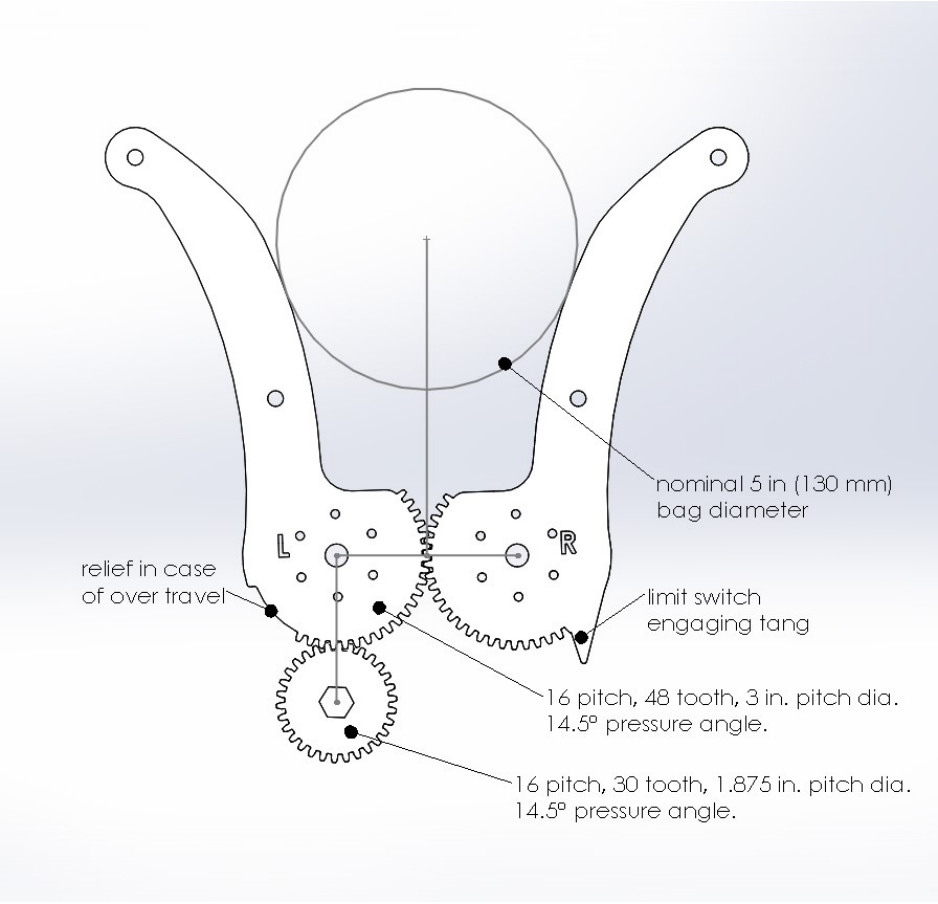


Figura 3.1: Sección transversal del brazo y del engranaje [3].

- **Engranaje grande (parte inferior de los brazos):** paso 16, 48 dientes, diámetro de paso de 3 pulgadas, Ángulo de presión de 14,5, grosor de 0,25 pulgadas.

- **Piñón impulsor:** paso 16, 30 dientes, diámetro de paso de 1.875 pulgadas. Ángulo de presión

de 14.5. 0,5 pulgadas de grosor. Esto es para adaptarse a la desalineación axial con los engranajes de los brazos.

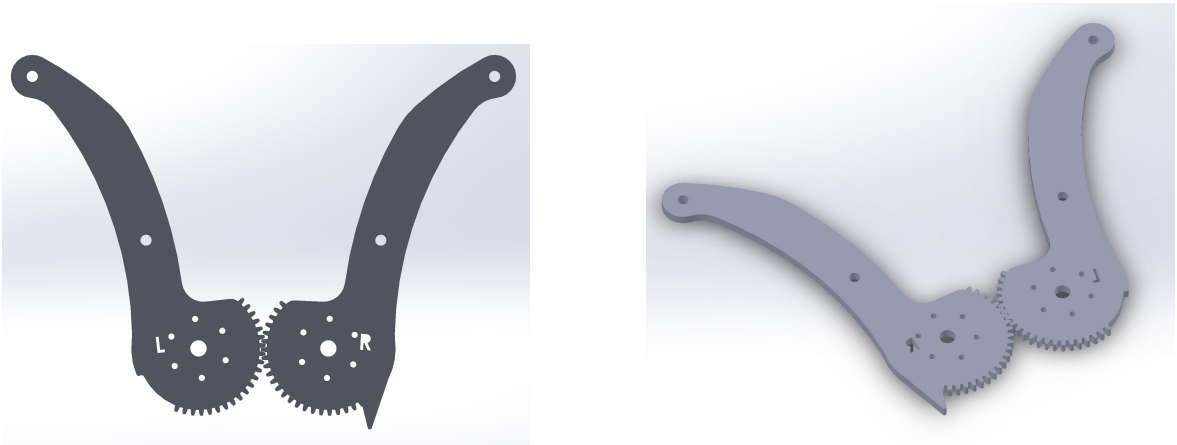
- **Relación de engranajes:** 1,6 (brazo/piñón)

- Con base en el par estimado (τ) de 10 Nm por brazo, dado en el cálculo de potencia, dividido por la relación de transmisión, llegamos a 12.5 Nm aplicados al piñón de diámetro (d) 0.0476 m (1.875 pulgadas) con ángulo de presión (ϕ) de 14,5° y la carga radial neta (F) sobre el piñón viene dada por: $F = 2\tau/(\cos(\phi) * d)$ $F = 2 * 12,5/(\cos(14,5)\,0,0476) = 550\ N$. El diagrama usado para este diseño se muestra en el apéndice F

- Esta carga radial se aplica al piñón aproximadamente a 2 cm de la cara de la caja de cambios, lo que da como resultado una fuerza de flexión en el eje de la caja de cambios que debe ser soportada por los cojinetes de la caja de cambios. Para esto se debe consultar al fabricante del motor.

- **La elección del material es extremadamente importante:** los prototipos creados fueron basados en los materiales disponibles en el país. La vida útil del engranaje del brazo y del piñón impulsor debe comprobarse en busca de desgaste y fatiga, en función de la selección del material y del ancho de las piezas, debido a que esta es una carga oscilante con fuerza en la carrera de entrada, mientras que la carrera de retorno está casi descargada.

Luego, se usó el software Solidworks para representar de manera correcta la interacción de los brazos con engranajes y su posterior construcción. Se puede observar el corte frontal y el corte en perspectiva del diseño de los engranajes en las figuras 3.2a y 3.2b



(a) Corte Frontal (b) Corte en perspectiva

Figura 3.2: Cortes en Solidworks del diseño de los brazos con engranajes.

El material usado para los brazos fue acero inoxidable no magnético y el método para construirlos fue por corte con chorro de agua. El material fue escogido debido a su conveniencia y seguridad, ya que, en los centros de salud no se debe manejar equipos médicos con partes magnéticas. En las figuras 3.3b, 3.3a y 3.3c se muestra los brazos con engranajes construidos.

(a) Brazo izquierdo. (b) Brazo derecho. (c) Brazos en conjunto

Figura 3.3: Brazos con engranajes construidos en acero inoxidable.

En cuanto al piñón impulsor que transmite el movimiento del motor, se adquirió un piñón que cumplía con las características de diseño. En las figuras 3.4 y 3.5 se puede apreciar en piñón y el sistema de engranajes acoplado.

Figura 3.4: Piñón transmisor de movimiento

Figura 3.5: Conjunto de engranajes acoplado.

3.3. Cálculo de potencia

El diseño del prototipo propuesto, como ya se ha mencionado, consiste en un par de brazos mecánicos que aplastarán la bolsa Ambu, para que ésta a su vez, brinde la mezcla de gases necesarios para la respiración asistida del paciente. Esta sección estima la potencia requerida por el motor. Cabe mencionar que probar con diferentes tipos de diseños puede derivar en un cambio de las especificaciones del motor a elegir, sin embargo, la potencia del mismo debería mantenerse igual.

3.3.1. Potencia teórica requerida del motor

Para poder escoger correctamente el motor a utilizar en el prototipo, es imperativo tomar el peor caso de uso del mismo y en dicho caso realizar el cálculo de potencia requerido, las variables de las cuales dependerá la potencia del motor son las siguientes:

- Presión máxima en las vías respiratorias: $P_{airway,\text{máx}} = 40\ cm\ H_2O$ (Presión de ruptura)
- Tasa de respiración máxima: $RR_{\text{máx}} = 40\ bpm$
- Tasa inhalación:Exhalación mínima: $I : E_{\text{mín}} = 4$
- Volumen de aire entregado máximo: $V_{\text{máx}} = 800\ cm^3$

Con las variables entregadas, se procede a tomar el peor caso de funcionamiento posible, en donde el dispositivo tiene que brindar una presión de aire correspondiente a 40 $cm\ H_2O$, en un tiempo correspondiente de 0,3 segundos, que viene dado por la ecuación (3.1):

$$t_{inhalación} = \frac{60\ seg}{\frac{RR_{\text{máx}}}{(1+I:E_{\text{mín}})}} \tag{3.1}$$

De esta manera, la tasa de flujo volumétrica en el peor caso, o caso pico está mostrada en la ecuación (3.2):

$$Q_{Vrespiratorias} = \frac{V_{\text{máx}}}{t_{inhalación}} = 0,0027\ {}^{m^3}/_{s} \tag{3.2}$$

La salida de potencia, en forma de flujo de volumen presurizado en las vía respiratorias se muestra en la ecuación (3.3):

$$P_{Vrespiratorias} = P_{Vrespiratorias,\text{máx}} Q_{Vrespiratorias} = 10{,}46\ W \tag{3.3}$$

Como es de esperarse, algo de potencia usada para aplastar la bolsa Ambu se pierde debido a la deformación de la bolsa, fricción, etc. Estimando que el 50 % de potencia se convierte en flujo de volumen presurizado, la potencia requerida por el brazo es el mostrado en la ecuación 3.4:

$$P_{brazo} = 2P_{Vrespiratorias} = 20,92\ W \tag{3.4}$$

Por tanto, el poder real requerido del motor será superior, que tan superior depende de los diseños eléctricos y mecánicos, asumiendo que la salida de poder del motor se pierde a la mitad del ingresado, debido a ineficiencias eléctricas y mecánicas, el poder real requerido por el motor se muestra en la ecuación (3.5):

$$P_{motor} = 2P_{brazo} = 41{,}84\ W \tag{3.5}$$

3.3.2. Requerimiento de potencia para un diseño de dos brazos

Se muestra a continuación una aproximación adicional al cálculo de potencia, para este cálculo se hace uso de un diseño de doble brazo como se muestra en la figura 3.6.

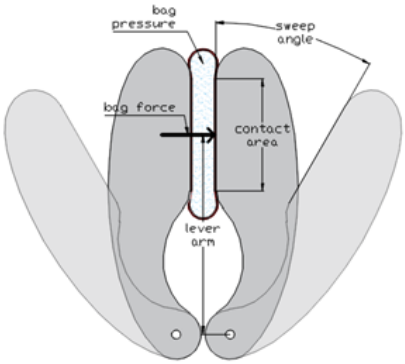


Figura 3.6: Diseño de dos brazos [3].

Es posible realizar un cálculo de potencia más directo para el diseño de la figura 3.6, teniendo en cuenta que las siguientes cantidades pueden ser medidas:

- Área de contacto de la bolsa Ambu
- Longitud del brazo para los dedos
- Ángulo de barrido

Para el prototipo diseñado, se tienen las siguientes cantidades:

- Área de contacto de la bolsa Ambu = 90 mm x 115 mm
- Longitud del brazo para los dedos = 12 cm
- Ángulo de barrido = 30°

La fuerza máxima sometida a la bolsa Ambu por un dedo, cuando se encuentra completamente aplastada, usando la misma eficiencia de transmisión de presión como explicado anteriormente y para las variables descritas se obtiene el resultado de la ecuación (3.6).

$$F_{dedo} = 2A_{bolsa}P_{Vrespiratorias,\text{máx}} = 81,199\ N \tag{3.6}$$

El torque máximo necesitado por cada dedo es el mostrado en la ecuación (3.7)

$$\tau_{dedo} = F_{dedo}l_{dedo} = 9,74\ N \tag{3.7}$$

De esta manera, se puede calcular la potencia requerida por los brazos, que sujetan a la bolsa Ambu y la aplastan usando la velocidad angular de barrido, en un tiempo de 0,3 segundos, como se muestra en la ecuación (3.8).

$$P_{brazos} = 2 \times \tau_{dedo}W_{dedo} = 34,01\ W \tag{3.8}$$

La potencia total requerida por el motor, asumiendo el uso de un solo motor, aplicando adicionalmente, una eficiencia del motor y del engranaje del 50 % quedaría como en la ecuación (3.9).

$$P_{motor} = 2 \times P_{brazos} = 68{,}03\ W \sim\ 70\ W \tag{3.9}$$

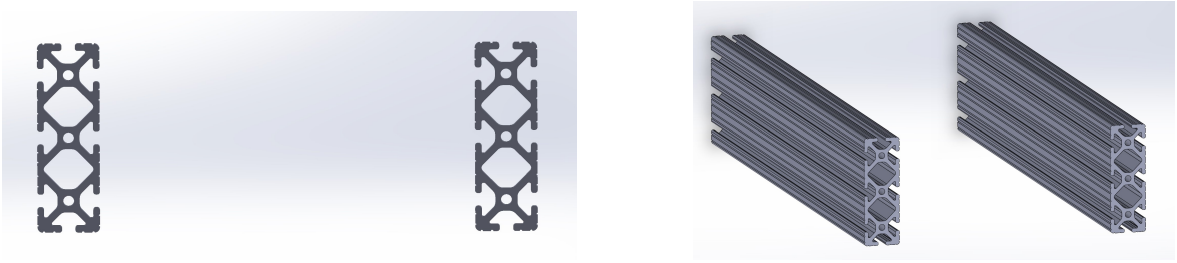
Por tanto, el requerimiento de motor para el prototipo es de mínimo 70 W, por tanto, una fuente de 12 V con una corriente mínima de 5.8 a 6 A, debería ser usada.

3.4. Diseño de los soportes y partes impresas en 3D

Para continuar con la construcción mecánica, los soportes del motor, bolsa ambu y resto de componentes, se diseñó un modelo completo de la parte mecánica del ventilador en el software Solidworks. A continuación, se detalla cada parte diseñada y construida y su utilización dentro del prototipo.

3.4.1. Soportes de aluminio

Para que todo el prototipo se sostenga, se usó dos perfiles de aluminio de 20x80. Estos se presentan en las figuras 3.7a y 3.7b. Cabe destacar que estos perfiles están presentes en el mercado, por lo cual, se usó diseños hechos por el fabricante.



(a) Corte Frontal (b) Corte en perspectiva

Figura 3.7: Cortes en Solidworks del diseño de los perfiles de aluminio.

3.4.2. Soportes para brazos, soporte para motor y conjunto de brazos.

Luego, se diseñó los soportes para el motor y para los brazos de acero inoxidable. Como se puede apreciar en las figuras el soporte de motor, los brazos y el soporte de los brazos van unidos mediante vástagos de acero. Estos vástagos permiten que haya movimiento mientras están sujetados a los soportes de aluminio.

A continuación, se puede observar el diseño del soporte para los brazos en las figuras 3.8a y 3.8b. Así también el soporte ya construido en la figura 3.9

(a) Corte Frontal (b) Corte en perspectiva

Figura 3.8: Cortes en Solidworks del diseño del soporte para los brazos.

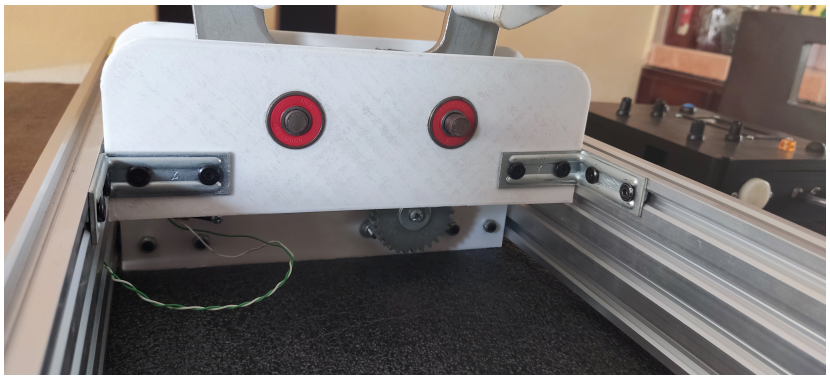
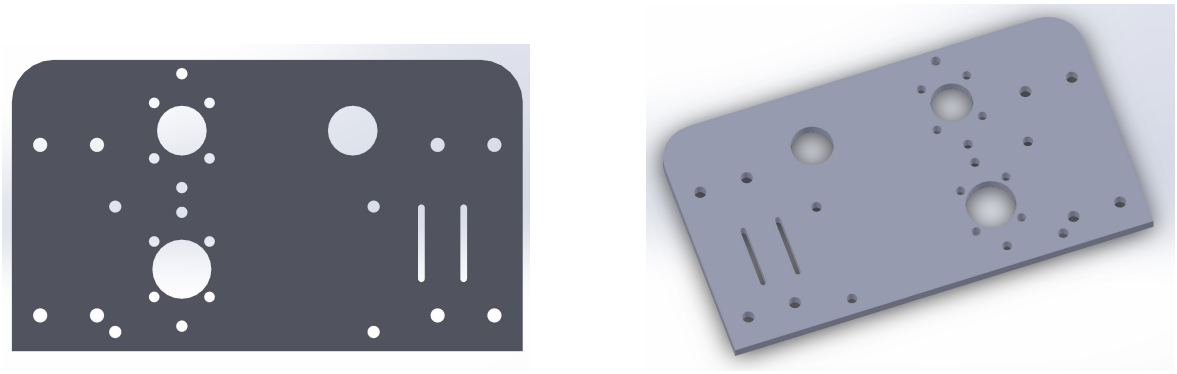


Figura 3.9: Soporte para los brazos construido.

A continuación, se diseñó el soporte para motor. Este soporte sostiene el fin carrera, el motor, los brazos con engranajes y el piñón transmisor del movimiento. A continuación, en las figuras 3.10a y 3.10b se puede observar el diseño en Solidworks y en la figura 3.11.



(a) Corte Frontal (b) Corte en perspectiva

Figura 3.10: Cortes en Solidworks del diseño del soporte para el motor.

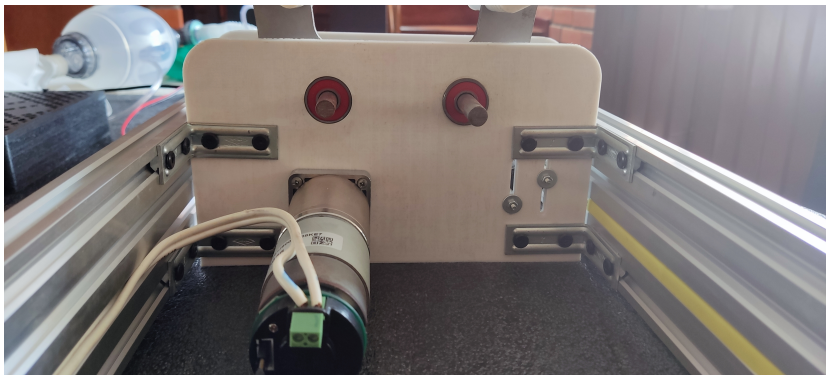
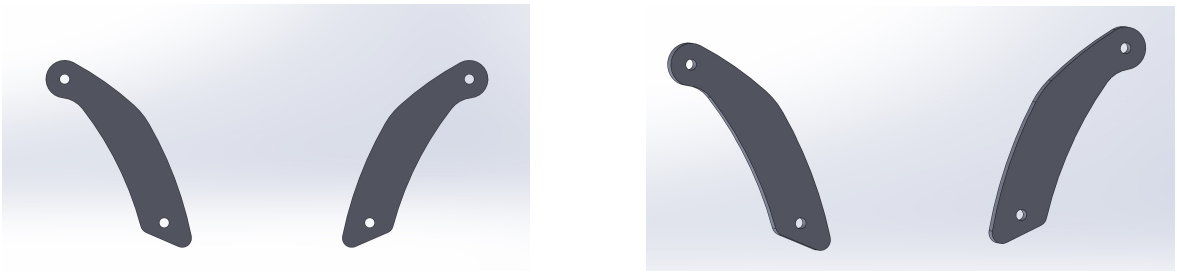


Figura 3.11: Soporte para el motor construido.

Luego, para tener una superficie completa que pueda presionar la bolsa ambu, se construyeron 30 paletas que funcionan como dedos complementarios para los brazos mecánicos. El diseño de los dedos se puede apreciar en las figuras 3.12a y 3.12b y su construcción en la figura 3.13.



(a) Corte Frontal (b) Corte en perspectiva

Figura 3.12: Cortes en Solidworks del diseño de los dedos.

Figura 3.13: Dedos construidos.

Estas 30 paletas se sujetan a los brazos mecánicos y forman un conjunto como el mostrado en la figura 3.14.

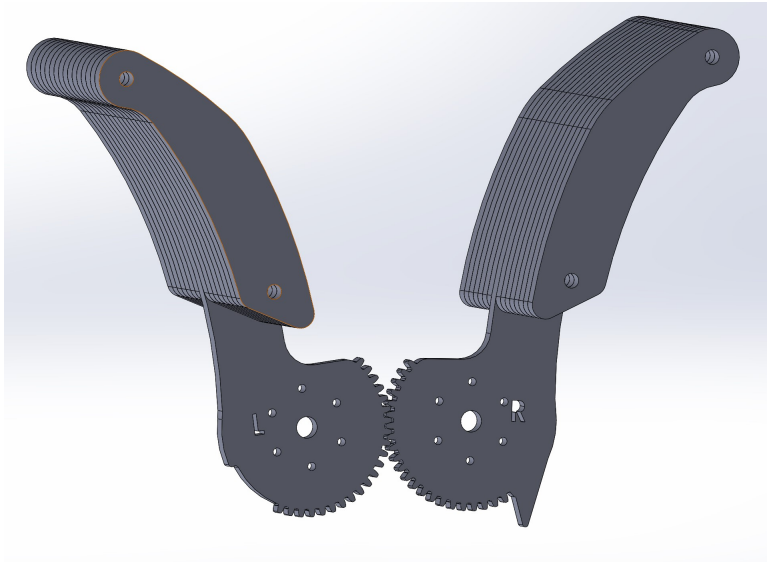
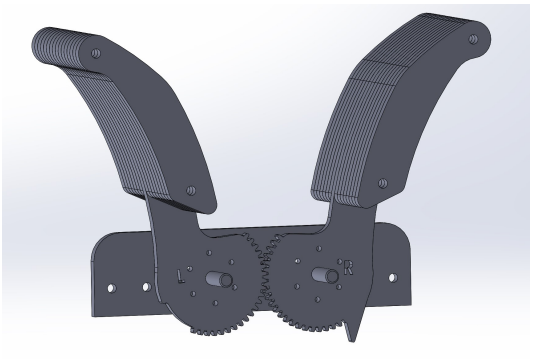
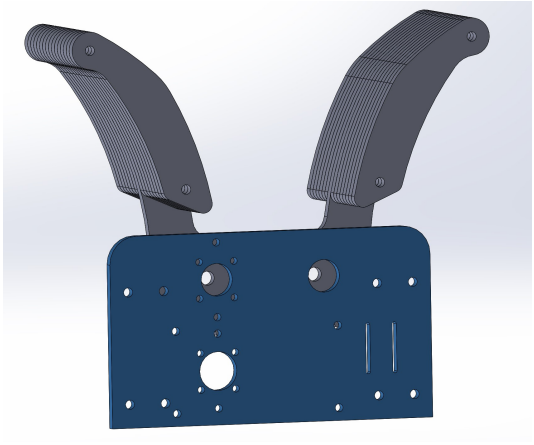


Figura 3.14: Conjunto de brazos para presionar la bolsa.

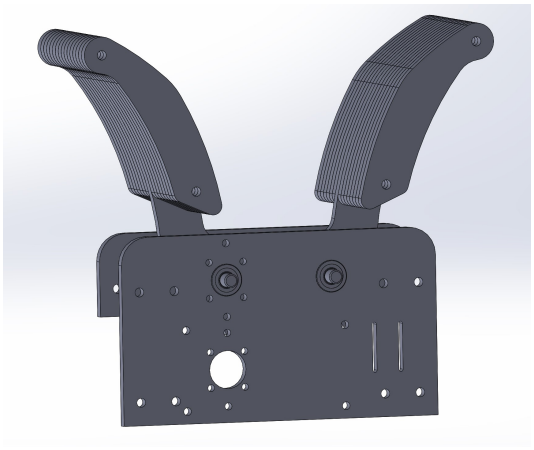
Finalmente, todo se une en un conjunto de soporte para los brazos mecánicos, motor y piñón. En las figuras 3.15a, 3.15b y 3.15c se puede observar el montaje de los soportes y los brazos con engranajes.



(a) Ensamble del soporte de los brazos con los vástagos y el conjunto de dedos y brazos.



(b) Ensamble del conjunto de brazos con el soporte para motor.

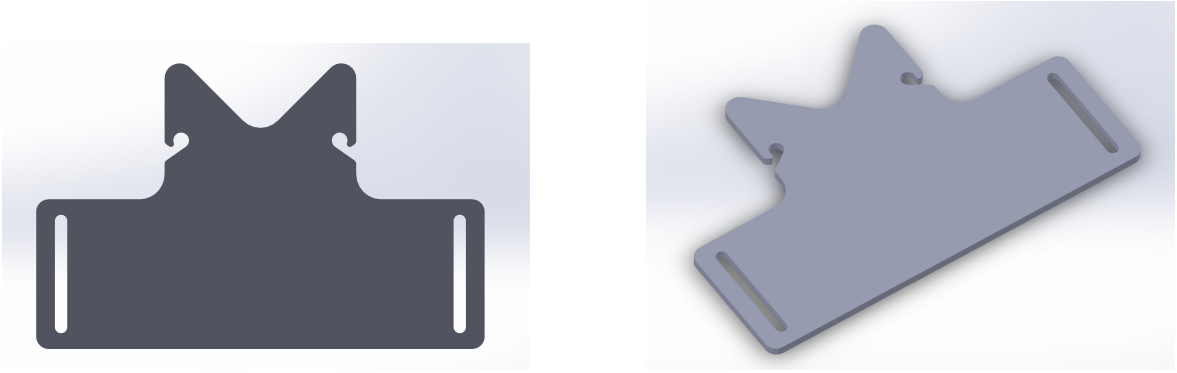


(c) Ensamble completo de los soportes y brazos.

Figura 3.15: Montaje de brazos con engranajes y soportes para brazos y motor.

3.4.3. Soportes para bolsa ambu

Los soportes para poder contener la bolsa ambu desde los extremos, fue lo último en ser diseñado y construido. Estos soportes van unidos a los perfiles de aluminio y se encuentra uno en cada extremo del ventilador. El diseño se puede apreciar en las figuras 3.16a y 3.16b y el soporte construido en la figura 3.17.



(a) Corte Frontal (b) Corte en perspectiva

Figura 3.16: Cortes en Solidworks del soporte para la bolsa ambu.

Figura 3.17: Soporte para la bolsa ambu construida.

3.5. Ensamble del ventilador

El ensamble de ventilador fue la parte final. Todas las partes descritas antes fueron sujetadas a los perfiles de aluminio mediante los soportes y tornillos necesarios. Este ensamble se puede apreciar en las figuras 3.18 y 3.19

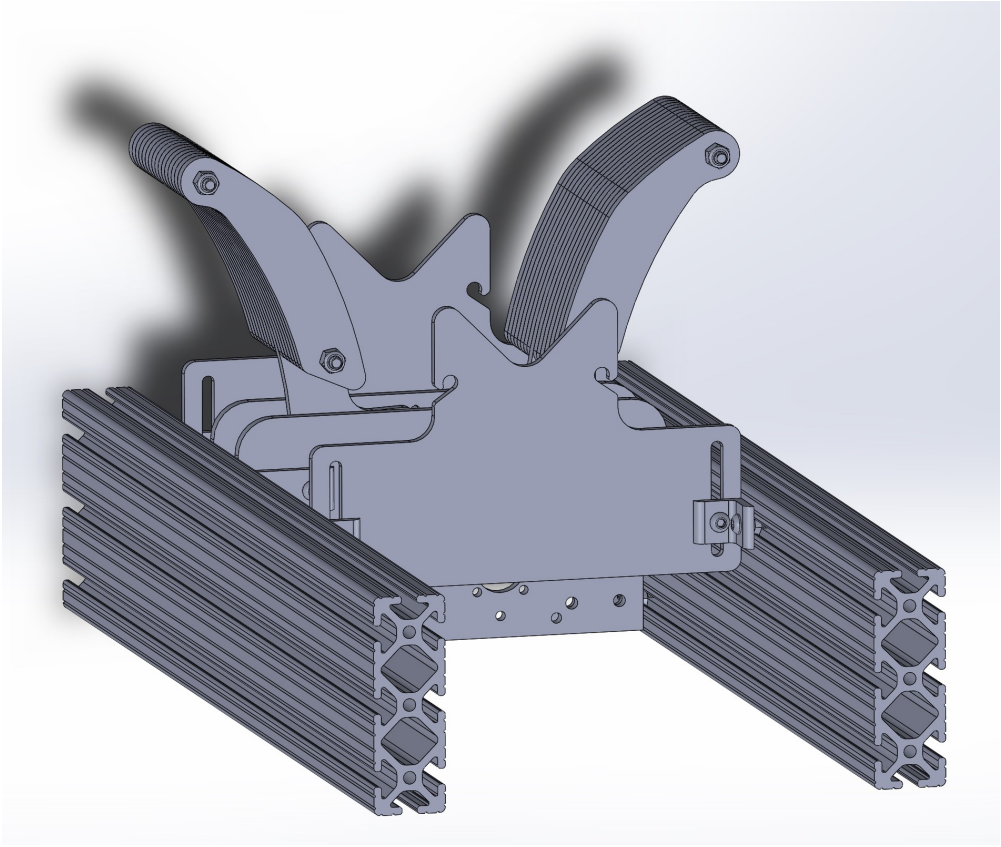


Figura 3.18: Ensamble de las partes diseñadas en solidworks.

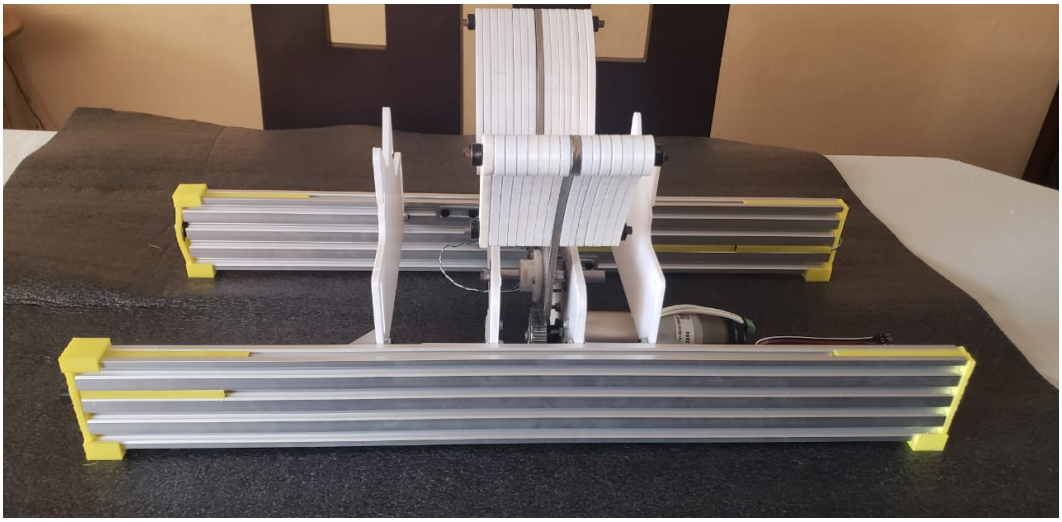


Figura 3.19: Ensamble completo del ventilador.

3.6. Sistema neumático y plomería

Como parte final del sistema mecánico, el ensamble del sistema neumático es importante. Para esto se tiene requisitos de diseño críticos del circuito de respiración del paciente. Esto detalla un problema

clave del espacio muerto, que si no se aborda, provocará que el paciente respire el CO_2 expulsado y se desoxigene rápidamente con un resultado adverso inmediato.[27]
En la figura 3.20 se muestran los bloques del sistema neumático usado para este prototipo. Este circuito muestra el circuito mejor ensamblado con resucitador de silicona, manómetro analógico manual, tubo de extensión, válvula adicional para el paciente, válvula PEEP, filtro HELP y tubo endotraqueal. Se debe considerar que la válvula del paciente en la bolsa sirve parcialmente al exhalar, pero no participa en la inhalación y todo el aire pasa a través del filtro HEPA.

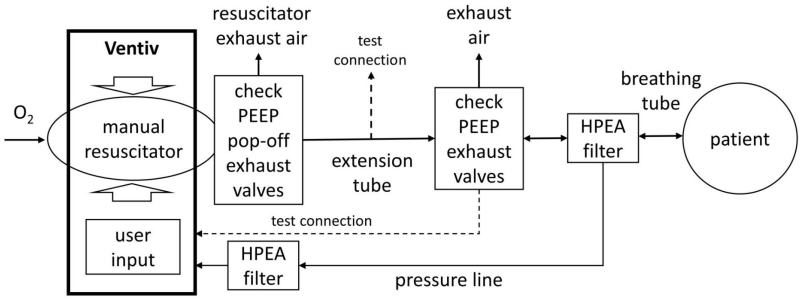


Figura 3.20: Sistema neumático implementado para el ventilador manual. Partes importantes y bloques esenciales [4].

Normalmente, los resucitadores manuales autoinflables se conectan directamente al adaptador del tubo endotraqueal del paciente. Los resucitadores manuales tienen una "válvula del paciente"que dirige la mezcla de gas oxígeno/aire al paciente y desvía el gas exhalado al medio ambiente. Integradas en la máscara de la válvula de la bolsa terminal (BVM) hay una serie de características críticas. Todo esto se puede apreciar en la figura 3.21

- Conexión y depósito de oxígeno
- Válvula de seguridad (la ubicación no es importante)
- Válvula unidireccional que guía el aire al paciente
- Válvula de exhalación (permanece cerrada mientras hay presión sobre la bolsa)
- Válvula PEEP que se instala después de la válvula de exhalación y mantiene la contrapresión
- Puerto de detección para la conexión del manómetro (lo usamos para la conexión al sensor de presión)

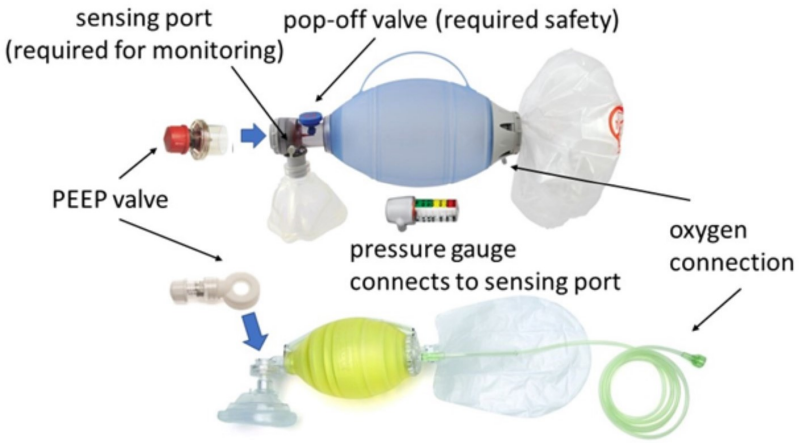


Figura 3.21: Partes del sistema neumático usado para resucitación manual [4].

Se debe considerar que las bolsas de reanimación manual no están aprobadas por la FDA para su uso como soluciones de ventilación a largo plazo. Algunas consideraciones sobre cómo se debe conectar el paciente a un ventilador basado en resucitador manual incluyen:

- El ventilador debe colocarse lo más cerca posible del paciente.
- La bolsa debe asegurarse al ventilador para evitar que un paciente despierto tire de ella o desenganche la bolsa del mecanismo. Esta es una condición de falla que debe detectarse mediante la detección de presión.
- Se debe tener cuidado para evitar que se vuelva a inhalar CO_2 debido a las mangueras largas. Un desafío fundamental es la ubicación de las válvulas unidireccionales y espiratorias, que normalmente se integran directamente en la bolsa.

Cuando los pacientes están paralizados, el efecto de parálisis puede desaparecer en ocasiones y se debe considerar cómo mantener al paciente a salvo de la desconexión o extubación involuntaria del circuito respiratorio. Por lo tanto, se necesita un método seguro para extender el alcance y la flexibilidad del resucitador manual a un paciente acostado en una cama de hospital. Si se usa un tubo simple para hacerlo, crea un problema de seguridad crítico de espacio muerto. Por ejemplo, en un tubo de 1 m de largo y 2 cm de diámetro nominal, hay un espacio muerto inaceptable de 314 ml en el que el paciente inhalará y exhalará y no se oxigenará.

El espacio muerto simplemente significa volumen en el circuito respiratorio que no participa en el intercambio de gases en los pulmones. Nuestra anatomía natural también tiene espacio muerto. Teniendo en cuenta que el intercambio de gases se produce en los alvéolos de nuestros pulmones, cada estructura anatómica por encima de ella puede considerarse espacio muerto: conductos nasales/orales, faringe, laringe, tráquea y bronquios primarios/secundarios/terciarios. La extensión del tubo a través del cual ocurre el flujo bidireccional de la mezcla de gas inhalado/exhalado solo aumenta el espacio muerto.

Encontrar una forma de acercar la "válvula del paciente" del reanimador manual al paciente es fundamental para resolver este problema. Los circuitos de ventilación estándar tienen dos ramas, una

para la inspiración y otra para la espiración, de modo que el ventilador pueda recuperar los gases. Los circuitos de ventilación de una sola rama con una válvula del paciente ubicada distalmente ya existen en el mercado, pero no están necesariamente optimizados para su uso con un reanimador manual. Resolver este problema requiere creatividad; y hasta el momento de fabricación de este prototipo, ningún fabricante de resucitadores manuales elabora una solución aprobada y ningún fabricante construye todas las piezas que se ensamblarán correctamente. En la figura 3.22 se muestra el circuito neumático armado para este prototipo.

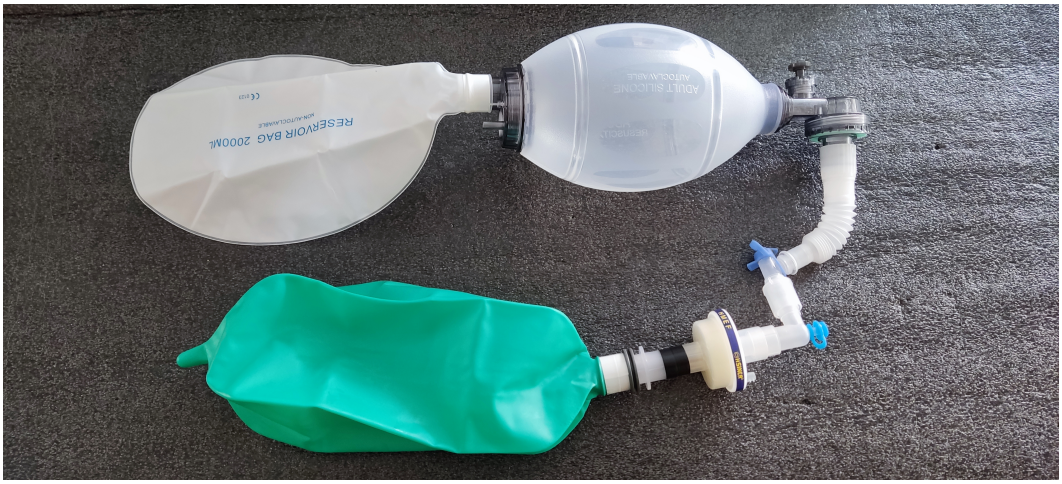


Figura 3.22: Partes básicas ensambladas para el resucitador manual.

3.7. Conclusiones

El diseño mecánico presentó varios retos en el momento de construcción. Todo esto debido a la situación de la industria ecuatoriana. Existe pocas opciones en el mercado para la construcción de los brazos con engranajes, lograr su construcción en acero inoxidable llevo tiempo y una extensa búsqueda en la industria local.

En cuanto respecta a la construcción del resto de partes, la impresión 3D aportó de gran manera para resolverlo. Debido al bajo costo de fabricación y la versatilidad de materiales a ser usados, se pudo satisfacer las características de todas las parte mecánicas.

Así mismo, en cuanto a la parte neumática se debe decir qué los soportes de las bolsas en cualquier diseño deben ser ajustables. La clave es encontrar puntos en la bolsa que brinden soporte y restricción lateral, pero que permitan que la bolsa se flexione a medida que se comprime. La bolsa debe estar centrada lateral y verticalmente entre las pinzas. Por motivos prácticos, la separación de los soportes de la bolsa a 21 cm debe ajustarse a la mayoría de las bolsas, pero los soportes deben poder ajustarse verticalmente.

También se debe tener en cuenta ciertas notas sobre la industria ecuatoriana. Al revisar los productos disponibles en el mercado, se debe acotar que:

- Ningún fabricante de bolsas suministra mangueras de extensión con los accesorios adecuados.

Las bolsas diseñadas para su reutilización, son las únicas bolsas que potencialmente pueden sobrevivir a un uso repetido. No se tiene ninguna información sobre la vida útil.

- Algunas bolsas no tienen cabezales desmontables, y tampoco incorporan válvulas PEEP. Se puede instalar válvulas externas de fácil apertura y conexión en sus diseños. Solo se pueden usar si se extienden con un cabezal y un tubo de extensión separado.
- Cuando se usa un tubo largo, sin un circuito de doble rama y una válvula unidireccional para abordar el problema del espacio muerto, esto puede afectar el volumen entregado al paciente; puede ser necesario aumentar el volumen inspirado.
- La adición del filtro HEPA provocará una caída de presión y puede afectar la configuración de PEEP.
- La firmeza de todas las conexiones es importante.
- En el peor de los casos, colocar la cabeza lo más cerca posible del paciente reducirá el espacio muerto, pero no es una solución óptima o segura, especialmente para pacientes con volumen inspiratorio reducido.

Capítulo 4

Diseño e implementación electrónica

Este capítulo abarca la metodología empleada para enfrentar la problemática expuesta, mostrando las características de los dispositivos utilizados, diseño del prototipo de ventilador en su parte eléctrica y electrónica. Así mismo, se describe los métodos utilizados para el abaratamiento de costos y la adaptación a la realidad de la industria ecuatoriana.

4.1. Introducción

Esta sección proporciona una descripción de la arquitectura del sistema, la estrategia de control y la lógica subyacente. Cabe mencionar que, las bolsas Ambu poseen gran disponibilidad en el medio, lo que las convierte en un medio conveniente para usar en intubación y entregar respiración asistida, sin embargo, las bolsas Ambu poseen pocas medidas de seguridad, contando entre ellas una válvula de liberación de presión y una válvula PEEP, que generalmente se debe configurar manualmente.

Cualquier ventilador debe poseer medición de presión y debe monitorear de forma activa tanto la presión Plateau como la Presión Pico. El ventilador posee dos modos de control:

- **Modo de control de volumen:** La respiración asistida configurada por un médico se entregan de manera constante y automática con medición de presión usada solo por seguridad. Este modo es viable solamente para pacientes que se encuentren sedados y paralizados. Volumen Tidal, Respiraciones por minuto y Tasa I:E son configurados de acuerdo a guía médica.

- **Modo de control asistido:** Cuando un paciente intenta respirar, genera una caída de presión, el sensor de presión detecta el esfuerzo del paciente y consiguientemente entrega la respiración de manera automática, es decir, el ventilador funciona en sincronismo con la respiración del paciente, asistiendo de esta manera su respiración. En este modo, las respiraciones por minuto se seleccionan por debajo del ritmo estimado de respiración del paciente, este proceso activa un temporizador que se reiniciará cada vez que el paciente realice una respiración por su propia

cuenta, si el temporizador finaliza antes, el ventilador comienza un ciclo de respiración por su cuenta.

Así mismo, se tienen los siguientes parámetros de control:

- Respiraciones por minuto (*Breaths per minute, BMP*)
- Volumen tidal (*Tidal Volume, TV*)
- Tasa de inspiración y espiración (*Inspiration to Expiration ratio, I:E*)
- Presión de disparo (Solo disponible en modo asistido)

4.2. Hardware eléctrico

El hardware usado para el desarrollo del proyecto tiene como base un Arduino Mega, este dispositivo posee una serie de señales de entrada y de salida que comandan el correcto funcionamiento del sistema en conjunto con su contra parte mecánica y médica, a continuación se describen en detalle los componentes eléctricos y electrónicos usados.

4.2.1. Componentes de entrada

Estos componentes eléctricos/electrónicos de entrada son responsables de ingresar información al sistema, esta información puede ser de configuración o de control, a continuación se describe cada componente.

4.2.1.1. Dispositivos de configuración

- Potenciómetro para configurar las respiraciones por minuto
- Potenciómetro para configurar el volumen tidal
- Potenciómetro para configurar la Tasa de inhalación/exhalación I:E
- Potenciómetro para configurar el umbral de control asistido
- Botón de confirmación de configuración
- Botón de reset/apagado
- Botón para silenciar la alarma
- Botón de apagado de emergencia

4.2.1.2. Dispositivos de control

- Sensor de presión Honeywell
- Encoder de cuadratura de realimentación
- Switch de límite para los brazos mecánicos

A continuación se describe en mayor medida cada componente.

Componentes de salida:

Estos componentes son responsables de ejecutar cierta tarea o de mostrar información sobre el sistema, se describe a continuación cada uno de ellos.

- Driver de motor RoboClaw Solo

- Motor DC engranado
- LCD Display
- Alarma

Encoder y motor:
El sistema ha sido diseñado teniendo en cuenta que se debe emplear un motor bajo un control de lazo cerrado, para información de retroalimentación al sistema se usa un encoder de cuadratura integrado, en la Sección 3.3, se realiza un cálculo de la potencia necesaria del motor en orden de generar la fuerza necesitada para comprimir la bolsa Ambu.

Fuente de poder:
De acuerdo a las necesidades provistas tanto por el circuito de control como por el circuito de potencia se han realizado los cálculos de la potencia necesaria, por tanto una fuente que suministre 12 V y 5 A, es suficiente para la correcta entrega de energía del sistema.

Cabe destacar que el comportamiento del motor en funcionamiento tiende a generar picos de voltaje, lo que se debe suprimir mediante el uso de capacitores para una protección adicional.

Controlador:
Como ya se ha mencionado en anterioridad, el control se ha realizado mediante un Arduino Mega con su respectivo microcontrolador, el motivo de usar este Arduino es debido a la gran cantidad de puertos que posee, ya sea para canales analógicos, digitales y de comunicación. Además al ser su uso de software libre, poseer gran capacidad de desarrollo de programas, soporte online bastante extendido, gran cantidad de documentación y soporte de comunidad activa, resulta una herramienta lo suficientemente poderosa como para usarla en el proyecto.

Driver de motor:
Cualquier driver que permita obtener el suficiente voltaje y corriente necesarios para satisfacer las especificaciones del motor elegido puede ser usado, de igual manera. Sin embargo, el procedimiento seguido tiene como base el uso de un controlador PWM en conjunto con un puente H, teniendo en cuenta de cumplir con los siguientes puntos:

- Comando de posición y velocidad de manera segura y robusta
- Aceptar entrada de encoder de cuadratura para retroalimentación de lazo cerrado
- Características de seguridad integradas tales como temperatura y corriente que puedan hacerse llegar al microcontrolador.
- Capacidad de soportar picos aleatorios de corriente y voltaje

Con todos los puntos mencionados anteriormente, se ha decidido hacer uso de un controlador de motor RoboClaw solo, que posee control de velocidad y posición de tipo PID, esto puede configurarse a través de una herramienta de software denominada *Motion Studio de Basic Micro*, que además permite realizar varias pruebas al motor.

Sensor de presión:
El sensor de presión recibe un voltaje proporcional a la presión que se encuentra en los pulmones del paciente, lo que permite determinar la presión máxima alcanzada durante el proceso de inspiración,

y de esta manera activar el sistema cuando el paciente haga el esfuerzo por respirar en el modo de respiración asistida, el sensor de presión debe cumplir los siguientes requisitos:

- Ser diferencial, es decir, capacidad de detectar presiones positivas y negativas.
- Rango de hasta 100 cm H_2O, se ha sobredimensionado por dos, por seguridad.
- Precisión en el orden de 0,5 cm H_2O.

Potenciómetros:

- Potenciómetro 1: Varía el volumen inspirado, permite seleccionar una oscilación angular en los brazos de la máquina, siendo el máximo movimiento permitido de aproximadamente 20°, correspondientes a la completa compresión de una bolsa Ambu para adulto, es decir, esta perilla permite la variación de compresión de la bolsa Ambu de 0 % a 100 %.
- Potenciómetro 2: Varía las respiraciones por minuto, siendo las máximas recomendadas en la Sección de referencias médicas 2.3.
- Potenciómetro 3: Varía la tasa de Inhalación:Exhalación *I:E*, dadas en la Sección de referencias médicas 2.3.
- Potenciómetro 4: Selección del límite de presión para la activación del control asistido, con las configuraciones recomendadas en la Sección de referencias médicas 2.3.

Botones:

- Botón de encendido y apagado
- Botón para silenciar temporalmente alarmas
- Botón para aplicar cambios en los potenciómetros
- Switch límite, usado para establecer una posición predeterminada de los brazos.

Display LCD:
Este display LCD muestra los valores seleccionados por las perillas de los potenciómetros, la presión medida por el sensor de presión y diferentes alarmas que pudiesen activarse mientras el sistema se encuentre en funcionamiento.

4.3. Control de alto nivel

El objetivo del control es proveer un volumen controlado de aire al paciente en un período de tiempo predeterminado, existen dos fases de control: la fase de inspiración y la fase de espiración. De igual manera, existen tres parámetros de control que ya han sido descritos en secciones anteriores:

- Volumen tidal: Es el volumen total de aire que será entregado al paciente.
- Respiraciones por minuto: Denominado también *Respiratory Rate (RR)* o Ratio de respiración, que generalmente varía de 8 a 30 respiraciones por minuto.
- Tasa de inhalación a exhalación (I:E): Es la tasa de duración de la inhalación a la duración de la exhalación, típicamente, varía de 1:1 a 1:3, con un máximo de 1:4, en pacientes con COVID-19 según reportes de hospitales [17].
- Límite para el control asistido: Especifica la presión debajo de la presión PEEP para activar un ciclo de respiración accionado por el esfuerzo del paciente.

En adición a los parámetros mencionados, el controlador usa dos parámetros más, la posición del encoder del motor y la presión del sistema. El funcionamiento a grandes rasgos del controlador posee dos etapas, la primera realiza un control de alto nivel, en donde con todos los parámetros mencionados en anterioridad se realiza un cálculo, el cual tiene como finalidad encontrar dos entradas, posición y velocidad, los cuales serán entregados a un controlador de bajo nivel que consiguientemente comandará directamente al motor.

Cabe mencionar que en ningún momento el dispositivo hace una medición directa del volumen de aire entregado, por tanto, el Volumen Tidal, se especifica como el porcentaje de la compresión total de la bolsa Ambu en lugar de en litros, este porcentaje de compresión, el cual varía de 0 a 100 %, es a su vez codificado en pulsos que se traducen al movimiento de los dedos compresores de la bolsa y a su vez permiten estimar el volumen de aire entregado.

4.3.1. Cálculo de la forma de onda y su duración

La forma de onda de un período respiratorio de un paciente común se estima como en la Figura 4.1, como ya se mencionó, esta forma de onda estará dada en función de tres variables configurables: volumen tidal V_T, respiraciones por minuto o *Breaths per Minute* BPM y tasa de inhalación a exhalación o *Inhalation to Exhalation Rate* IE, con esto en cuenta el controlador determina las duraciones mostradas en la Figura 4.1, como sigue a continuación [5]:

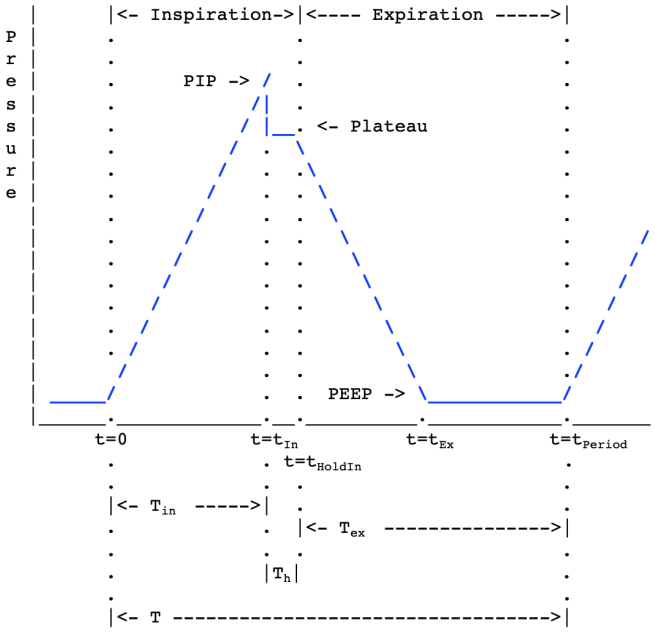


Figura 4.1: Forma de onda de presión en ventilación de volumen mostrando tiempos y duraciones [5].

Período (T): Es la duración en segundos de un ciclo de inhalación exhalación.

$$T = \frac{60}{BPM} \tag{4.1}$$

T_h**:** Es la duración en segundos de una retención inspiratoria o pausa al final de una inhalación

durante la fase de inspiración para la presión Plateau, en el sistema desarrollado es una constante más no una variable configurable.

T_{in}**:** Es la duración en segundos del tiempo de inhalación en la fase de inspiración.

$$T_{in} = \frac{T}{(1 + IE) - T_H} \tag{4.2}$$

T_{ex}**:** Es la duración en segundos de la fase espiratoria.

$$T_{ex} = T - (T_{in} + T_h) \tag{4.3}$$

En adición a los parámetros dependientes del tiempo, el controlador determina las siguientes tasas de rotación:

V_{in}**:** Es la tasa de rotación de la fase de inspiración, se mide en pulsos por segundo.

$$V_{in} = \frac{V_T}{T_{in}} \tag{4.4}$$

V_e**:** Es la tasa de rotación de los dedos compresores de la bolsa en la fase de inspiración, se mide en pulsos por segundo; cabe destacar que durante la exhalación, el dispositivo no controla la velocidad de flujo que sale del paciente, esta velocidad es sencillamente la velocidad de los dedos abriéndose, y no se encuentra relacionada con la tasa de flujo espiratorio.

Existen también cuatro parámetros correspondientes a presiones medibles que deben ser tomados en consideración:

$P_{\text{máx}}$**:** Es la presión máxima permitida (fija en 40 cmH_2O), por seguridad los circuitos de entubación poseen algún mecanismo que físicamente impida sobrepasar dicha presión.

PIP:: Es la presión inspiratoria pico, correspondiente a la presión máxima alcanzada durante el período de inhalación. Por seguridad, se considera un máximo de 40 cmH_2O), que como ya se ha mencionado, corresponde a la presión máxima permitida en ciertos circuitos de entubación.

P_{plat}**:** Es la presión plateau de la inhalación, generalmente suele ser un valor importante para los médicos.

PEEP: Es la presión de final de espiración positiva, que es una presion residual en el sistema luego de la exhalación, este valor no se controla de forma directa en este sistema, aunque generalmente, puede controlarse mediante una válvula PEEP presente en la bolsa Ambu o en el circuito de entubación.

Una vez definidos los parámetros, se muestra a continuación el proceso seguido para realizar el control de la máquina mediante las Maquinas de Estados de cada modo de funcionamiento.

4.3.2. Máquina de estados del modo de control de Volumen

Se define al tiempo t, como la cantidad de tiempo que se demora terminar un ciclo de respiración (inhalación y exhalación), los tiempos y lecturas de presiones de la Figura 4.1, son usados para generar una máquina de estados que es capaz de cambiar entre fases en el bucle de control, la máquina de estados para el modo de control de volumen puede observarse en la Figura 4.2.

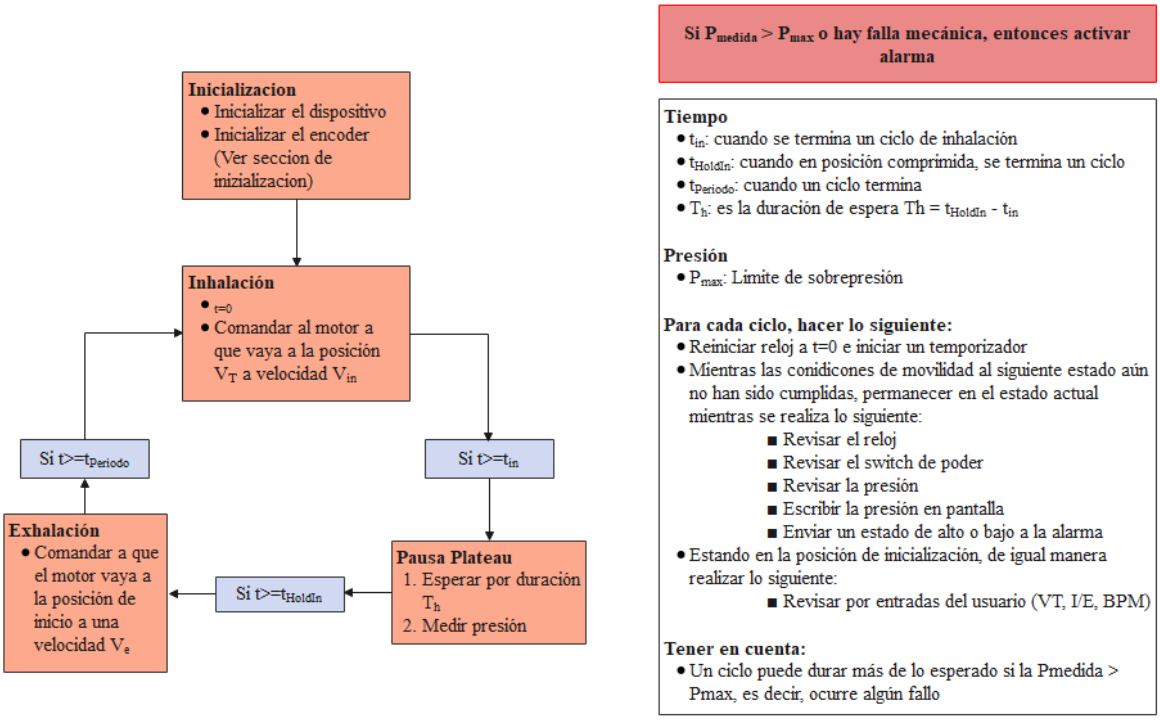


Figura 4.2: Máquina de estados para el modo de control de volumen en un ciclo de respiración.

Durante la fase de configuración, se inicializa el programa, se inicializan las comunicaciones seriales con el encoder del motor y se posiciona en el estado de inicialización al encoder, el proceso de inicialización puede observarse en la máquina de estados correspondiente a la inicialización en la Figura 4.3.

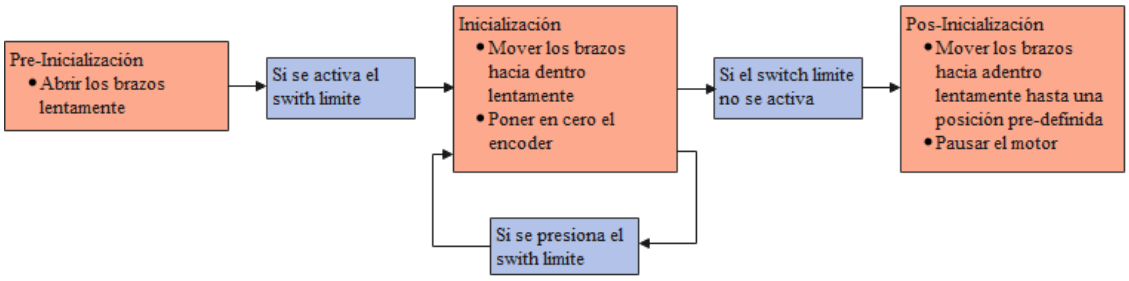


Figura 4.3: Máquina de estados para el proceso de inicialización.

En el proceso de inicialización, Figura 4.3, se ha desarrollado el proceso previo al funcionamiento como tal de la máquina, primeramente, se abren los brazos mecánicos hasta activar el switch límite, una vez activado el switch límite, se cierran los brazos lentamente y se encera el encoder, de darse el caso en donde no se haya activado el switch limitador, se cerrarán los brazos mecánicos hasta una posición predefinida en el código y se pausará el motor. Este proceso es necesario para asegurar que el funcionamiento mecánico de la máquina se encuentre correctamente.

En la fase de inspiración, primero se pone el tiempo t en cero, a continuación se comanda al

motor para que se mueva a una posición V_T a una velocidad V_{in}, pasado un tiempo T_{in}, se pasa a un estado de pausa. En el estado de pausa se espera por un tiempo T_h y se mide la presión Plateau, a continuación se cambia al estado de exhalación. Finalmente, en el estado de espiración, se comanda al motor a que se dirija a la posición 0 o inicial, a una velocidad V_e, pasado un tiempo T_{ex}, se pasa nuevamente al estado de la fase de inspiración repitiéndose el proceso para cada ciclo de respiración.

Cada momento en donde los brazos mecánicos se cierran para apretar la bolsa Ambu, se ha implementado una pausa de 0.1 segundos antes de que se vuelvan a abrir. Esta pausa no afecta la tasa I:E, y es necesaria para mantener el aire dentro del paciente por un período de tiempo. En esta fase de espera, se mide la presión de las vías respiratorias y se muestra en pantalla. Dicha presión medida, servirá de referencia para los médicos al momento de tomar decisiones para el paciente, la siguiente medición de presión se realizará una vez se haya completado el ciclo.

4.3.2.1. Máquina de estados para el modo de control asistido

El modo de control asistido difiere del modo de control de volumen en que el estado de exhalación se divide en tres subestados adicionales, la máquina de estados para el modo de control asistido puede observarse en la Figura 4.4.

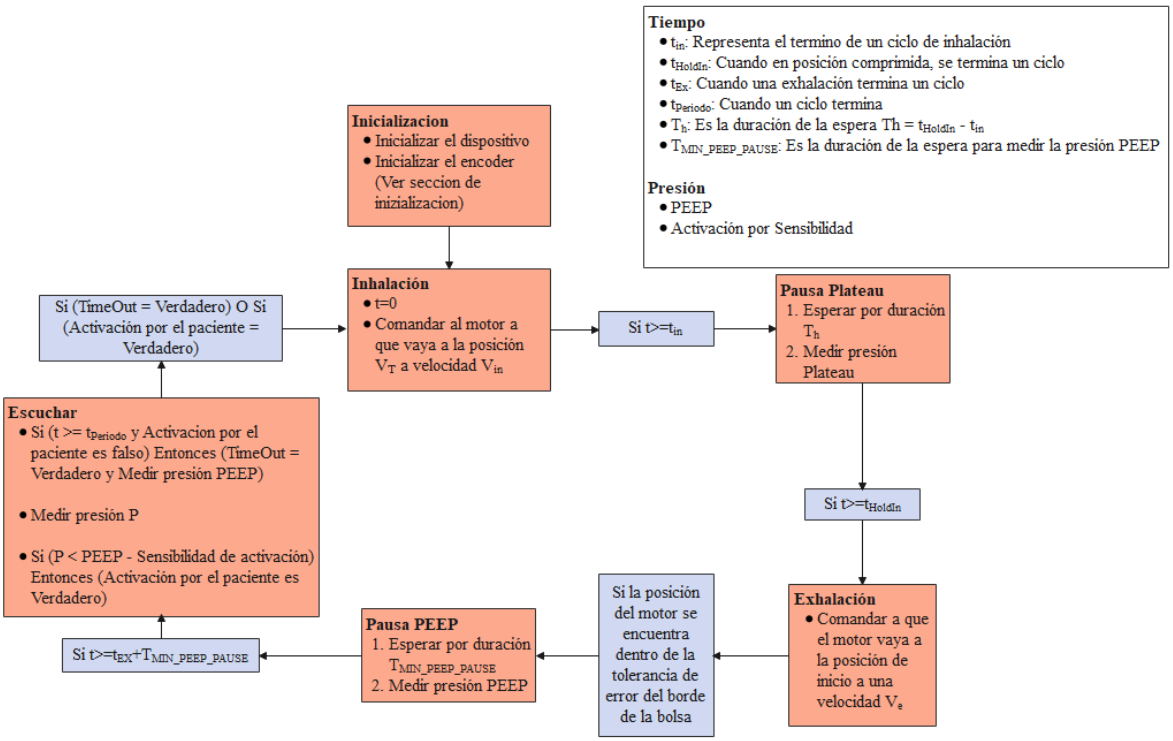


Figura 4.4: Máquina de estados para el proceso de inicialización.

En el primer subestado de exhalación, los brazos se mueven a su posición inicial hasta el borde de la bolsa Ambu, en el segundo subestado, se realiza una pausa en la exhalación por un pequeño periodo de tiempo, aquí se mide la presión PEEP, en el tercer subestado se espera a, ya sea la activación realizada

por el esfuerzo del paciente para que a su vez se inicialice el estado de inhalación, o al *timeout* de la activación por esfuerzo de paciente, que consiguientemente activa la inhalación automática como el caso del modo de control de volumen.

El programa del control descrito en esta sección puede observarse en el anexo A.

4.4. Casos de Uso

El funcionamiento general del prototipo de ventilador mecánico construido, puede observarse en la Figura 4.5, en ella se puede observar una gráfica UML de casos de uso, se intenta ser lo más simplista en la explicación del funcionamiento del sistema.

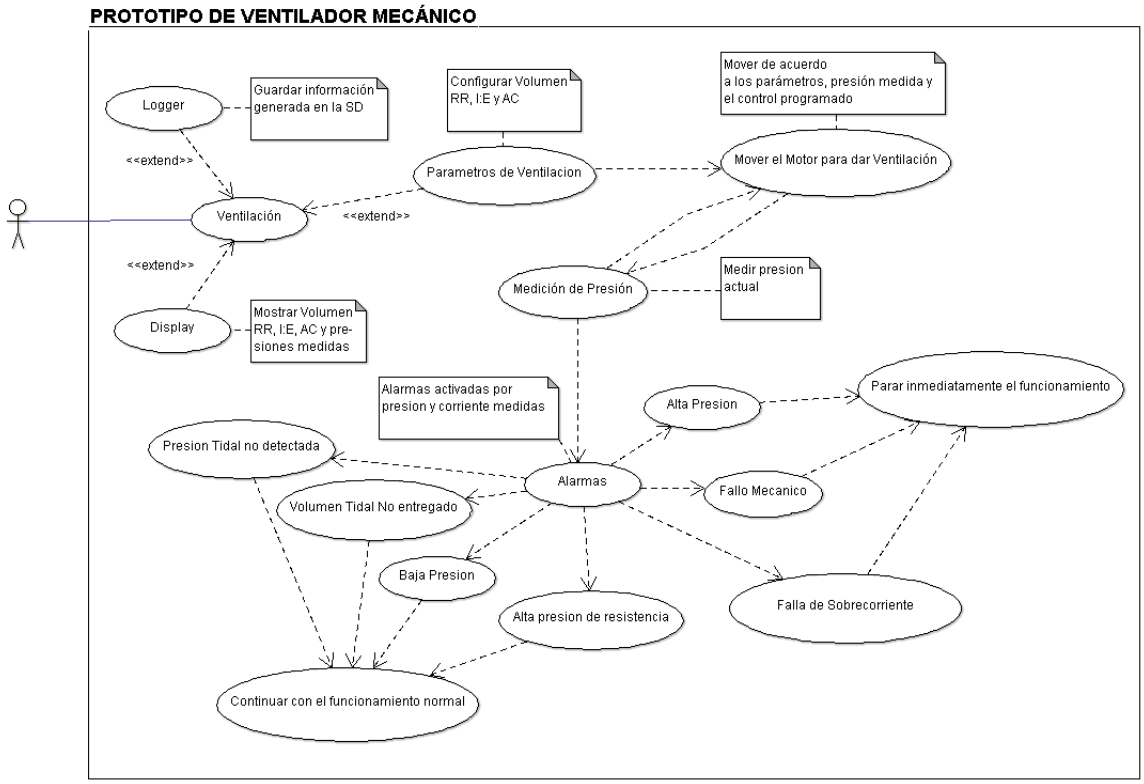


Figura 4.5: Diagrama de uso para el prototipo de ventilador construido.

El usuario, el cual generalmente será el médico tratante o un profesional calificado, será el usuario del dispositivo construido. El usuario requiere obtener del dispositivo la respiración asistida para un paciente, para esto se deben configurar en el dispositivo 4 parámetros:

- Volumen de aire entregado
- Tasa de respiración entregada (RR)
- Tasa inhalación a exhalación (I:E)
- Control asistido (Presión de disparo)

Cabe mencionar que el dispositivo al iniciarse tomará los parámetros que se encuentren configurados en ese momento y comenzará su funcionamiento sin la necesidad de que se realice algún otro proceso,

por tanto es necesario que la configuración preliminar se realice con el paciente desconectado.

Mientras el dispositivo se encuentra funcionando y si se desea modificar los parámetros del sistema, se debe presionar el botón de pare, esto hará que el dispositivo deje de funcionar, en este estado, se puede realizar la configuración de parámetros, una vez se hayan configurado, se debe presionar el botón *set*, y la máquina comenzará a funcionar con los parámetros configurados.

Mientras el prototipo está en funcionamiento, en la micro SD, se guarda información de las alarmas que se han activado a lo largo del funcionamiento de la máquina, esto en orden de que cualquier técnico pueda diagnosticar estados alterados del funcionamiento de la máquina.

Como ya es obvio, el control actuará moviendo el motor, el cuál moverá los brazos mecánicos para presionar la bolsa de acuerdo a los parámetros configurados y teniendo como retroalimentación las mediciones de presión obtenidas.

A lo largo del funcionamiento del dispositivo, varios estados del paciente o la máquina pueden activar una serie de alarmas, la descripción y el motivo de la activación de las alarmas se realiza en la sección 4.5, dependiendo de la alarma activada la máquina responderá en consecuencia, ya sea activando una indicación sonora o parando el funcionamiento del dispositivo, adicionalmente, las alarmas entregan información al médico tratante, el cual deberá actuar de acuerdo a su criterio médico.

4.5. Medición de presión y alarmas

Hay que hacer un especial énfasis en la importancia de la medición de presión y monitoreo de un paciente, aunque sea de la forma más simple posible en los escenarios de ventilación más elementales.

El sensor de presión, debe ser conectado en una posición en donde permita la medición de presión de las vías respiratorias del paciente, como se ha mencionado en 2.5. En la emergencia sanitaria que se está dando en estos momentos, transductores con las especificaciones correctas para aplicación en vías respiratorias se encuentran bajos en oferta, además, estos dispositivos deberán no ser de un único uso.

Para protección del sensor de presión, se recomienda conectarlo a un filtro HEPA y este a su vez deberá estar conectado en el circuito neumático del paciente, esto en orden de impedir migración viral entre pacientes.

4.5.1. Mediciones esenciales

A continuación se muestran mediciones esenciales que deberá realizar el prototipo, para que el médico esté en la capacidad de continuar con el correcto curso de tratamiento para el paciente:

- Medición de la presión PEEP (Presión de final de espiración positiva), generalmente se configura manualmente en la válvula PEEP
- Presión Plateau, varía en función de la resistencia presente en las vías respiratorias

- PIP (Presión inspiratoria pico), varía en función de la resistencia presente en las vías respiratorias
- Detección de inhalación, corresponde a una caída de presión, relacionada con la presión PEEP en reposo, indica que un paciente no paralizado y sedado está intentando respirar y que la máquina en cuestión deberá iniciar un ciclo en respiración (esta presión no es esencial, pero en el caso del ventilador desarrollado se convierte en una condición de control adicional)

4.5.2. Verificaciones de importancia y alarmas

En cada ciclo de respiración se deberá verificar consistencia entre las relaciones que existen entre las presiones PEEP, Plateau y PIP, si llegase a existir alguna discrepancia entre estas presiones se analiza la posibilidad de activar alguna alarma [28].

1. En el Modo de control de volumen se espera que la presión PIP sea mayor a la presión Plateau
2. Se espera que la presión Plateau sea mayor a la PEEP en orden de tener una presión de distensión positiva, que es la resta entre la presión Plateau menos la presión PEEP
3. La presión PEEP no debería ser menor a un valor específico, para prevenir colapso alveolar y dicho valor deberá coincidir con el valor configurado manualmente en la válvula PEEP
4. La presión Plateau no deberá exceder un valor predeterminado, nominalmente 30 cm H_2O en pacientes con SDRA
5. La presión PIP no deberá exceder un valor predeterminado, generalmente 30 cm H_2O. Este valor deberá ser configurado en el programa, ya que si se superase pudiese dañarse la válvula de escape
6. Alguna fuga o desconexión deberá ser detectada por el sistema, ya que la presión Plateau caerá por debajo de cierto valor, en el caso de este ventilador 5cm H_2O

4.5.3. Listado de alarmas

Las alarmas que se han configurado en este dispositivo han sido desarrolladas siguiendo estándares de ventiladores mecánicos presentes en el mercado, así como de la opinión de médicos y enfermeras de cuáles son las alarmas más importantes en un ambiente de respiración asistida y cómo deberían ser tratadas y abordadas.

Se ha seguido un patrón para silenciar las alarmas, el cual es mostrado a continuación

- El botón de silenciar alarma deberá silenciar solamente el sonido de la misma, más no la notificación visual mostrada en pantalla
- Todas las alarmas deberán ser silenciadas dentro de dos minutos, aunque una nueva alarma sea activada, la alarma silenciada deberá ser mantenida
- Presionar el botón de silenciar alarma una segunda vez, antes de que los dos minutos hayan trascurrido, deberá activar nuevamente el sonido de las alarmas
- Si múltiples alarmas se encuentran activadas, deberán ser mostradas mediante un indicador LED, en pantalla, los mensajes deberán mostrarse uno tras otro.

La tabla 5.1, desarrolla en más detalle cada una de las alarmas configuradas para el dispositivo.

Tabla 4.1: Listado de Alarmas

Nombre de la Alarma	Detección	Condición de no activación	Tono	Respuesta	Mensaje mostrado
Presión PEEP excedida	P>$P_{máx}$ = 40 cmH_2O Esto asegura que la unidad nunca puede entregar aire a una presión superior que 40 cmH_2O para mitigar el riesgo de barotrauma	Un completo ciclo de respiración sin sobre-presión	Todo de emergencia	Inmediatamente parar lacompresión, abrir completamente los brazos mecánicos, reanudar el funcionamiento normal al inicio de un ciclo de respiración	“ALTA PRESIÓN"
Baja presión	P_{plat} < $P_{plat\_mín}$ = 5cmH_20 Esta alarma detecta cualquier desconexión o fugas y notifica al médico que revise el tubo de respiración	Presión Plateau medida es normal	Tono de emergencia	Continuar con el funcionamiento normal	“PRESION BAJA DESCONEXION?"
Presión de resistencia alta	$P_{pip}-$ P_{Plat} > $P_{resist\_máx}$ = 10cmH_20 Esta alarma notifica que existe una resistencia inusual en el circuito neumático o en las vías respiratorias del paciente	Medición de la presión de resistencia en rango normal	Chirrido de notificación	Continuar con el funcionamiento normal	“PRESION DE RESITENCIA ALTA"
Falla de sobrecorriente	$I \geq I_{máx}$ = 4,5A Esta alarma detecta al motor siendo demasiado presionado. Pudiese tratarse de algo trabado en el mecanismo o de un bloqueo en el tubo de respiración	Un ciclo de respiración completo sin un evento de sobrecorriente	Tono de emergencia	Inmediatamente para la compresión, abrir completamente los brazos mecánicos, reanudar funcionamiento normal en un nuevo ciclo de respiración	“FALLA DE SOBRECORRIENTE"
Volumen Tidal no entregado	V_{final} < $V_{set} - V_{threshold}$ = 50mL	Volumen Tidal entregado en el rango incorrecto	Tono de emergencia	Continuar con el funcionamiento normal	“VOLUMEN TIDAL NO ALCANZADO"
Presión Tidal no detectada	$P_{peak}-$ P_{PEEP} < $P_{Tidal\_mín}$ = 5cmH_20 (2 *ciclos*) Esta alarma detecta situaciones en donde la presión no refleja respiración, pudiendo darse sin ningún tipo de fuga, por ejemplo, la bolsa Ambu fue removida de la máquina	Un ciclo de respiración completo sin un fallo de Presión Tidal	Tono de emergencia	Continuar con el funcionamiento normal	“NO VOLUMEN TIDAL"

4.6. Armado y construcción del circuito

Todos los parámetros de diseño antes mencionados fueron usados para el armado y construcción del circuito. Se cargó el archivo de programa en la plataforma Arduino Mega, dicho código se puede apreciar en el anexo B. Finalmente, se construyo un PCB para la conexión de botones, perillas y elementos de alarma, a continuación se presenta el circuito simulado y construido.

4.7. Diagrama del circuito

El diagrama del circuito, como ya se ha mencionado antes, tiene como base un Arduino Mega, al cual se encuentran conectados todos los demás dispositivos electrónicos y su respectiva fuente de poder, el Arduino tiene conectados a los potenciómetros, botones, luces, alarma, switch de fin carrera, display, sensor, encoder y un slot micro SD. Todos los componentes se encuentran conectados como se muestra en la Figura 4.6.

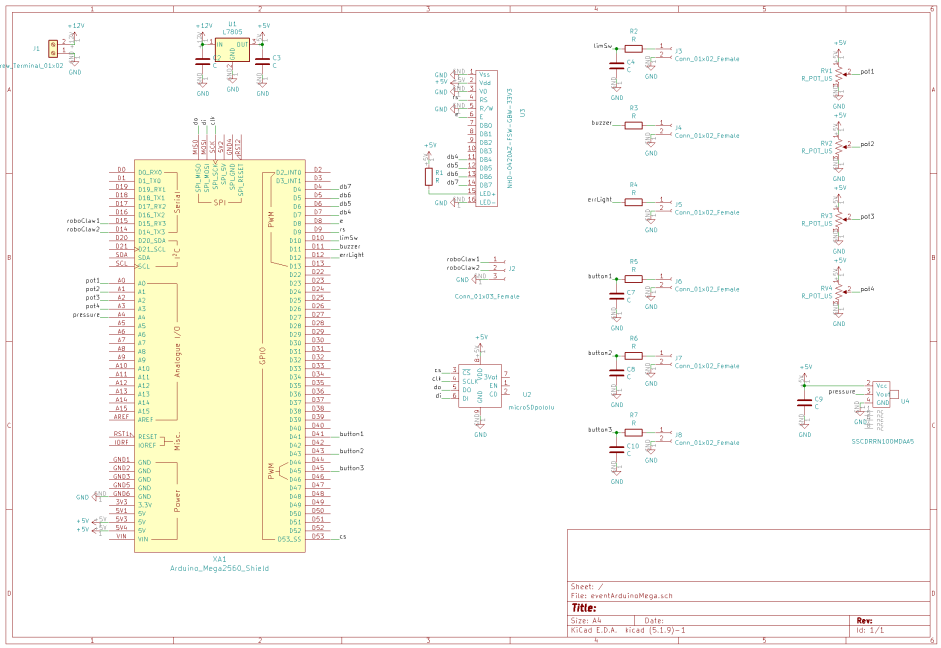


Figura 4.6: Circuito de control esquemático del ventilador con base en un Arduino Mega.

4.8. Placa PCB del control

Partiendo del circuito esquemático, mostrado en la Figura 4.6, se ha diseñado el respectivo PCB, teniendo particularmente en cuenta el tamaño del dispositivo, ya que al ser usado en situaciones en donde se requiere movilidad, el tamaño del mismo deberá ser lo más pequeño y robusto posible. El PCB del circuito puede observarse en la Figura 4.7.

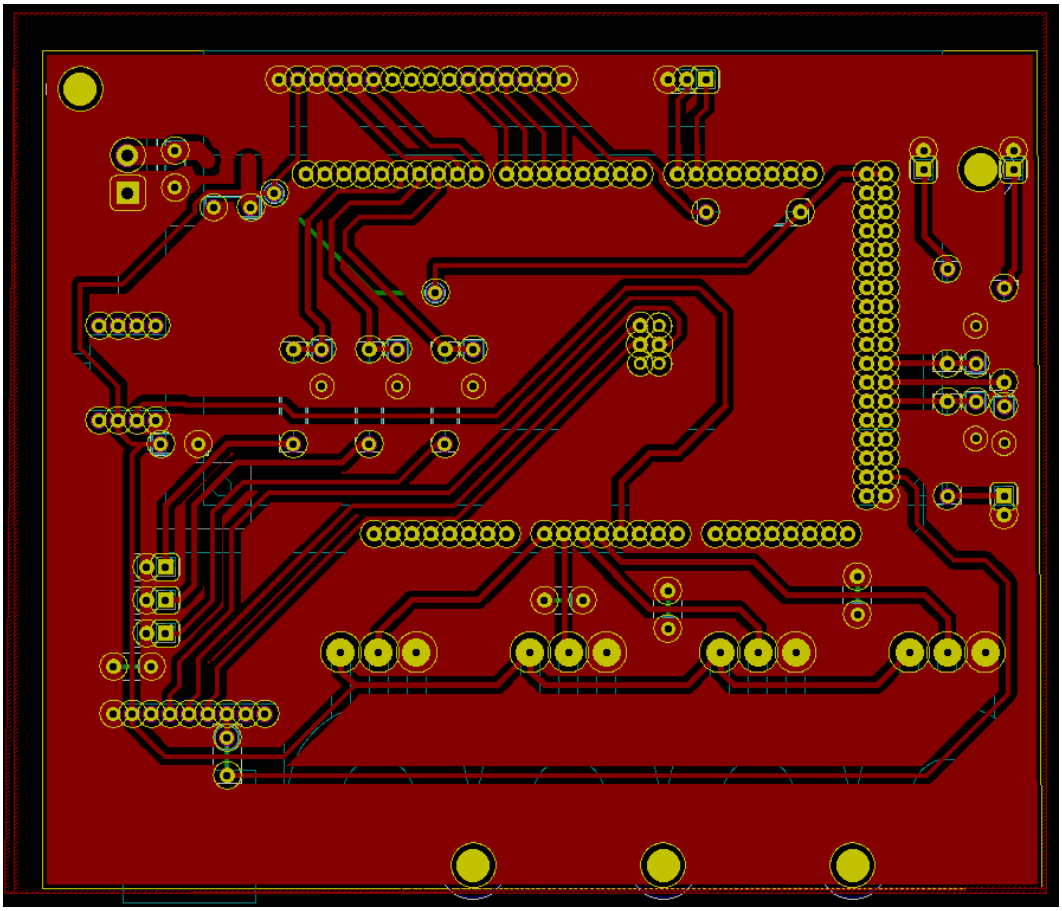
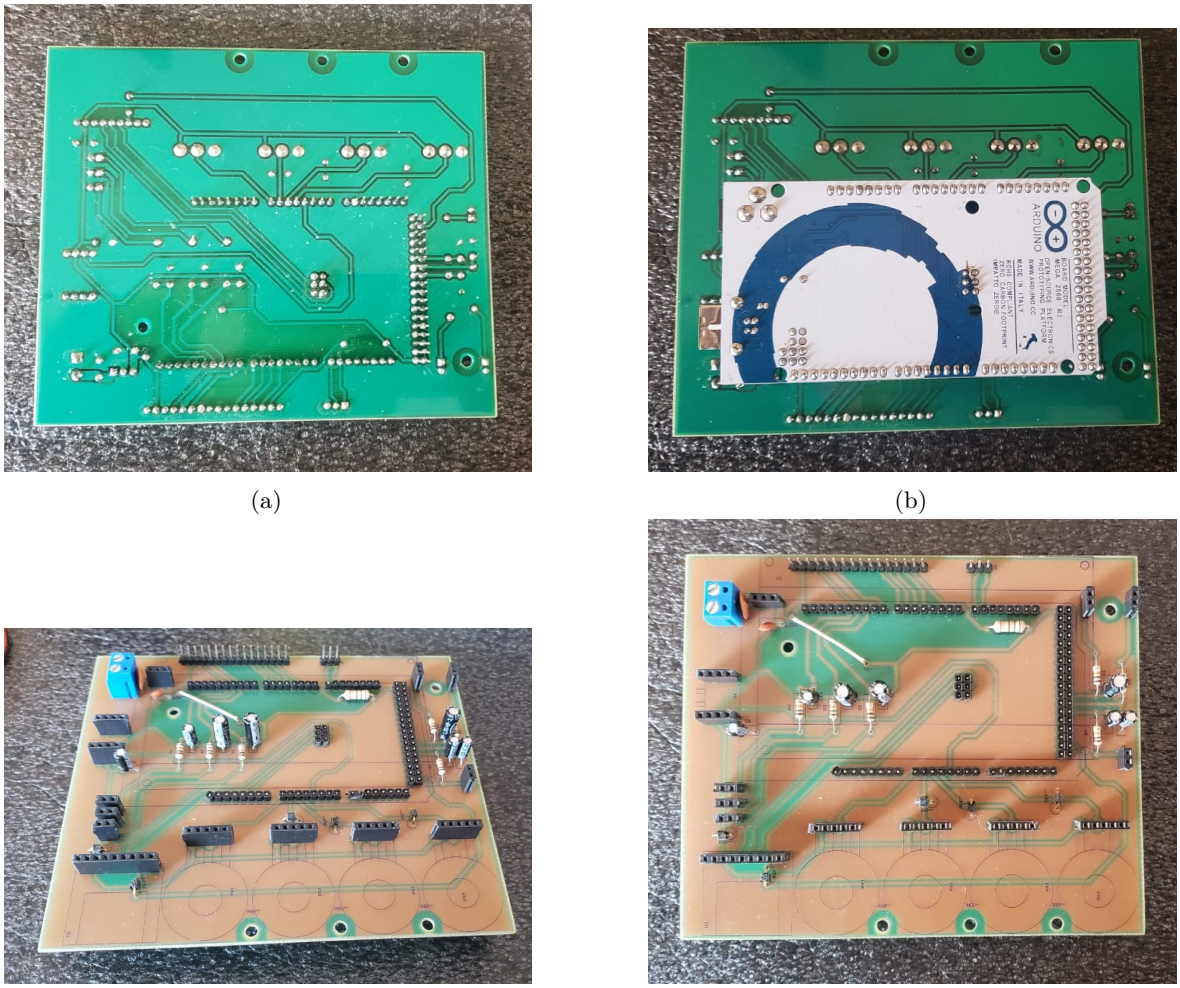


Figura 4.7: Circuito de control PCB del ventilador con base en un Arduino Mega.

Finalmente, en las figuras 4.8a, 4.8b, 4.8c y 4.8d se puede apreciar el PCB construido. Como se puede observar todo el proceso de control se lleva en el Arduino, el resto del circuito se dedica a la conexión con los dispositivos de entrada y salida de información.



(a) (b)

(c) (d)

Figura 4.8: Distintas vistas del circuito PCB construido.

4.9. Conclusiones

El control tiene base en dos dispositivos fundamentales para este prototipo, la placa Arduino Mega y el driver de motor Roboclaw Solo, ambos realizan tareas que en su conjunto marcan el correcto funcionamiento del dispositivo.

La placa Arduino Mega tiene programado el control de alto nivel, dicho control obtiene, partiendo de gráficas de respiración que a su vez definen ecuaciones de control, parámetros indispensables como tiempos de funcionamiento de cada ciclo, que a su vez afectan a las tasas de respiración de los pacientes, rangos de movimiento de los brazos mecánicos para lograr obtener volúmenes de aire requeridos, presiones indispensables en la ventilación mecánica de los pacientes etc. Estos parámetros se conjugan en dos variables, posición y velocidad, que a su vez son pasados al driver Roboclaw, el cual, internamente tiene ya programados controles PID para posición y velocidad, además posee un software con interfaz de alto nivel, que permite la configuración del driver en simples pasos, este control es el responsable de comandar directamente el movimiento del motor.

En el ámbito local es fácil conseguir las placas Arduino, sin embargo, conseguir el driver Roboclaw, así como el motor con encoder usado para este proyecto no es posible, se pueden considerar otras opciones disponibles en el mercado, sin embargo, se deberá ajustar el código de Arduino al funcionamiento del nuevo driver a usar, inclusive, de ser necesario deberá ser programado el control PID como requiera las nuevas opciones escogidas.

La fuente de poder también es de importancia, mientras se realizaban pruebas, había ocasiones en donde el motor requería de un suministro de energía elevado, haciendo que la fuente conectada a él se bloqueara por integridad de la misma fuente, esto llevó a que se usaran dos fuentes distintas, una usada exclusivamente para el motor y otra usada exclusivamente para el apartado de control, hay que tener en cuenta los procedimientos necesarios para el funcionamiento en conjunto de ambas fuentes en un sistema que engloba todo el prototipo.

Es importante mencionar que el funcionamiento del prototipo tiene contemplada la ventilación por volumen de aire y la ventilación asistida solamente, ambos tipos de funcionamiento son los más comunes en un ambiente de ventilación asistida, sin embargo, otros ventiladores del medio con prestaciones más altas, son capaces de comandar más variables de entrada, lo que permite variar el funcionamiento intrínseco del ventilador, por tanto, como ya se ha mencionado, el prototipo construido intenta solventar necesidades presentes en ambientes de emergencia saturados por la presente emergencia sanitaria. El prototipo construido no viene a reemplazar los ventiladores más sofisticados del medio, sino más bien, sirve como un ventilador de transición para pacientes que empeoran o mejoran su estado de salud o para pacientes que necesitan de un ventilador y no disponen de uno en el preciso momento, por ello, es indispensable que, cuando el prototipo se encuentre en funcionamiento, se tenga un profesional de la salud monitoreando en todo momento el estado del paciente.

Capítulo 5

Experimentos y resultados

En este capítulo se analizan las mediciones de parámetros de desempeño obtenidas por el MIT, grupo liberador de la patente, y los parámetros que se obtuvieron como resultado de la puesta en marcha del prototipo de ventilador mecánico. Se realizan mediciones de los valores de presión PEEP, Plateu y presión máxima. Así mismo, se muestran mediciones de volumen en función de la bolsa de ventilación manual usada, resultados más detallados se presentan en el anexo D. Finalmente, se muestra una comparación de todos estos parámetros con el modelo ventilador OxyMag y el Astral ResMed 100-150.

5.1. MIT Antecedentes: estudios porcinos y pruebas de banco

El equipo del MIT presentó su prototipo con resultados trabajados con laboratorios de experimentación con animales. Dichos laboratorios están certificados para realizar estudios en animales bajo los protocolos aprobados por IACUC (Comité Institucional de Cuidado y Uso de Animales). Para estos estudios eligieron un modelo porcino, ya que los cerdos tienen un sistema respiratorio muy similar al de los seres humanos. Esto les fue esencial para evaluar el rendimiento y la seguridad del sistema.

A continuación, se presenta una explicación detallada de todo el proceso que el equipo del MIT llevó a cabo para determinar los valores de volúmenes, presiones y factores mecánicos y neumáticos óptimos para un prototipo totalmente funcional.

5.1.1. Pruebas de banco

Las pruebas y experimentación se realizan para validar el funcionamiento del sistema bajo carga, investigar la calibración de la posición del brazo al volumen administrado y comprender el volumen real administrado en función de los parámetros fisiológicos y del circuito respiratorio.

5.1.1.1. Propósito

Esta prueba de banco fue diseñada para evaluar la estabilidad mecánica y eléctrica y la durabilidad de un sistema bajo una variedad de condiciones de carga. Estas pruebas de banco son importantes, ya que se debe inspeccionar el sistema mecánico para ver si está desgastado y el software junto con el sistema de control para verificar su desempeño, incluida la sincronización y las funciones correctas de las alarmas. Esto ayuda a determinar el entorno operativo del sistema o el peor escenario de uso en términos de volumen Tidal, frecuencia respiratoria y relación I:E.

Esta prueba también les permitió comprender e identificar los posibles modos de falla y sus implicaciones para el funcionamiento del sistema y la seguridad del paciente. Para evitar sesgos, esta prueba se realizó con instrumentación independiente del sistema de ventilación.

Una vez que completadas las pruebas de banco, el sistema fue conectado a un simulador de respiración de grado hospitalario, utilizado para calibrar y validar el funcionamiento del ventilador del hospital. A continuación, se detalla los equipos, métodos, procesos y resultados que fueron obtenidos por el grupo del MIT y que sirvieron de base para la construcción del prototipo replica documentado aquí.

5.1.1.2. Equipos de prueba

Tabla 5.1: Equipos de prueba para medición

Artículo	Función/Nota
Multimetro	Monitorea la corriente durante el ciclo. Monitorea el termopar
Termopar, Cámara térmica, Pistola térmica	Supervision de la temperatura de los componentes, incluido el motor, la caja de engranajes y el controlador del motor. Cámara Flir IR, Fluke 56X
Cronómetro y contador	Se utiliza para verificar manualmente la sincronización del sistema. Consultar la traza de presión o volumen del transductor de presión y asi el espirómetro o simulador de respiración proporcionará una mayor fidelidad.
Transductor de presión y registrador de datos	Monitorea el perfil de presión en el circuito respiratorio durante todo el ciclo. Ejemplo: Sensor de presión de gas Vernier Go Direct, LabQuest 2, Logger Pro. Nota: estos son solo para uso en banco, no para pruebas clínicas; para pruebas clinicas, se debe utilizar un simulador de respiración calibrado.
Espirómetro	Registra el flujo de aire y el volumen corriente. Ejemplo: espirómetro Vernier.
Cámara digital de video	Se utiliza para registrar secuencias de prueba, incluidos sonidos y observaciones específicas.
Hoja blanca	Se utiliza para recolectar partículas/escombros debajo de un dispositivo mecánico y facilitar la fotografía.
Simulador de respiración	Estos son sistemas de grado hospitalario que se utilizan para validar el funcionamiento del ventilador. Ejemplo: Analizador de flujo de gas/comprobador de ventilador de Fluke Biomedical utilizado junto con un simulador de pulmón de Michigan Instruments o un IngMar 5000, que permite leer todos los parámetros directamente. Nota: Debido a COVID-19, estos productores están atrasados, se recomienda buscar en las instalaciones clínicas locales para obtener disponibilidad inmediata.

5.2. Prueba de forma de onda

Específicamente para detectar problemas de sincronización, es fundamental realizar un barrido que comprenda múltiples RR, I:E y volúmenes corrientes, con un espirómetro o un transductor de presión (e idealmente ambos) conectados y los datos registrados para su examen. Realizaron este barrido de datos, con RR que fue variado en incrementos de 5 de 10 a 40 BPM para relaciones I:E de 1:1, 1:2, 1:3 y 1:4, mientras que el volumen solicitado lo dejaron en 600 mL.

En el caso de nuestro El script de MATLAB utilizado para este análisis logro generar un tren de pulsos ïdeal"basado en I:E y RR de la siguiente manera. Ignoraba el primer pulso, luego sincronizaba el tren con el primer ciclo de los datos medidos y luego generaba los pulsos a partir de ahí, sin más sincronización. Esta forma de ciclos fuera de sincronización pudo ser detectada mediante inspección. En las figuras 5.1 y 5.2 se muestra esto.

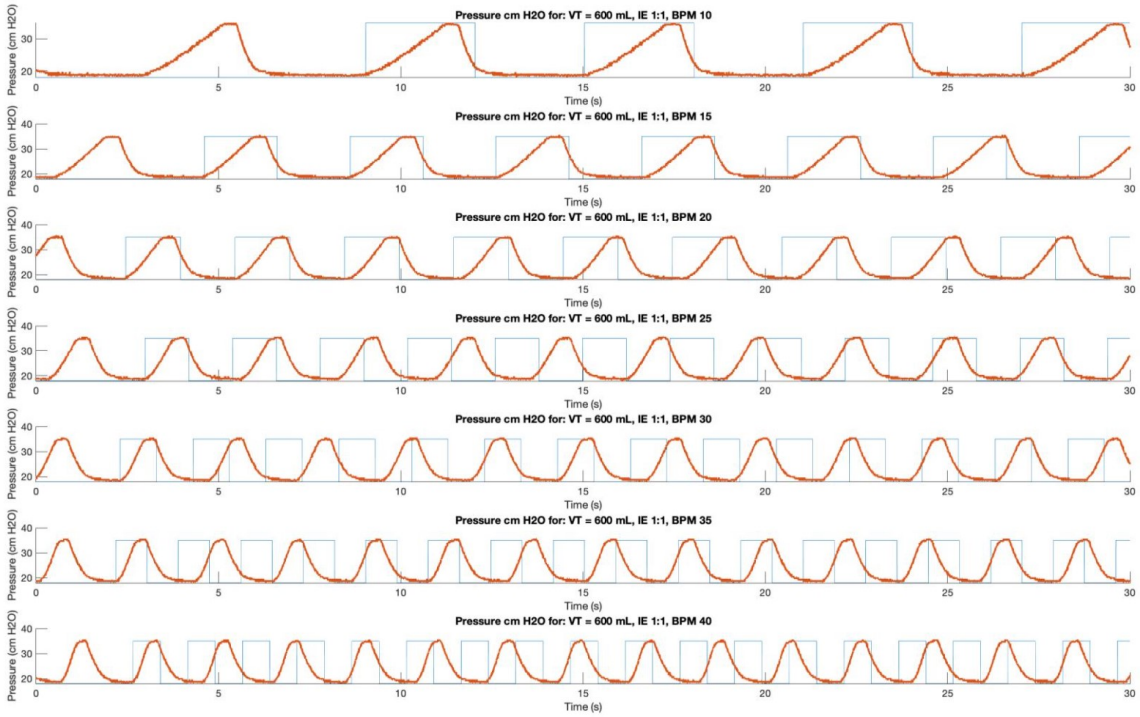


Figura 5.1: Datos de presión del barrido de frecuencia respiratoria a VT = 600 e I:E = 1:1 [6].

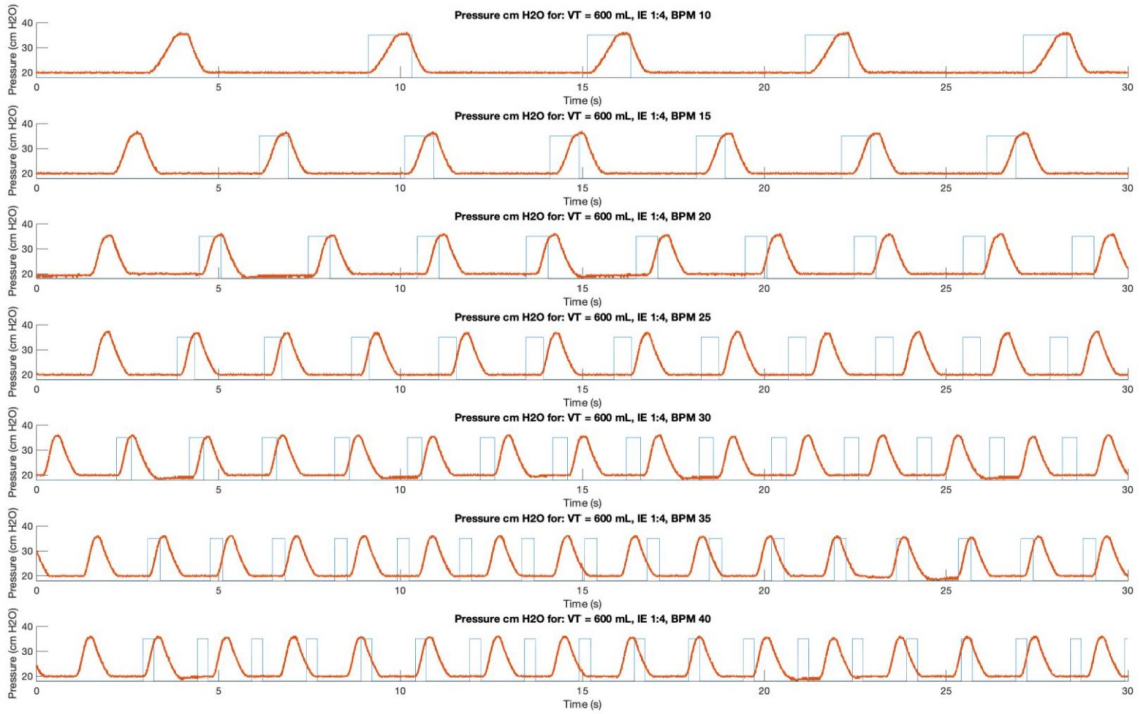


Figura 5.2: Datos de presión del barrido de frecuencia respiratoria a VT = 600 e I:E = 1:4 [6].

La inhalación bajo el movimiento del brazo de control de velocidad está indicada por la pendiente ascendente; la meseta pequeña es la pausa de 0,1 s durante la cual se mide y se muestra la presión Plateau; la pendiente descendente representa la exhalación y se rige por la mecánica de fluidos de la bolsa y el circuito respiratorio; y la permanencia más baja es el tiempo durante el cual se mide la presión PEEP. Según el código que liberaron para realizar las pruebas del circuito neumático, el tiempo de inspiración (I) debe ser la suma del tiempo de la pendiente ascendente más la pausa, mientras que el tiempo de exhalación (E) debe ser la suma de la pendiente descendente y la meseta inferior.

Debido a que la bolsa y el circuito de anestesia no simulan la resistencia de las vías respiratorias, lo que conduce a una contrapresión durante el flujo de aire, las presiones PIP y Plateau no deberían ser sustancialmente diferentes. En general, la PIP debería ser más alta que la Plateu bajo ventilación con control de volumen.

En los gráficos, se puede ver que la meseta superior es relativamente consistente en las relaciones I:E, pero se vuelve un poco más corta en función del aumento de RR. En general, a medida que aumentaban el RR e I:E, el sistema sufría retrasos. Para I:E de 1:3 y superior, la RR aumentaba, el sistema no podía moverse a la velocidad de inspiración deseada. Esto lo rastrearon hasta que determinaron que no se debía tener en cuenta el tiempo de aceleración del motor y eso provoco mejoras en el diseño del controlador del motor.

5.3. Configuración del dispositivo

Durante estas pruebas, colocaron una bolsa de reanimación manual en el ventilador de emergencia y lo hicieron funcionar bajo un conjunto específico de condiciones de prueba, incluido el volumen tidal (VT), la relación I:E y la frecuencia respiratoria (RR). El enfoque se basó en evaluar el desempeño de la bolsa y sus válvulas, así como monitorear los componentes mecánicos y eléctricos. La configuración de prueba se muestra en la Figura 5.3.

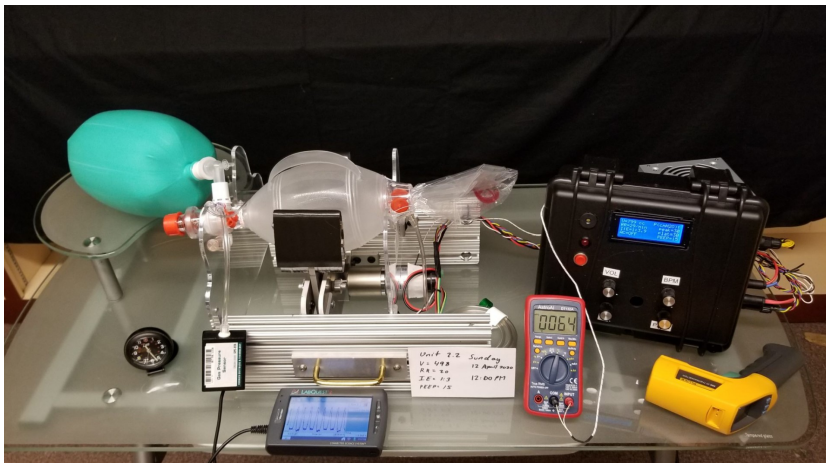


Figura 5.3: Unidad de prueba junto con equipos de medición de corriente, tiempo, presión y térmica. El espirómetro no se muestra, pero se puede conectar en línea con el circuito respiratorio [7].

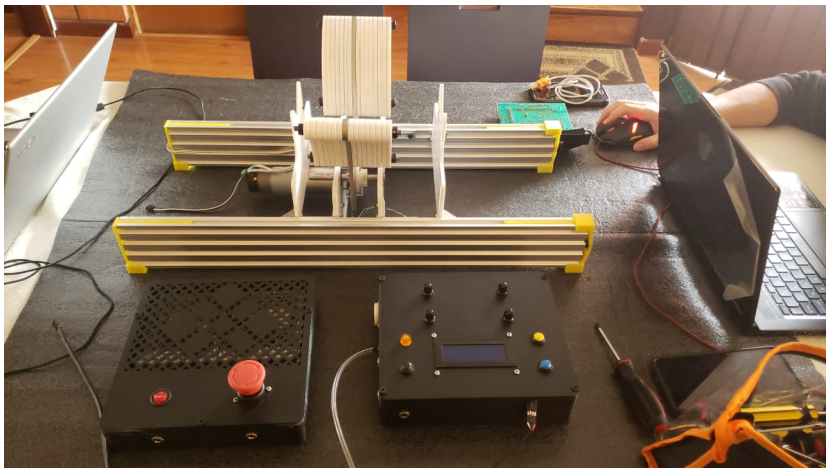


Figura 5.4: Unidad construida como replica del ventilador mecánico.

Es importante destacar de este circuito neumático que la bolsa de anestesia elástica solo modela parcialmente la distensibilidad pulmonar humana. Esto se expresa como $cm\ H_2O$ O por ml de volumen administrado, es decir, un globo se vuelve más difícil de inflar a medida que se administra más volumen. Los rangos de conformidad típicos para COVID-19 son: 15 - 50 mL / $cm\ H_2O$. y los rangos de resistencia típicos para COVID-19 son: 5 - 20 $cm\ H_2O/\left(L/s\right)$

Para esta prueba, la bolsa de anestesia no fue cargada, pero la premisa que plantearon es que se podría colocar pesos sobre ella para aumentar la carga para una prueba funcional. Por ejemplo, con un área de superficie estimada de la bolsa de 100 cm^2 (cuando se aplana), colocar una carga plana encima de 5 kg simulará aproximadamente 40 $cm\ H_2O$. Para todo esto tuvieron en cuenta que esta es una presión constante, no un cumplimiento real, y la resistencia no es ajustable, como es posible con un simulador de respiración, pero aumentará la tensión en los componentes mecánicos y eléctricos para fines de prueba.

5.4. Comparación del modelo construido con otros modelos de ventilación mecánica

En el mercado existen un sinnúmero de ventiladores para respiración asistida de una gran variedad de marcas y con distintos propósitos, como se ha mencionado en secciones anteriores, el prototipo construido no entra a competir con ventiladores de alta gama, es más bien, un dispositivo cuyo objetivo es solventar la escasez de dichos ventiladores de altas prestaciones en ambientes saturados por la emergencia sanitaria que vivimos hoy en día, siendo un ventilador de tránsito a ventiladores más sofisticados o usado en el peor de los casos, es decir cuando no se disponga de un ventilador profesional.

Como comparativa, se ha tomado el ventilador mecánico para respiración asistida ResMed Astral modelo 100 - 150, este ventilador posee varios modos de ventilación, y controles mucho más precisos y sofisticados que el modelo construido, por lo que el estudio comparativo se ha realizado para los modos de ventilación de volumen y control asistido solamente, así también se ha realizado una comparación de las distintas alarmas disponibles en ambos ventiladores.

5.4.1. Principio de funcionamiento

Como ya se ha mencionado en secciones anteriores, el dispositivo construido basa su funcionamiento en el movimiento de dos brazos mecánicos que aplastan una bolsa Ambu, la cual es la responsable de brindar la respiración asistida al paciente, que tanto y que tan veloz sea aplastada la bolsa Ambu depende del control implementado y de la retroalimentación que se tiene con el sensor de presión, además ciertas medidas importantes en la respiración asistida dependen de los componentes del circuito neumático del paciente (v. gr. válvula PEEP).

El funcionamiento del dispositivo para respiración mecánica asistida Astral ResMed 100 - 150 [8], basa su funcionamiento en dos turbinas, la primera usada para brindar la respiración asistida y la segunda para mantener una presión PEEP deseada, adicionalmente posee dos ramas de circulación de aire y una gran variedad de sensores, como sensor de presión para las vías respiratorias del paciente, sensor de flujo respiratorio, sensor de flujo de salida, sensor de presión de salida, sensor de FiO_2, sensor de temperatura ambiente, además de válvulas de control de flujo de aire, de seguridad, para control I:E, de retención, etc. Un diagrama del circuito neumático del ventilador ResMed Astral 100 - 150 puede observarse en la Figura 5.5. Lo que se acaba de mencionar es correspondiente a la bolsa Ambu y el filtro HEPA del dispositivo construido.

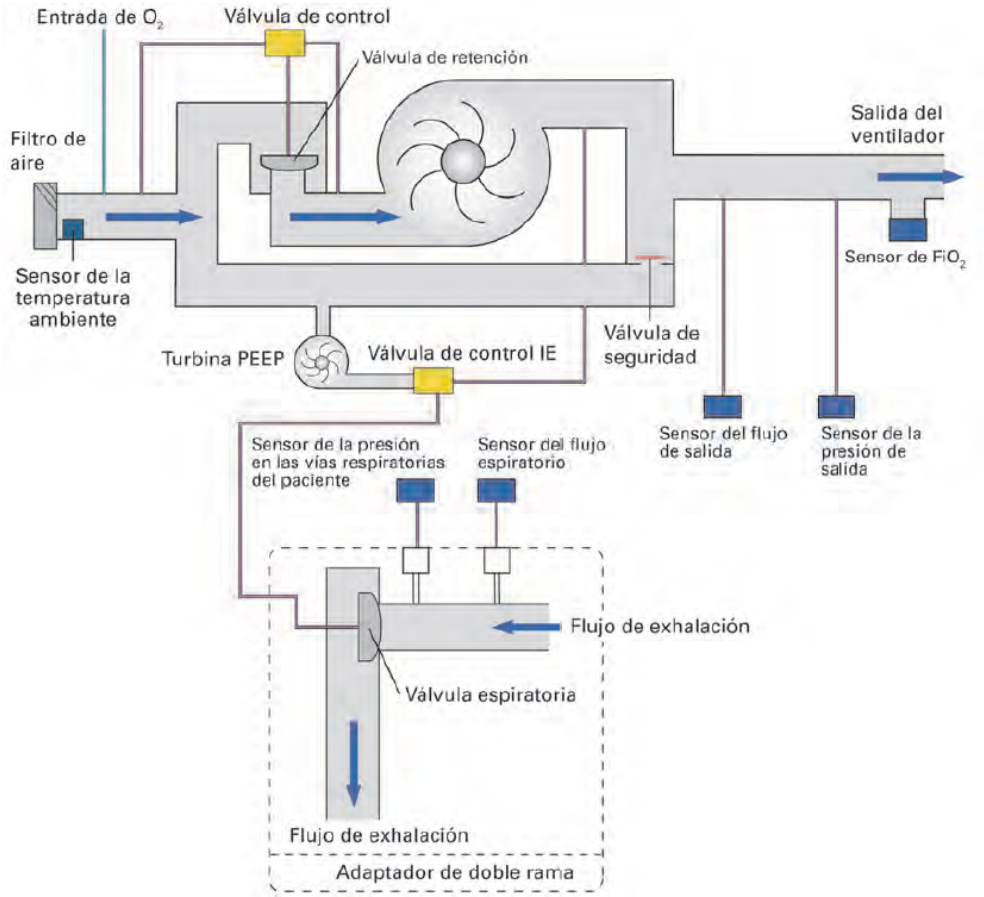


Figura 5.5: Flujo de aire en el ventilador mecánico de respiración asistida Astral 100 - 150 [8]

El ventilador mecánico para respiración asistida ResMed Astral 100 - 150 es, en todas maneras, más sofisticado que el dispositivo construido, tiene un principio de funcionamiento distinto, además de poseer muchos más modos de ventilación, sensores y válvulas, lo que hace que el mismo sea mucho más confiable y seguro que el dispositivo construido. Sin embargo, también es mucho más costoso, en el mercado local, puede conseguirse aproximadamente al rededor de $ 16,500.00 dólares.

En cuanto al dispositivo de ventilación asistida OxyMag [9], posee así mismo gran variedad de tipos de ventilación, con control de variables que el prototipo construido no toma en cuenta, además de hacer uso de un proceso de ventilación centrado en el funcionamiento de un Venturi, además de poseer una gran variedad de válvulas, como válvula reguladora de presión, válvulas proporcionales, válvulas de sobrepresión, unidireccional, además de sensores de presión, flujo inspiratorio, de flujo proximal, y otros dispositivos que se salen del alcance del presente proyecto, en la Figura 5.6, puede observarse el esquemático neumático del ventilador mencionado.

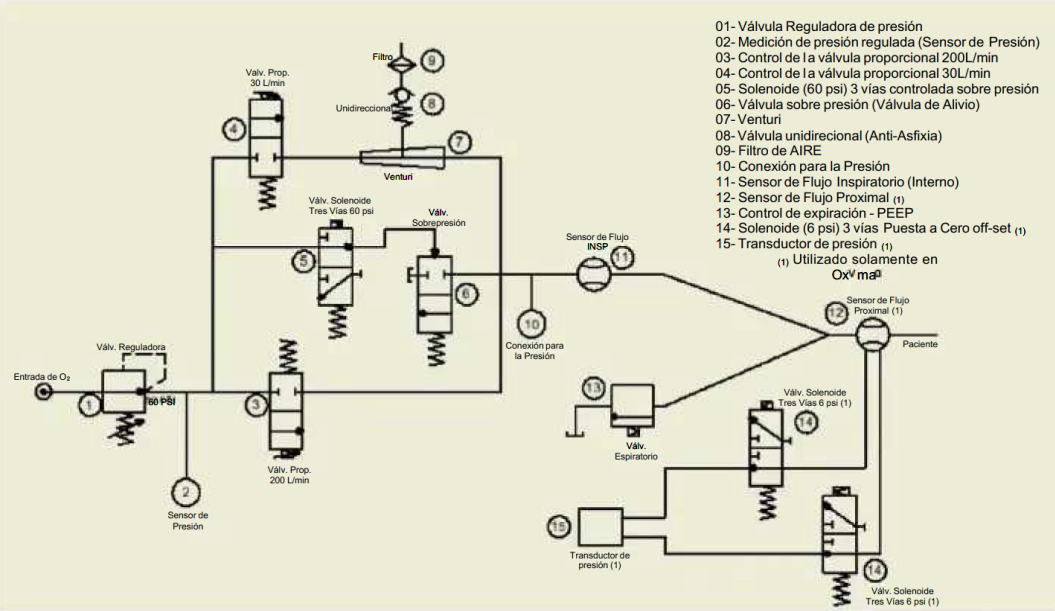


Figura 5.6: Flujo de aire en el ventilador de respiración asistida OxyMag [9]

El principio de funcionamiento del ventilador OxyMag varía en gran manera del de los primeros dos, mientras que los primeros usan técnicas mecánicas para brindar respiración, el ventilador OxyMag usa un efecto tipo venturi, completamente neumático, en cuanto a la comparativa de funcionamiento entre el ventilador de turbina y el venturi, así como una descripción del funcionamiento de ambos ventiladores, se salen del alcance del proyecto. En el medio se ha encontrado el ventilador OxyMag al rededor de los $ 14,000.00 dólares.

5.4.2. Límites y rangos de funcionamiento

Antes de comenzar la comparativa hay que realizar una distinción entre los límites de funcionamiento y los rangos de funcionamiento.

Los límites de funcionamiento son constantes que se monitorizan periódicamente y que no deberán ser superadas nunca durante el funcionamiento de los dispositivos, de ser superados dichos límites se puede ocasionar lesiones graves al paciente conectado. Los rangos de funcionamiento son variables configurables o no configurables con los que los dispositivos realizan las diferentes tareas de control y que sirven de base para el funcionamiento del sistema.

Se ha tomado los límites y rangos de funcionamiento del prototipo construido y se han comparado con los límites y rangos de funcionamiento del ventilador mecánico ResMed Astral 100 - 150 y del ventilador OxyMag, como se pueden observar en las Tablas 5.2, 5.3, 5.4, 5.5 respectivamente.

Tabla 5.2: Límites de funcionamiento entre el prototipo construido y el ventilador mecánico ResMed Astral 100 - 150

Prototipo construido

	Adulto
Peso del paciente	>40 Kg
Volumen por aplastamiento	$\geq$ 600 ml
Volumen de la bolsa	1500 ml $\pm$ 200 ml
Dimensiones	212 mm x 131 mm
Limite de presión de la válvula	40 $cm\ H_2O$ - 60 $cm\ H_2O$
Resistencia inspiratoria	$\leq$ 5 $cm\ H_2O$ (50 L/min)
Resistencia espiratoria	$\geq$ 5 $cm\ H_2O$ (50 L/min)

Astal 100 - 150

	Adulto
Peso del paciente	>40 Kg
Volumen	100 - 2500 ml
Dimensiones	285 mm x 215 mm
Limite de presion	61,184 $cm\ H_2O$
Resistencia inspiratoria	5,812 $cm\ H_2O$ (30 L/min)
Resistencia espiratoria	4,282 $cm\ H_2O$ (30 L/min)

Tabla 5.3: Límites de funcionamiento entre el prototipo construido y el ventilador OxyMag

Prototipo construido

	Adulto
Peso del paciente	>40 Kg
Volumen por aplastamiento	$\geq$ 600 ml
Volumen de la bolsa	1500 ml $\pm$ 200 ml
Dimensiones	212 mm x 131 mm
Limite de presión de la valvula	40 $cm\ H_2O$ - 60 $cm\ H_2O$
Resistencia inspiratoria	$\leq$ 5 $cm\ H_2O$ (50 L/min)
Resistencia espiratoria	$\geq$ 5 $cm\ H_2O$ (50 L/min)

OxyMag

	Adulto
Peso del paciente	>40 Kg
Volumen	100 - 2500 ml
Dimensiones	231 mm x 254 mm
Limite de presion	60 $cm\ H_2O$
Resistencia inspiratoria	0 - 200 $cm\ H_2O$
Resistencia espiratoria	

Con respecto a la comparativa de límites de funcionamiento mostrados en la Tabla 5.2, pueden observarse límites muy similares, ya que estos vienen dados por estándares de seguridad para los pacientes, el dispositivo construido tiene que ser usado por pacientes adultos, por tanto, la información obtenida es concorde a ello.

Ocurre de la misma forma con los límites de funcionamiento mostrados en la Tabla 5.3, al igual que con el ventilador Astral, existen limites similares, como ya se dijo, esto se debe a que generalmente estos límites están estandarizados para la seguridad de los pacientes, en este caso pacientes adultos.

Cabe destacar que, los límites de funcionamiento del prototipo construido vienen dados tanto por la bolsa Ambu como por los diferentes componentes del apartado neumático (filtros, válvulas, tubos, etc.), ya que ellos poseen seguridades intrínsecas ya configuradas, en el apartado de control programado, también se establecieron límites de funcionamiento, sin embargo estos se encuentran dentro de los límites de seguridad establecidos por los componentes usados así que no se han tomado en cuenta en esta comparativa y se han usado más bien para la comparación de rangos de funcionamiento.

Tabla 5.4: Rangos de funcionamiento entre el prototipo construido y el ventilador mecánico ResMed Astral 100 - 150

Prototipo construido

	Adulto
Peso del paciente	>40 Kg
Presión Máxima	40 $cm\ H_2O$
Presión Plateau mínima	5 $cm\ H_2O$
Presión de resistencia máxima	10 $cm\ H_2O$
Presión tidal mínima	5 $cm\ H_2O$
Límite de error de volumen entregado	50 $cm\ H_2O$
Presión del AC mínima	2 $cm\ H_2O$
Presión del AC máxima	7 $cm\ H_2O$
Respiraciones por minuto mínimos	6
Respiraciones por minuto máximos	35
Tasa I:E mínima	1
Tasa I:E máxima	4
Volumen entregado mínimo	100 ml
Volumen entregado máximo	800 ml

Astral 100 -150

	Adulto
Peso del paciente	>40 Kg
Presión Máxima	0 - 99 $cm\ H_2O$
Presión Plateau mínima	N/A
Presión de resistencia máxima	N/A
Presión tidal mínima	N/A
Límite de error de volumen entregado	N/A
Presión del AC mínima	N/A
Presión del AC máxima	N/A
Respiraciones por minuto mínimos	0 - 99
Respiraciones por minuto máximos	0 - 99
Tasa I:E mínima	1:9,9 - 9,9:1
Tasa I:E máxima	1:9,9 - 9,9:1
Volumen entregado mínimo	0 - 3000 ml
Volumen entregado máximo	0 - 3000 ml

Tabla 5.5: Rangos de funcionamiento entre el prototipo construido y el ventilador mecánico OxyMag

Prototipo construido

	Adulto
Peso del paciente	>40 Kg
Presión Máxima	40 $cm\ H_2O$
Presión Plateau mínima	5 $cm\ H_2O$
Presión de resistencia máxima	10 $cm\ H_2O$
Presión tidal mínima	5 $cm\ H_2O$
Límite de error de volumen entregado	50 $cm\ H_2O$
Presión del AC mínima	2 $cm\ H_2O$
Presión del AC máxima	7 $cm\ H_2O$
Respiraciones por minuto mínimos	6
Respiraciones por minuto máximos	35
Tasa I:E mínima	1
Tasa I:E máxima	4
Volumen entregado mínimo	100 ml
Volumen entregado máximo	800 ml

OxyMag

	Adulto
Peso del paciente	>40 Kg
Presión Máxima	-20 a 90 $cm\ H_2O$
Presión Plateau mínima	0 a 90 $cm\ H_2O$
Presión de resistencia máxima	N/A
Presión tidal mínima	N/A
Límite de error de volumen entregado	N/A
Presión del AC mínima	N/A
Presión del AC máxima	N/A
Respiraciones por minuto mínimos	0 - 200
Respiraciones por minuto máximos	
Tasa I:E mínima	0:100,0 - 100,0:1
Tasa I:E máxima	
Volumen entregado mínimo	0 - 3000 ml
Volumen entregado máximo	

En cuanto a los rangos de funcionamiento mostrados en la Tabla 5.4, puede observarse una variabilidad entre los dos dispositivos, esto se debe, generalmente a que el ventilador mecánico ResMed Astral 100 - 150, es un dispositivo más sofisticado, es decir, se pueden tener control sobre más variables, además de tener un rango de funcionamiento superior en cuestiones de volumen y presión. Adicionalmente el principio de funcionamiento de ambos ventiladores es distinto, el prototipo construido usa una bolsa Ambu para dar respiración asistida, mientras que el equipo Astral centra su funcionamiento en dos turbinas que son las responsables de dar la respiración asistida al paciente. Finalmente, existen variables que no están mostradas en la Tabla 5.4, esto se debe a que son variables importantes en el

control del dispositivo, dichas variables no se presentan en el manual de usuario del ventilador ResMed Astral 100 - 150 por motivos de confidencialidad.

Ocurre de la misma forma con la comparación entre el prototipo construido y el ventilador OxyMag, mostrado en 5.5, de igual manera, el dispositivo posee prestaciones superiores al del dispositivo construido en este proyecto, y ciertas variables no se encuentran descritas en el manual de usuario debido a temas de confidencialidad, sin embargo, en las que si se encuentran disponibles, es posible observar una similitud en cuanto a rango de funcionamiento.

5.4.3. Comparativa de alarmas

Como se ha observado en la Tabla 5.1, el dispositivo construido para este proyecto consta de 6 alarmas para cada caso específico, y cada alarma tiene su respuesta, así mismo, los ventiladores RedMed Astral 100 - 150 y OxyMag, poseen listados de alarmas variados enfocados en tres niveles de prioridad, como se puede observar en las Tablas 5.6 y 5.7.

Tabla 5.6: Comparativa del listado de alarmas del ventilador mecánico ResMed Astral 100 - 150 y el prototipo construido

Astral 100 - 150			Prototipo construido
Alta prioridad	Prioridad media	Alarmas de baja prioridad	Alarmas
Fallo total de alimentación	Presión alta	Alimentación desconectada	Presión PIP excedida
Presion baja	PEEP bajo	Uso de batería interna	Presión baja
Obstrucción / Presión alta	PEEP alto	Falla de batería 1	Presión de resistencia alta
Apnea	Frecuencia de pulso bajo	Falla de batería 2	Fallo de sobrecorriente
VMe bajo	Frecuencia de pulso alto	Falla de alimentación / Sin cargar	Volumen Tidal no entregado
VMi bajo	Sobrecalentamiento del dispositivo		Presión Tidal no detectada
VMi alto	Línea de presión desconectada		
VMe alto	Ultima autoevaluación fallida		
Vce bajo	Sensor de flujo no calibrado		
Vce alto	Sin monitorización de SpO2		
Vci bajo	Sin monitorización de FiO2		
Vci alto	Batería interna baja		
Frecuencia respiratoria baja			
Frecuencia respiratoria alta			
SpO2 bajo			
SpO2 alto			
FiO2 bajo			
FiO2 alto			
Mascarilla sin ventilación (ventilación bloqueada)			
Ventilación no iniciada, adaptador incorrecto			
Batería interna críticamente baja			
Falla del circuito			
Reinicio inesperado			
Batería interna inoperable			

El ventilador Astral posee muchas más alarmas especializadas, esto se debe a que posee más modos de funcionamiento, más sensores, puede funcionar con batería, permite el control de la mezcla de

gases que pasan al paciente, permite medir los signos vitales del paciente así como diagnosticar ciertas patologías relacionadas con el ámbito respiratorio. El manual de alarmas del ventilador Astral indica el motivo aparente de la activación de cada alarma, la reacción del dispositivo y la reacción que debe tener el operador hacia cada una de ellas.

Tabla 5.7: Comparativa del listado de alarmas del ventilador OxyMag y el prototipo construido

Alta prioridad	Prioridad media	Alarmas de baja prioridad	Prototipo Construido
Batería Baja	Volumen mínimo alto	Sin red eléctrica	Presión PIP excedida
Apnea	Volumen mínimo bajo	Sensor SpO2	Presión baja
Presión O2 baja	Frecuencia alta	Verificar SpO2	Presión de resistencia alta
Obstrucción	Frecuencia baja	Verificar el cable	Fallo de sobrecorriente
Desconexión	PEEP alto	Perfusión baja	Volumen tidal no entregado
Presión Máxima Alta	PEEP bajo	Buscando pulso	Presión tidal no detectada
Presión Máxima Baja	Adaptador IRMA	Activando SpO2	
Volumen Alto	Reiniciar IRMA	SpO2 demo	
Volumen Bajo	Cambiar IRMA		
FiO2 Alto	CO2 fuera de escala		
FiO2 Bajo	Error de lectura IRMA		
EtCO2 alto	Calibrar IRMA		
EtCO2 bajo			
Co2i			
FC alta			
FC baja			
SpO2 bajo			

Similar al ventilador Astral, el ventilador Oximag posee su listado de alarmas teniendo en cuenta prioridades, así también, posee una gran variedad de alarmas en concordancia al número de parámetros configurables y el control de los varios sensores disponibles, igual que con el ventilador Astral, permite realizar la medición de signos vitales y de estimar el estado de los pacientes en temáticas de patología respiratoria.

En el caso del ventilador construido para este proyecto se han programado las alarmas más imperativas con respecto a la supervivencia del paciente, teniendo en cuenta el modo de respiración brindado por el mismo y el sensor de presión disponible. El dispositivo reacciona a cada alarma de las formas ya descritas, no existe un nivel de prioridad, sin embargo, de darse la activación de alguna alarma que pudiese ser extremadamente perjudicial para el paciente el ventilador actuará en consecuencia (v. gr. se detendrá).

5.5. Pruebas de volumen con el ventilador construido

Finalmente, las pruebas que se muestran a continuación, son pruebas realizadas con distintas configuraciones volumen tidal. Estas son las únicas pruebas que pueden ser captadas en imágenes para que la diferencia entre una configuración y otra pueda ser apreciada. Para poder apreciar el funcionamiento completo y la variación de los distintos parámetros se puede observar el video de la referencia [29].

En la figura 5.7 se puede observar la configuración para un volumen de 200 ml. Igualmente en las figuras 5.8a y 5.8b se muestra imágenes de la apertura y cierre de los dedos al aplastar la bolsa ambu con la misma configuración.

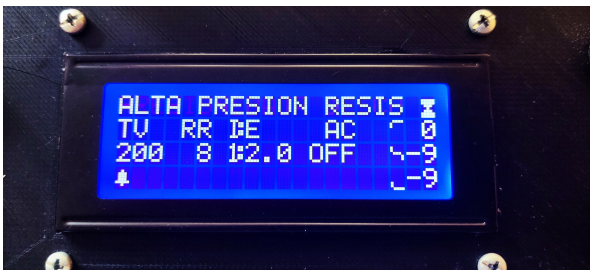


Figura 5.7: Configuración para un volumen tidal de 200 ml.

(a)

(b)

Figura 5.8: Apertura y cierre de los dedos para un volumen tidal de 200 ml.

Igualmente, en la figura 5.9 se puede observar la configuración para un volumen de 300 ml. Igualmente en las figuras 5.10a y 5.10b se muestra imágenes de la apertura y cierre de los dedos al aplastar la bolsa ambu con la misma configuración.

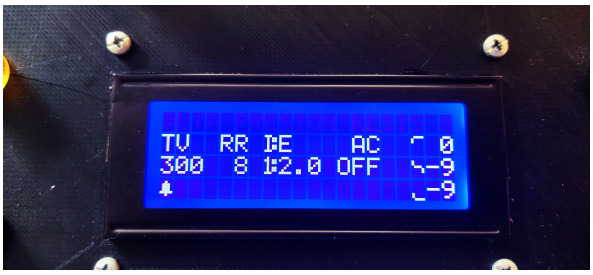
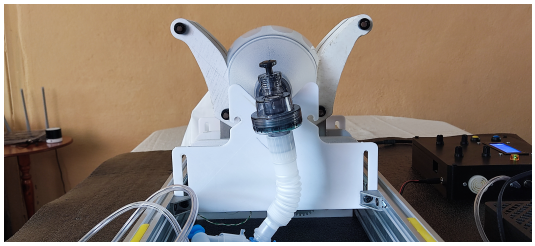


Figura 5.9: Configuración para un volumen tidal de 300 ml.



(a)

(b)

Figura 5.10: Apertura y cierre de los dedos para un volumen tidal de 300 ml.

Asi mismo, en la figura 5.11 se puede observar la configuración para un volumen de 500 ml. Igualmente en las figuras 5.12a y 5.12b se muestra imágenes de la apertura y cierre de los dedos al aplastar la bolsa ambu con la misma configuración.

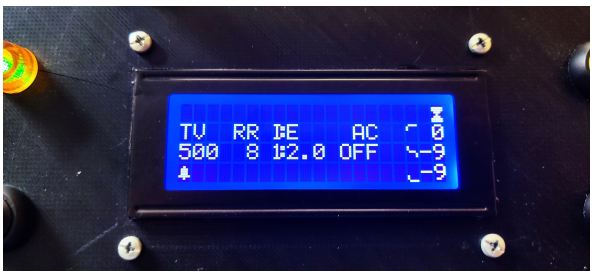
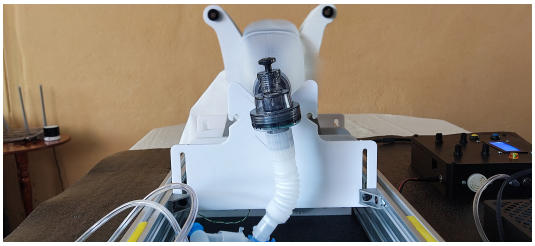


Figura 5.11: Configuración para un volumen tidal de 500 ml.

(a)



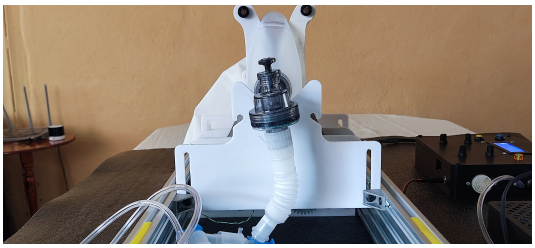
(b)

Figura 5.12: Apertura y cierre de los dedos para un volumen tidal de 200 ml.

Finalmente, en la figura 5.13 se puede observar la configuración para un volumen de 700 ml. Igualmente en las figuras 5.14a y 5.14b se muestra imágenes de la apertura y cierre de los dedos al aplastar la bolsa ambu con la misma configuración.

Figura 5.13: Configuración para un volumen tidal de 700 ml.

(a)



(b)

Figura 5.14: Apertura y cierre de los dedos para un volumen tidal de 700 ml.

5.5.1. Conclusiones y recomendaciones

En cuanto a las pruebas realizadas por el MIT se puede decir que: el propósito de las pruebas de banco es identificar fácil y rápidamente problemas potenciales con el sistema de control. También permiten que la envolvente del rendimiento máximo del sistema se determine mediante inspección. Esto debe formalizarse en un procedimiento de calificación del sistema para los dispositivos construidos. Se pueden realizar pruebas similares para investigar el volumen por minuto utilizando el espirómetro conectado al resucitador manual. El uso de un simulador de respiración permite aplicar cumplimientos y resistencias más realistas y registrar el rendimiento del sistema.

Así mismo, en cuanto a las mediciones tomadas en el prototipo réplica se puede decir que: Según lo analizado en temas de comparación entre el prototipo desarrollado en este proyecto y los dos ventiladores de grandes prestaciones, ResMed Astral 100 - 150 y Magnamed OxyMag, se concluye lo siguiente:

- El ventilador construido tiene su principio de funcionamiento centrado en la portabilidad y disponibilidad de elementos de hospital comunes.
- El ventilador construido no viene a competir contra ventiladores de gama alta, más bien es un ventilador de paso o en casos extremos de emergencia cuando se tengan problemas de saturación.
- Existen gran variedad de modos de ventilación y cada modo puede ser brindado mediante diferentes principios de funcionamiento del ventilador (v. gr. bolsa Ambu, turbina, venturi)
- Hay similitud en las variables de límite de seguridad de los tres ventiladores, esto se debe a que son límites estandarizados para el bienestar general de los pacientes.
- Mientras que en el ventilador construido, los parámetros de seguridad dependen no solamente del control programado, sino de los elementos usados en el circuito neumático (válvulas, filtros, tubos,

etc.), es decir, sus componentes externos, en los ventiladores más sofisticados sus características constructivas y de control abordan directamente dichas variables dentro del ventilador, además de poder brindar seguridad por sus componentes externos conectados.

- Los parámetros de funcionamiento de los ventiladores sofisticados tienen un rango de variabilidad superior al del prototipo construido, esto se debe a la gran cantidad de modos de ventilación, que necesitan control sobre parámetros que para el ventilador construido son valores fijos, además de elementos adicionales que no se disponen en el prototipo construido (sensores, válvulas, etc.).
- Las alarmas disponibles en los ventiladores de gama alta tienen un sistema de prioridades, lo que hace posible que el operario decida actuar a su debido tiempo, en el caso del prototipo construido, se tienen las alarmas necesarias para el bienestar del paciente, y a su vez, el dispositivo está dirigido para pacientes víctimas de un caso de emergencia extrema, por ello es indispensable que al encontrarse en funcionamiento se tenga un profesional de la salud monitorizando al paciente en todo momento.
- Existe una gran diferencia entre el precio del ventilador construido y el precio de los ventiladores de gama alta en el mercado ecuatoriano, los ventiladores de gama alta rondan aproximadamente los $15,000.00 dólares americanos, mientras que el precio constructivo del prototipo desarrollado así como de los componentes usados puede observarse en el apéndice E.

Adicionalmente se tienen las siguientes recomendaciones:

- Si bien el principio de funcionamiento del ventilador usa una bolsa Ambu, que es un elemento hospitalario común y desechable, es recomendable basar el funcionamiento de los ventiladores en instrumentación más segura (turbinas, venturi, etc.).
- Si bien los elementos externos al ventilador construido, tales como válvulas y filtros, poseen ya seguridades constructivas para el paciente, es recomendable buscar la manera de controlar dichas variables desde el control del ventilador, ya sea dentro del ventilador, constructivamente o dentro del código programado para el control (el prototipo construido si tiene programado en el control, límites para las variables importantes).
- Es recomendable intentar controlar más variables de salida del ventilador, como flujo o presiones pulmonares importantes, para esto deberían implementarse los sensores que sean necesarios.
- Aumentar el listado de alarmas es recomendable, esto puede hacerse cuando el dispositivo sea probado en ambientes de ventilación asistida.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

Se ha construido un prototipo de ventilador mecánico que basa su principio de funcionamiento en el aplastamiento de una bolsa Ambu, el ventilador tiene como objetivo entrar en las zonas de tratamiento UCI que se encuentran saturadas por la emergencia sanitaria se encuentran atravesando los hospitales.

La intención del ventilador construido en este trabajo no es competir contra los ventiladores de alta gama, sino más bien, servir como un ventilador de transición a pacientes que salen o entran a la UCI y, en el peor de los casos, ser usado como un ventilador de emergencia, cabe destacar que es necesario que un profesional de la salud se encuentre monitorizando la evolución del paciente en todo momento.

Se ha construido el ventilador siguiendo la bibliografía pertinente así como recomendaciones de entidades como la ISO y la AAMI, cabe destacar que el estudio realizado en este proyecto ha tenido que ver más con el apartado tecnológico que con el apartado médico, ya que la profundización de temas puramente médicos se sale de nuestro campo de experiencia.

La construcción del prototipo desarrollado ha tenido como base la patente liberada por el MIT, aplicando dicho conocimiento a la realidad del Ecuador, con procedimientos, componentes y herramientas disponibles en el medio, intentando abaratar los costos lo máximo posible, aún así, existieron componentes, tales como el motor, su respectivo driver y el sensor de presión que no pudieron ser conseguidos de forma local, sino que tuvieron que ser importados.

Como se ha podido observar en los resultados (Capítulo 5), los desarrolladores de la MIT han realizado distintos estudios basados en modelamiento teórico y mediciones en campo, el procedimiento seguido por ellos, así como sus resultados han sido plasmados en dicho capítulo. Mediciones con el prototipo desarrollado han sido realizadas bajo condiciones cuestionables, ya que no disponemos de un ambiente de simulación de circuito de respiración, ni de la posibilidad de realizar pruebas con animales, ni

con personas, además no poseemos las herramientas de medición requeridas para cada prueba específica.

Se ha realizado una comparativa entre dos modelos de ventiladores disponibles en el mercado ecuatoriano con el prototipo de ventilador desarrollado, los puntos más importantes de dicha comparativa vienen a centrarse en los parámetros de funcionamiento límite y sus rangos de funcionamiento seguro. Tanto para el ventilador construido, como para el ventilador Astral y el ventilador OxyMag, dichos parámetros vienen ya estandarizados por recomendaciones que enfocan sus esfuerzos en la seguridad del paciente, por tanto, las variables de control programadas en el prototipo construido se encuentran dentro de lo recomendable por las distintas entidades de estandarización, las variables comparadas, por tanto tienen una similitud bastante próxima (Tablas 5.2, 5.3, 5.4 y 5.5), dando a entender que sus límites de funcionamiento y sus rangos de funcionamiento son bastante próximos.

6.2. Recomendaciones

Conforme se investigó, construyó y probó el prototipo desarrollado se han encontrado varias recomendaciones, éstas se muestran a continuación:

- Es importante centrar el desarrollo en los parámetros recomendados por las entidades de control, ya que ellas estandarizan los rangos de funcionamiento seguro de los ventiladores.
- Es importante tener en cuenta que distintas patologías pulmonares requieren de distintos modos de ventilación, por tanto antes de usar el prototipo desarrollado es importante que el médico esté seguro de que el modo de ventilación implementado se encuentra dentro de lo recomendable para el tratamiento del paciente.
- Se recomienda que los componentes escogidos para la construcción del prototipo soporten las diferentes pruebas de estrés y de funcionamiento continuo, ya que este tipo de dispositivos están destinados a funcionar durante largos períodos de tiempo.
- Se recomienda que el paciente sea sedado e inmovilizado al momento de utilizar el dispositivo, ya que se debe usar un aparato de traqueotomía para su correcto funcionamiento.
- Es recomendable usar componentes externos al ventilador, como la válvula PEEP o filtros HEPA, que deben conectarse por seguridad del paciente.
- La mezcla de gases que se dan al paciente se realiza debe ser configurada por el médico en las conexiones de la bolsa Ambu.
- Si se fuese a usar materiales metálicos en la construcción del dispositivo, se tiene que tener en cuenta que estos deben ser materiales inoxidables y no magnéticos.
- El circuito neumático usado deberá reducir a lo mínimo posible el espacio muerto.
- El sensor de presión usado es bastante frágil, se recomienda manipularlo con cuidado.
- Se recomienda consultar la Tabla 5.1, para conocer las distintas alarmas programadas, por que activan y que hacer en caso de que se active alguna en particular.
-

6.3. Trabajos futuros

Existen varias mejoras que pudiesen aplicarse al dispositivo construido, así como diferentes pruebas y mediciones que pueden ser realizadas, a continuación son descritas algunas de ellas:

- Es importante realizar mediciones en los ambientes de medición requeridos, ya sea mediante herramientas que simulen la respiración humana o mediante el uso de animales o personas.
- Conforme la máquina sea usada se pueden ir aumentando más alarmas dependiendo de estudios médicos de campo.
- Se pudiese aumentar el número de modos de ventilación, esto requiere que se aumenten sensores y se cambie el control de acuerdo a los paradigmas de ventilación pertinentes, cabe destacar que ciertos modos de ventilación requieren principios de funcionamiento distintos al del construido, por lo que un ventilador que tiene como base una bolsa Ambu pudiera no ser recomendable.
- Se pudiera agregar el control de la mezcla de gases que van al paciente haciendo uso de los sensores respectivos.
- Pudiese agregarse un control del pulso del paciente, así como detectarse ciertas patologías respiratorias que pueden darse por la ventilación asistida.
- Pudiese desarrollarse un sistema capaz de modificar la presión PEEP de forma digital, en vez de utilizar una válvula externa.
- Se pudiera realizar un estudio del impacto que tuviera la implementación en masa de este tipo de dispositivos en ambientes saturados por la presente emergencia sanitaria, teniendo en cuenta el bajo costo de producción en comparación con ventiladores de más alta gama.

Anexo A

Código del programa en Arduino

A.1. Código del programa

A continuación se comparte todo el código implementado en el controlador Arduino [14].

```
#include "LiquidCrystal.h"
#include "src/thirdparty/RoboClaw/RoboClaw.h"
#include "cpp_utils.h"
#include "Alarms.h"
#include "Buttons.h"
#include "Constants.h"
#include "Display.h"
#include "Input.h"
#include "Logging.h"
#include "Pressure.h"


using namespace input;
using namespace utils;

unsigned long cycleCount = 0;
float tCycleTimer;
float tIn;
float tHoldIn;
float tEx;
float tPeriod;
float tPeriodActual;
float tLoopTimer;
float tLoopBuffer;
```

```
States state;
bool enteringState;
float tStateTimer;

RoboClaw roboclaw(&Serial3, 10000);
int motorCurrent, motorPosition = 0;

LiquidCrystal lcd(LCD_RS_PIN, LCD_EN_PIN, LCD_D4_PIN, dLCD_D5_PIN, LCD_D6_PIN,
LCD_D7_PIN);
display::Display displ(&lcd, AC_MIN);

alarms::AlarmManager alarm(BEEPER_PIN, SNOOZE_PIN, LED_ALARM_PIN,
&displ, &cycleCount);

Pressure pressureReader(PRESS_SENSE_PIN);

buttons::PressHoldButton offButton(OFF_PIN, 2000);
buttons::DebouncedButton confirmButton(CONFIRM_PIN);

logging::Logger logger(true/*Serial*/, false/*SD*/, false/*labels*/,
",\t"/*delim*/);

struct Knobs {
  int volume();
  int bpm();
  float ie();
  float ac();
  SafeKnob<int> volume_ = SafeKnob<int>(&displ, display::VOLUME, CONFIRM_PIN, &alarm,
VOL_RES);
  SafeKnob<int> bpm_ = SafeKnob<int>(&displ, display::BPM, CONFIRM_PIN,
  &alarm, BPM_RES);
  SafeKnob<float> ie_ = SafeKnob<float>(&displ, display::IE_RATIO,
  CONFIRM_PIN, &alarm, IE_RES);
  SafeKnob<float> ac_ = SafeKnob<float>(&displ, display::AC_TRIGGER,
  CONFIRM_PIN, &alarm, AC_RES);
  void begin();
  void update();
} knobs;

bool patientTriggered = false;

void setState(States newState);
```

```
void calculateWaveform();

void handleErrors();

void setupLogger();

void setup() {
  Serial.begin(SERIAL_BAUD_RATE);
  while(!Serial);

  if (DEBUG) {
    setState(DEBUG_STATE);
  } else {
    setState(PREHOME_STATE);
  }

  delay(1000);

  pinMode(HOME_PIN, INPUT_PULLUP);
  setupLogger();
  alarm.begin();
  displ.begin();
  offButton.begin();
  confirmButton.begin();
  knobs.begin();
  tCycleTimer = now();

  roboclaw.begin(ROBOCLAW_BAUD);
  roboclaw.SetM1MaxCurrent(ROBOCLAW_ADDR, ROBOCLAW_MAX_CURRENT);
  roboclaw.SetM1VelocityPID(ROBOCLAW_ADDR, VKP, VKI, VKD, QPPS);
  roboclaw.SetM1PositionPID(ROBOCLAW_ADDR, PKP, PKI, PKD, KI_MAX, DEADZONE,
  MIN_POS, MAX_POS);
  roboclaw.SetEncM1(ROBOCLAW_ADDR, 0);
}

void loop() {
  if (DEBUG) {
    if (Serial.available() > 0) {
      setState((States) Serial.parseInt());
      while(Serial.available() > 0) Serial.read();
    }
  }
```

```
tLoopTimer = now();
logger.update();
knobs.update();
calculateWaveform();
readEncoder(roboclaw, motorPosition);
readMotorCurrent(roboclaw, motorCurrent);
pressureReader.read();
handleErrors();
alarm.update();
displ.update();
offButton.update();

if (offButton.wasHeld()) {
  goToPositionByDur(roboclaw, BAG_CLEAR_POS, motorPosition, MAX_EX_DURATION);
  setState(OFF_STATE);
  alarm.allOff();
}

switch (state) {

  case DEBUG_STATE:
    roboclaw.ForwardM1(ROBOCLAW_ADDR, 0);
    break;

  case OFF_STATE:
    alarm.turningOFF(now() - tStateTimer < TURNING_OFF_DURATION);
    if (confirmButton.is_LOW()) {
      setState(PREHOME_STATE);
      alarm.turningOFF(false);
    }
    break;

  case IN_STATE:
    if (enteringState) {
      enteringState = false;
      const float tNow = now();
      tPeriodActual = tNow - tCycleTimer;
      tCycleTimer = tNow;
      goToPositionByDur(roboclaw, volume2ticks(knobs.volume()),
      motorPosition, tIn);
      cycleCount++;
```

```
    }

    if (now() - tCycleTimer > tIn) {
      setState(HOLD_IN_STATE);
    }
    break;

  case HOLD_IN_STATE:
    if (enteringState) {
      enteringState = false;
    }
    if (now() - tCycleTimer > tHoldIn) {
      pressureReader.set_plateau();
      setState(EX_STATE);
    }
    break;

  case EX_STATE:
    if (enteringState) {
      enteringState = false;
      goToPositionByDur(roboclaw, BAG_CLEAR_POS, motorPosition, tEx -
      (now() - tCycleTimer));
      displ.hideTimePatientIcon();
    }

    if (abs(motorPosition - BAG_CLEAR_POS) < BAG_CLEAR_TOL) {
      setState(PEEP_PAUSE_STATE);
    }
    break;

  case PEEP_PAUSE_STATE:
    if (enteringState) {
      enteringState = false;
    }

    if (now() - tCycleTimer > tEx + MIN_PEEP_PAUSE) {
      pressureReader.set_peep();
      displ.writePEEP(round(pressureReader.peep()));
      setState(HOLD_EX_STATE);
    }
    break;

  case HOLD_EX_STATE:
```

```
    if (enteringState) {
      enteringState = false;
    }

    patientTriggered = pressureReader.get() < (pressureReader.peep() - knobs.ac())
        && (knobs.ac() > AC_MIN - 1e-2);

    if (patientTriggered || now() - tCycleTimer > tPeriod) {
      if (!patientTriggered) pressureReader.set_peep();
      pressureReader.set_peak_and_reset();
      displ.writePeakP(round(pressureReader.peak()));
      displ.writePEEP(round(pressureReader.peep()));
      displ.writePlateauP(round(pressureReader.plateau()));
      setState(IN_STATE);

      if(patientTriggered) {
        displ.showPatientIcon();
      } else {
        displ.showTimeIcon();
      }
    }
    break;

  case PREHOME_STATE:
    if (enteringState) {
      enteringState = false;
      roboclaw.BackwardM1(ROBOCLAW_ADDR, HOMING_VOLTS);
    }

    if (homeSwitchPressed()) {
      setState(HOMING_STATE);
    }
    break;

  case HOMING_STATE:
    if (enteringState) {
      enteringState = false;
      roboclaw.ForwardM1(ROBOCLAW_ADDR, HOMING_VOLTS);
    }

    if (!homeSwitchPressed()) {
      roboclaw.ForwardM1(ROBOCLAW_ADDR, 0);
      delay(HOMING_PAUSE * 1000);
```

```
        roboclaw.SetEncM1(ROBOCLAW_ADDR, 0);
        setState(IN_STATE);
      }
      break;
  }
  tLoopBuffer = max(0, tLoopTimer + LOOP_PERIOD - now());
  delay(tLoopBuffer*1000.0);
}

void Knobs::begin() {
  volume_.begin(&readVolume);
  bpm_.begin(&readBpm);
  ie_.begin(&readIeRatio);
  ac_.begin(&readAc);
}

void Knobs::update() {
  volume_.update();
  bpm_.update();
  ie_.update();
  ac_.update();
}

inline int Knobs::volume() { return volume_.read(); }
inline int Knobs::bpm() { return bpm_.read(); }
inline float Knobs::ie() { return ie_.read(); }
inline float Knobs::ac() { return ac_.read(); }

void setState(States newState) {
  enteringState = true;
  state = newState;
  tStateTimer = now();
}

void calculateWaveform() {
  tPeriod = 60.0 / knobs.bpm();
  tHoldIn = tPeriod / (1 + knobs.ie());
  tIn = tHoldIn - HOLD_IN_DURATION;
  tEx = min(tHoldIn + MAX_EX_DURATION, tPeriod - MIN_PEEP_PAUSE);
}

void handleErrors() {
  const bool over_pressure = pressureReader.get() >= MAX_PRESSURE;
```

```
  alarm.highPressure(over_pressure);
  if (over_pressure) setState(EX_STATE);

  if (enteringState && state == IN_STATE) {
    alarm.badPlateau(pressureReader.peak() - pressureReader.plateau()
    > MAX_RESIST_PRESSURE);
    alarm.lowPressure(pressureReader.plateau() < MIN_PLATEAU_PRESSURE);
    alarm.noTidalPres(pressureReader.peak() - pressureReader.peep()
    < MIN_TIDAL_PRESSURE);
  }

  if (enteringState && state == EX_STATE) {
    alarm.unmetVolume(knobs.volume() - ticks2volume(motorPosition)
    > VOLUME_ERROR_THRESH);
  }

  if (motorCurrent >= MAX_MOTOR_CURRENT) {
    setState(EX_STATE);
    alarm.overCurrent(true);
  } else {
    alarm.overCurrent(false);
  }

  alarm.mechanicalFailure(state == EX_STATE && now() - tCycleTimer > tPeriod
  + MECHANICAL_TIMEOUT);
}

void setupLogger() {
  logger.addVar("Time", &tLoopTimer);
  logger.addVar("CycleStart", &tCycleTimer);
  logger.addVar("State", (int*)&state);
  logger.addVar("Pos", &motorPosition, 3);
  logger.addVar("Pressure", &pressureReader.get(), 6);
  logger.begin(&Serial, SD_SELECT);
}

#include "Alarms.h"
namespace alarms {

Tone::Tone(const Note notes[], const int& notes_length, const int* pin):
    notes_(notes),
    pin_(pin),
    length_(notes_length),
    tone_step_(length_) {}
```

```
void Tone::play() {
  if (length_ == 0) {
    return;
  }
  if (!playing_) {
    tone_timer_ = millis();
    tone_step_ = 0;
    playing_ = true;
  }
  tone_step_ %= length_;
  if (millis() > tone_timer_) {
    tone(*pin_, notes_[tone_step_].note, notes_[tone_step_].duration);
    tone_timer_ += notes_[tone_step_].duration + notes_[tone_step_].pause;
    tone_step_ ++;
  }
}

void Beeper::begin() {
  snooze_button_.begin();
  pinMode(beeper_pin_, OUTPUT);
}

void Beeper::update(const AlarmLevel& alarm_level) {
  if (snoozeButtonPressed()) {
    toggleSnooze();
  }

  if (snoozed_ && millis() - snooze_time_ > kSnoozeTime) {
    snoozed_ = false;
  }

  if (snoozed_) {
    stop();
    timeRemaining_ = (kSnoozeTime - (millis() - snooze_time_)) / 1000UL;
  } else {
    play(alarm_level);
  }

  if (alarm_level == NO_ALARM) {
    timeRemaining_ = 0;
    snoozed_ = false;
  }
```

```
}

bool Beeper::snoozeButtonPressed() const {
  return snooze_button_.is_LOW();
}

void Beeper::toggleSnooze() {
  if (snoozed_) {
    snoozed_ = false;
    timeRemaining_ = 0;
  } else {
    snoozed_ = true;
    snooze_time_ = millis();
  }
}

void Beeper::play(const AlarmLevel& alarm_level) {
  for (int i = 0; i < NUM_LEVELS; i++) {
    if (i != alarm_level) {
      tones_[i].stop();
    }
  }
  tones_[alarm_level].play();
}

void Beeper::stop() {
  for (int i = 0; i < NUM_LEVELS; i++) {
    tones_[i].stop();
  }
}

unsigned long Beeper::getRemainingSnoozeTime() {
  return timeRemaining_;
}

Alarm::Alarm(const String& default_text, const int& min_bad_to_trigger,
             const int& min_good_to_clear, const AlarmLevel& alarm_level):
  text_(default_text.substring(0, display::kHeaderLength-1)),
  min_bad_to_trigger_(min_bad_to_trigger),
  min_good_to_clear_(min_good_to_clear),
  alarm_level_(alarm_level) {}

void Alarm::reset() {
```

```
  *this = Alarm::Alarm(text_, min_bad_to_trigger_, min_good_to_clear_,
  alarm_level_);
}

void Alarm::setCondition(const bool& bad, const unsigned long& seq) {
  if (bad) {
    consecutive_bad_ += (seq != last_bad_seq_);
    last_bad_seq_ = seq;
    if (!on_) {
      on_ = consecutive_bad_ >= min_bad_to_trigger_;
    }
    consecutive_good_ = 0;
  } else {
    consecutive_good_ += (seq != last_good_seq_);
    last_good_seq_ = seq;
    if (on_) {
      on_ = consecutive_good_ < min_good_to_clear_;
    }
    consecutive_bad_ = 0;
  }
}

void Alarm::setText(const String& text) {
  if (text.length() == display::kFooterLength) {
    text_ = text;
  }
  else if (text.length() > display::kFooterLength) {
    text_ = text.substring(0, display::kFooterLength);
  }
  else {
    text_ = text;
    while (text_.length() < display::kFooterLength) {
      text_ += " ";
    }
  }
}

void AlarmManager::begin() {
  beeper_.begin();
  pinMode(led_pin_, OUTPUT);
}

void AlarmManager::update() {
```

```
  displ_->setAlarmText(getGeneralAlarmText());
  displ_->setUnconfirmedKnobAlarmText(getUnconfirmedKnobText());
  displ_->setSnoozeText(beeper_.getRemainingSnoozeTime());
  AlarmLevel highest_level = getHighestLevel();
  beeper_.update(highest_level);
  if (highest_level > NO_ALARM) {
    digitalWrite(led_pin_, led_pulse_.read() ? HIGH : LOW);
  }
  else {
    digitalWrite(led_pin_, LOW);
  }
}

void AlarmManager::allOff() {
  for (int i = 0; i < NUM_ALARMS; i++) {
    alarms_[i].reset();
  }
}

int AlarmManager::numON() const {
  int num = 0;
  for (int i = 0; i < NUM_ALARMS; i++) {
    num += (int)alarms_[i].isON();
  }
  return num;
}

int AlarmManager::numON_Confirm() const {
  int num = 0;
  bool aConfirmAlarm;
  for (int i = 0; i < NUM_ALARMS; i++) {
    aConfirmAlarm = (i==NOT_CONFIRM_TV || i==NOT_CONFIRM_RR ||
                     i==NOT_CONFIRM_IE || i==NOT_CONFIRM_AC);
    if(aConfirmAlarm) {
      num += (int)alarms_[i].isON();
    }
  }
  return num;
}

int AlarmManager::numON_NonConfirm() const {
  int num = 0;
  bool aConfirmAlarm;
```

```
  for (int i = 0; i < NUM_ALARMS; i++) {
    aConfirmAlarm = (i==NOT_CONFIRM_TV || i==NOT_CONFIRM_RR ||
                     i==NOT_CONFIRM_IE || i==NOT_CONFIRM_AC);
    if(!aConfirmAlarm) {
      num += (int)alarms_[i].isON();
    }
  }
  return num;
}

String AlarmManager::getGeneralAlarmText() const {
  const int num_on = numON_NonConfirm();
  String text = "";
  if (num_on > 0) {
    const int index = millis() % (num_on * kDisplayTime) / kDisplayTime;
    int count_on = 0;
    int i;
    bool aConfirmAlarm;
    for (i = 0; i < NUM_ALARMS; i++) {
      aConfirmAlarm = (i==NOT_CONFIRM_TV || i==NOT_CONFIRM_RR ||
                       i==NOT_CONFIRM_IE || i==NOT_CONFIRM_AC);
      if (!aConfirmAlarm && alarms_[i].isON()) {
        if (count_on++ == index) break;
      }
    }
    text = alarms_[i].text();
  }
  return text;
}

String AlarmManager::getUnconfirmedKnobText() const {
  const int num_on = numON_Confirm();
  String text = "";
  if (num_on > 0) {
    const int index = millis() % (num_on * kDisplayTime) / kDisplayTime;
    int count_on = 0;
    int i;
    bool aConfirmAlarm;
    for (i = 0; i < NUM_ALARMS; i++) {
      aConfirmAlarm = (i==NOT_CONFIRM_TV || i==NOT_CONFIRM_RR ||
                       i==NOT_CONFIRM_IE || i==NOT_CONFIRM_AC);
      if (aConfirmAlarm && alarms_[i].isON()) {
        if (count_on++ == index) break;
```

```
      }
    }
    text = alarms_[i].text();
  }
  return text;
}

AlarmLevel AlarmManager::getHighestLevel() const {
  AlarmLevel alarm_level = NO_ALARM;
  for (int i = 0; i < NUM_ALARMS; i++) {
    if (alarms_[i].isON()) {
      alarm_level = max(alarm_level, alarms_[i].alarmLevel());
    }
  }
  return alarm_level;
}

}

#ifndef Alarms_h
#define Alarms_h
#include "Arduino.h"
#include "Buttons.h"
#include "Display.h"
#include "pitches.h"

namespace alarms {

using display::Display;

enum AlarmLevel {
  NO_ALARM,
  NOTIFY,
  EMERGENCY,
  OFF_LEVEL,
  NUM_LEVELS
};

struct Note {
  int note;
  int duration;
  int pause;
};
```

```
static const Note kNotifyNotes[] = {
  {NOTE_B4, 200, 100},
  {NOTE_B4, 200, 2000}
};

static const Note kEmergencyNotes[] = {
  {NOTE_G4, 300, 200},
  {NOTE_G4, 300, 200},
  {NOTE_G4, 300, 400},
  {NOTE_G4, 200, 100},
  {NOTE_G5, 200, 1500}
};

static const Note kOffNotes[] = {
  {NOTE_G4, 200, 200}
};

class Tone {
public:
  Tone(): length_(0) {}

  Tone(const Note notes[], const int& notes_length, const int* pin);

  void play();

  inline void stop() { playing_ = false; }

private:
  Note* notes_;
  int length_;
  int* pin_;
  bool playing_ = false;
  int tone_step_;
  unsigned long tone_timer_ = 0;
};

class Beeper {

  static const unsigned long kSnoozeTime = 2 * 60 * 1000UL;

public:
  Beeper(const int& beeper_pin, const int& snooze_pin):
    beeper_pin_(beeper_pin),
```

```
    snooze_button_(snooze_pin) {
      const int notify_notes_length = sizeof(kNotifyNotes) /
      sizeof(kNotifyNotes[0]);
      tones_[NOTIFY] = Tone(kNotifyNotes, notify_notes_length,
      &beeper_pin_);

      const int emergency_notes_length = sizeof(kEmergencyNotes) /
      sizeof(kEmergencyNotes[0]);
      tones_[EMERGENCY] = Tone(kEmergencyNotes, emergency_notes_length,
      &beeper_pin_);

      const int off_notes_length = sizeof(kOffNotes) / sizeof(kOffNotes[0]);
      tones_[OFF_LEVEL] = Tone(kOffNotes, off_notes_length, &beeper_pin_);
    }

  void begin();

  void update(const AlarmLevel& alarm_level);

  unsigned long getRemainingSnoozeTime();

private:
  const int beeper_pin_;
  buttons::DebouncedButton snooze_button_;
  Tone tones_[NUM_LEVELS];

  unsigned long snooze_time_ = 0;
  unsigned long timeRemaining_ = 0;
  bool snoozed_ = false;

  bool snoozeButtonPressed() const;

  void toggleSnooze();

  void play(const AlarmLevel& alarm_level);

  void stop();

};

class Alarm {
public:
  Alarm() {};
```

```
  Alarm(const String& default_text, const int& min_bad_to_trigger,
        const int& min_good_to_clear, const AlarmLevel& alarm_level);

  void reset();

  void setText(const String& text);

  inline const bool& isON() const { return on_; }

  inline String text() const { return text_; }

  inline AlarmLevel alarmLevel() const { return alarm_level_; }

private:
  String text_;
  AlarmLevel alarm_level_;
  int min_bad_to_trigger_;
  int min_good_to_clear_;
  bool on_ = false;
  unsigned long consecutive_bad_ = 0;
  unsigned long consecutive_good_ = 0;
  unsigned long last_bad_seq_ = 0;
  unsigned long last_good_seq_ = 0;
};

class AlarmManager {

  static const unsigned long kDisplayTime = 2 * 1000UL;

  enum Indices {
    HIGH_PRESSU = 0,
    LOW_PRESSUR,
    BAD_PLATEAU,
    UNMET_VOLUM,
    NO_TIDAL_PR,
    OVER_CURREN,
    MECH_FAILUR,
    NOT_CONFIRM_TV,
    NOT_CONFIRM_RR,
    NOT_CONFIRM_IE,
    NOT_CONFIRM_AC,
    TURNING_OFF,
```

```
    NUM_ALARMS
  };

  static const int numConfirmKnob = 4;

public:
  AlarmManager(const int& beeper_pin, const int& snooze_pin, const int& led_pin,
               Display* displ, unsigned long const* cycle_count):
      displ_(displ),
      beeper_(beeper_pin, snooze_pin),
      led_pin_(led_pin),
      led_pulse_(500, 0.5),
      cycle_count_(cycle_count) {
    alarms_[HIGH_PRESSU] = Alarm("PRESION ALTA        ", 1, 2, EMERGENCY);
    alarms_[LOW_PRESSUR] = Alarm("PRESION BAJA DESCON?", 1, 1, EMERGENCY);
    alarms_[BAD_PLATEAU] = Alarm("ALTA PRESION RESISTE", 1, 1, NOTIFY);
    alarms_[UNMET_VOLUM] = Alarm("V TIDAL NO ALCANZADO", 1, 1, EMERGENCY);
    alarms_[NO_TIDAL_PR] = Alarm("NO PRESION TIDAL    ", 2, 1, EMERGENCY);
    alarms_[OVER_CURREN] = Alarm("FALLO SOBRECORRIENTE", 1, 2, EMERGENCY);
    alarms_[MECH_FAILUR] = Alarm("FALLO MECANICO      ", 1, 1, EMERGENCY);
    alarms_[NOT_CONFIRM_TV] = Alarm("CONFIRMAR?          ", 1, 1, NOTIFY);
    alarms_[NOT_CONFIRM_RR] = Alarm("CONFIRMAR?          ", 1, 1, NOTIFY);
    alarms_[NOT_CONFIRM_IE] = Alarm("CONFIRMAR?          ", 1, 1, NOTIFY);
    alarms_[NOT_CONFIRM_AC] = Alarm("CONFIRMAR?          ", 1, 1, NOTIFY);
    alarms_[TURNING_OFF] = Alarm("APAGANDO            ", 1, 1, OFF_LEVEL);
  }

  void begin();

  void update();

  void allOff();

  inline void highPressure(const bool& value) {
    alarms_[HIGH_PRESSU].setCondition(value, *cycle_count_);
  }

  inline void lowPressure(const bool& value) {
    alarms_[LOW_PRESSUR].setCondition(value, *cycle_count_);
  }

  inline void badPlateau(const bool& value) {
    alarms_[BAD_PLATEAU].setCondition(value, *cycle_count_);
```

```
}

inline void unmetVolume(const bool& value) {
  alarms_[UNMET_VOLUM].setCondition(value, *cycle_count_);
}

inline void noTidalPres(const bool& value) {
  alarms_[NO_TIDAL_PR].setCondition(value, *cycle_count_);
}

inline void overCurrent(const bool& value) {
  alarms_[OVER_CURREN].setCondition(value, *cycle_count_);
}

inline void mechanicalFailure(const bool& value) {
  alarms_[MECH_FAILUR].setCondition(value, *cycle_count_);
}

inline void unconfirmedChange(const bool& value, const String& message =
"", const display::DisplayKey& key = 0) {
  Indices NOT_CONFIRM;
  if (key == display::DisplayKey::VOLUME)      { NOT_CONFIRM = NOT_CONFIRM_TV;}
  if (key == display::DisplayKey::BPM)         { NOT_CONFIRM = NOT_CONFIRM_RR;}
  if (key == display::DisplayKey::IE_RATIO)    { NOT_CONFIRM = NOT_CONFIRM_IE;}
  if (key == display::DisplayKey::AC_TRIGGER)  { NOT_CONFIRM = NOT_CONFIRM_AC;}
  if (value) {
    alarms_[NOT_CONFIRM].setText(message);
  }
  alarms_[NOT_CONFIRM].setCondition(value, *cycle_count_);
}

inline void turningOFF(const bool& value) {
  alarms_[TURNING_OFF].setCondition(value, *cycle_count_);
}

inline const bool& getHighPressure()      { return alarms_[HIGH_PRESSU].isON(); }
inline const bool& getLowPressure()       { return alarms_[LOW_PRESSUR].isON(); }
inline const bool& getBadPlateau()        { return alarms_[BAD_PLATEAU].isON(); }
inline const bool& getUnmetVolume()       { return alarms_[UNMET_VOLUM].isON(); }
inline const bool& getNoTidalPres()       { return alarms_[NO_TIDAL_PR].isON(); }
inline const bool& getOverCurrent()       { return alarms_[OVER_CURREN].isON(); }
inline const bool& getMechanicalFailure() { return alarms_[MECH_FAILUR].isON(); }
inline const bool& getTurningOFF()        { return alarms_[TURNING_OFF].isON(); }
```

```
private:
  Display* displ_;
  Beeper beeper_;
  int led_pin_;
  utils::Pulse led_pulse_;
  Alarm alarms_[NUM_ALARMS];
  Alarm alarmsHeader_[NUM_ALARMS-numConfirmKnob];
  Alarm alarmsFooter_[numConfirmKnob];
  unsigned long const* cycle_count_;

  int numON() const;

  int numON_Confirm() const;

  int numON_NonConfirm() const;

  String getGeneralAlarmText() const;

  String getUnconfirmedKnobText() const;

  AlarmLevel getHighestLevel() const;
};


}

#endif

#include "Buttons.h"
namespace buttons {

void PressHoldButton::update() {
  const unsigned long time_now = millis();
  const unsigned long time_since_last_low = time_now - last_low_time_;
  const bool press_lost = time_since_last_low > kDebounceDelay;
  if (digitalRead(pin_) == LOW) {
    if (!press_lost) {
      current_hold_time_ += time_since_last_low;
    }
    last_low_time_ = time_now;
  }
  else if (press_lost) {
    current_hold_time_ = 0;
```

```
  }
}

DebouncedButton::DebouncedButton(const int& pin): pin_(pin) {}

void DebouncedButton::begin() {
  pinMode(pin_, INPUT_PULLUP);
}

bool DebouncedButton::is_LOW() {
  int reading = digitalRead(pin_);

  bool low_value = false;
  const unsigned long time_now = millis();
  if (reading == LOW) {
    if ((time_now - last_low_time_) > kDebounceDelay) {
      low_value = true;
    }
    last_low_time_ = time_now;
  }
  return low_value;
}


}

#ifndef Buttons_h
#define Buttons_h

#include "Arduino.h"
namespace buttons {

class PressHoldButton {

  const unsigned long kDebounceDelay = 100;

public:
  PressHoldButton(const int& pin, const unsigned long& hold_duration):
    pin_(pin),
    hold_duration_(hold_duration) {}

  inline void begin() { pinMode(pin_, INPUT_PULLUP); }

  void update();
```

```
  inline bool wasHeld() { return current_hold_time_ > hold_duration_; }

private:
  int pin_;
  unsigned long hold_duration_;
  unsigned long last_low_time_ = 0;
  unsigned long current_hold_time_ = 0;
};

class DebouncedButton {

  const unsigned long kDebounceDelay = 100;

public:
  DebouncedButton(const int& pin);

  void begin();

  bool is_LOW();

private:
  int pin_;
  unsigned long last_low_time_ = 0;
};


}


#endif

#ifndef Constants_h
#define Constants_h

enum States {
  DEBUG_STATE,
  IN_STATE,
  HOLD_IN_STATE,
  EX_STATE,
  PEEP_PAUSE_STATE,
  HOLD_EX_STATE,
  PREHOME_STATE,
  HOMING_STATE,
  OFF_STATE
```

```
};

const long SERIAL_BAUD_RATE = 115200;

const bool DEBUG = false;
const bool ASSIST_CONTROL = false;

const float LOOP_PERIOD = 0.03;
const float HOLD_IN_DURATION = 0.1;
const float MIN_PEEP_PAUSE = 0.05;
const float MAX_EX_DURATION = 1.00;

const float HOMING_VOLTS = 30;
const float HOMING_PAUSE = 1.0;
const int BAG_CLEAR_POS = 50;
const int BAG_CLEAR_TOL = 10;

const int VOL_PIN = A0;
const int BPM_PIN = A1;
const int IE_PIN = A2;
const int AC_PIN = A3;
const int PRESS_SENSE_PIN = A4;
const int HOME_PIN = 10;
const int BEEPER_PIN = 11;
const int SNOOZE_PIN = 43;
const int CONFIRM_PIN = 41;
const int SD_SELECT = 53;
const int OFF_PIN = 45;
const int LED_ALARM_PIN = 12;
const int LCD_RS_PIN = 9;
const int LCD_EN_PIN = 8;
const int LCD_D4_PIN = 7;
const int dLCD_D5_PIN = 6;
const int LCD_D6_PIN = 5;
const int LCD_D7_PIN = 4;

const int BPM_MIN = 6;
const int BPM_MAX = 35;
const int BPM_RES = 1;
const float IE_MIN = 1;
const float IE_MAX = 4;
const float IE_RES = 0.1;
const int VOL_MIN = 100;
```

```
const int VOL_MAX = 800;
const int VOL_RES = 25;
const float AC_MIN = 2;
const float AC_MAX = 7;
const float AC_RES = 0.5;
const int ANALOG_PIN_MAX = 1023;

const struct {float a, b, c;} COEFFS{1.29083271e-03, 4.72985182e-01, -7.35403067e+01};

const float MAX_PRESSURE = 40.0;
const float MIN_PLATEAU_PRESSURE = 5.0;
const float MAX_RESIST_PRESSURE = 10.0;
const float MIN_TIDAL_PRESSURE = 5.0;
const float VOLUME_ERROR_THRESH = 50.0;
const int MAX_MOTOR_CURRENT = 1000;
const float TURNING_OFF_DURATION = 5.0;
const float MECHANICAL_TIMEOUT = 1.0;

const unsigned long QPPS = 2000;
const float VKP = 6.38650;
const float VKI = 0.0;
const float VKD = 0.0;
const float PKP = 70.0;
const float PKI = 0.2;
const float PKD = 200.0;
const unsigned long KI_MAX = 10;
const unsigned long DEADZONE = 0;
const unsigned long MIN_POS = -100;
const unsigned long MAX_POS = 700;
const unsigned long VEL_MAX = 1800;
const unsigned long ACC_MAX = 200000;

const unsigned int ROBOCLAW_ADDR = 0x80;
const long ROBOCLAW_BAUD = 38400;
const unsigned long ROBOCLAW_MAX_CURRENT = 2000;

#endif


#include "Display.h"

namespace display {
```

```
void Display::begin() {
  lcd_->begin(kWidth, kHeight);
  lcd_->noCursor();
  addCustomCharacters();
  update();
}

void Display::addCustomCharacters() {
  lcd_->createChar(0, patientIcon);
  lcd_->createChar(1, timeIcon);
  lcd_->createChar(2, peakIcon);
  lcd_->createChar(3, plateauIcon);
  lcd_->createChar(4, peepIcon);
  lcd_->createChar(5, ieiIcon);
  lcd_->createChar(6, ie1Icon);
  lcd_->createChar(7, bellIcon);
}

void Display::update() {
  writeHeader();
  writeFooter();
}

void Display::setAlarmText(const String& alarm) {
  if (animation_.text() != alarm) {
    animation_.reset(alarm);
  }
}

void Display::setUnconfirmedKnobAlarmText(const String& alarm) {
  if (animationfooter_.text() != alarm) {
    animationfooter_.reset(alarm);
  }
}

void Display::setSnoozeText(const int& countdown) {
  snoozecountdown_ = countdown;
}

template <typename T>
void Display::write(const DisplayKey& key, const T& value) {
  if (alarmsON() && elements_[key].row == 0 && key != HEADER) {
    return;
```

```
  }
  switch (key) {
    case HEADER:
      writeHeader();
      break;
    case VOLUME:
      writeVolume(value);
      break;
    case BPM:
      writeBPM(value);
      break;
    case IE_RATIO:
      writeIEratio(value);
      break;
    case AC_TRIGGER:
      writeACTrigger(value);
      break;
    case PEAK_PRES:
      writePeakP(value);
      break;
    case PLATEAU_PRES:
      writePlateauP(value);
      break;
    case PEEP_PRES:
      writePEEP(value);
      break;
  }
}

void Display::writeBlank(const DisplayKey& key) {
  if (alarmsON() && elements_[key].row == 0 && key != HEADER) {
    return;
  }
  write(elements_[key].row, elements_[key].col, elements_[key].blank);
}

void Display::writeHeader() {
  if (!alarmsON()) {
    write(elements_[HEADER].row, elements_[HEADER].col, "Running...         ");
  }
  else {
    const String line = animation_.getLine();
    if (line.length() > 0) {
```

```
      write(elements_[HEADER].row, elements_[HEADER].col, line);
    }
    else {
      writeBlank(HEADER);
    }
  }
}

void Display::writeFooter() {
  if (!alarmsON()) {
    writeBlank(FOOTER);
    hideBellIcon();
  }
  else {
    showBellIcon();
    const String line = animationfooter_.getLine();
    if (line.length() > 0) {
      write(elements_[FOOTER].row, elements_[FOOTER].col, line);
    }
    else {
      writeBlank(FOOTER);
    }
  }
  writeSnoozeTime();
}

void Display::writeSnoozeTime() {
  char buff[7];
  if (snoozecountdown_>0){
    sprintf(buff, "%3lu%   ", snoozecountdown_);
  } else {
    sprintf(buff, "%6s", "      ");
  }
  write(elements_[SNOOZE].row, elements_[SNOOZE].col, buff);
}

void Display::writeVolume(const int& vol) {
  const int vol_c = constrain(vol, 0, 999);
  char buff[5];
  sprintf(buff, "%2s  ", getLabel(VOLUME).c_str());
  write(elements_[VOLUME].row, elements_[VOLUME].col, buff);
  sprintf(buff, "%3s ", toString(VOLUME, vol_c).c_str());
  write(elements_[VOLUME].row+1, elements_[VOLUME].col, buff);
```

```
}

void Display::writeBPM(const int& bpm) {
  const int bpm_c = constrain(bpm, 0, 99);
  char buff[4];
  sprintf(buff, "%2s ", getLabel(BPM).c_str());
  write(elements_[BPM].row, elements_[BPM].col, buff);
  sprintf(buff, "%2s ", toString(BPM, bpm_c).c_str());
  write(elements_[BPM].row+1, elements_[BPM].col, buff);
}

void Display::writeIEratio(const float& ie) {
  const float ie_c = constrain(ie, 0.0, 9.9);
  showIEiIcon(elements_[IE_RATIO].row, elements_[IE_RATIO].col);
  showIE1Icon(elements_[IE_RATIO].row+1, elements_[IE_RATIO].col);
  char buff[5];
  sprintf(buff, "E   ");
  write(elements_[IE_RATIO].row, elements_[IE_RATIO].col+1, buff);
  sprintf(buff, "%3s ", toString(IE_RATIO, ie_c).c_str());
  write(elements_[IE_RATIO].row+1, elements_[IE_RATIO].col+1, buff);
}

void Display::writeACTrigger(const float& ac_trigger) {
  const float ac_trigger_c = constrain(ac_trigger, 0.0, 9.9);
  char buff[6];
  const String trigger_str = toString(AC_TRIGGER, ac_trigger_c);
  sprintf(buff, "%3s  ", getLabel(AC_TRIGGER).c_str());
  write(elements_[AC_TRIGGER].row, elements_[AC_TRIGGER].col, buff);
  sprintf(buff, "%3s  ", trigger_str.c_str());
  write(elements_[AC_TRIGGER].row+1, elements_[AC_TRIGGER].col, buff);
}

void Display::writePeakP(const int& peak) {
  const int peak_c = constrain(peak, -9, 99);
  showPeakIcon(elements_[PEAK_PRES].row, elements_[PEAK_PRES].col);
  char buff[3];
  sprintf(buff, "%2s", toString(PEAK_PRES, peak_c).c_str());
  write(elements_[PEAK_PRES].row, elements_[PEAK_PRES].col+1, buff);
}

void Display::writePlateauP(const int& plat) {
  const int plat_c = constrain(plat, -9, 99);
  showPlateauIcon(elements_[PLATEAU_PRES].row, elements_[PLATEAU_PRES].col);
```

```
  char buff[3];
  sprintf(buff, "%2s", toString(PLATEAU_PRES, plat_c).c_str());
  write(elements_[PLATEAU_PRES].row, elements_[PLATEAU_PRES].col+1, buff);
}

void Display::writePEEP(const int& peep) {
  const int peep_c = constrain(peep, -9, 99);
  showPEEPIcon(elements_[PEEP_PRES].row, elements_[PEEP_PRES].col);
  char buff[3];
  sprintf(buff, "%2s", toString(PEEP_PRES, peep_c).c_str());
  write(elements_[PEEP_PRES].row, elements_[PEEP_PRES].col+1, buff);
}

template <typename T>
String Display::toString(const DisplayKey& key, const T& value) const {
  switch (key) {
    case VOLUME:
      return String(value);
    case BPM:
      return String(value);
    case IE_RATIO:
      return String(value, 1);
    case AC_TRIGGER:
      return (value > trigger_threshold_ - 1e-2) ? String(value, 1) : "OFF";
    case PEAK_PRES:
      return String(value);
    case PLATEAU_PRES:
      return String(value);
    case PEEP_PRES:
      return String(value);
    case SNOOZE:
      return String(value);
    default:
      return "N/A";
  }
}

void Display::showPatientIcon() {
  lcd_->setCursor(elements_[BREATHING].col, elements_[BREATHING].row);
  lcd_->write(byte(0));
}

void Display::showTimeIcon() {
```

```
  lcd_->setCursor(elements_[BREATHING].col, elements_[BREATHING].row);
  lcd_->write(byte(1));
}

void Display::hideTimePatientIcon() {
  hideIcon(elements_[BREATHING].row, elements_[BREATHING].col);
}

void Display::hideIcon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(32));
}

void Display::showPeakIcon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(2));
}

void Display::showPlateauIcon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(3));
}

void Display::showPEEPIcon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(4));
}

void Display::showIEiIcon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(5));
}

void Display::showIE1Icon(const int& row, const int& col) {
  lcd_->setCursor(col, row);
  lcd_->write(byte(6));
}

void Display::showBellIcon() {
  lcd_->setCursor(elements_[BELL].col, elements_[BELL].row);
  lcd_->write(byte(7));
}
```

```
void Display::hideBellIcon() {
  hideIcon(elements_[BELL].row, elements_[BELL].col);
}

template <typename T>
void Display::write(const int& row, const int& col, const T& printable) {
  lcd_->setCursor(col, row);
  lcd_->print(printable);
}


}

#ifndef Display_h
#define Display_h

#include "Arduino.h"
#include <LiquidCrystal.h>

#include "Utilities.h"

#include "Indicators.h"

namespace display {

static const int kWidth = 20;
static const int kHeight = 4;
static const int kHeaderLength = 19;
static const int kFooterLength = 10;

class TextAnimation {
public:
  TextAnimation(const unsigned long& period, const float& on_fraction):
      pulse_(period, on_fraction) {}

  inline void reset(const String& text = "") { text_ = text; }

  inline bool empty() { return text_.length() == 0; }

  inline const String& text() { return text_; }

  inline const String getLine() { return pulse_.read() ? text_ : ""; }

private:
```

```
  String text_;
  utils::Pulse pulse_;
  unsigned long reset_time_;
};


enum DisplayKey {
  HEADER,
  BREATHING,
  VOLUME,
  BPM,
  IE_RATIO,
  AC_TRIGGER,
  PRES_LABEL,
  PEAK_PRES,
  PLATEAU_PRES,
  PEEP_PRES,
  SNOOZE,
  BELL,
  FOOTER,
  NUM_KEYS
};

class Display {

  struct Element {
    Element() = default;

    Element(const int& r, const int& c, const int& w, const String& l = ""):
        row(r), col(c), width(w), label(l) {
      while (blank.length() < width) blank += " ";
    }
    int row;
    int col;
    int width;
    String label;
    String blank;
  };

public:
  Display(LiquidCrystal* lcd, const float& trigger_threshold):
      lcd_(lcd),
      trigger_threshold_(trigger_threshold),
```

```
    animation_(1000, 0.5),
    animationfooter_(1000, 0.5),
    snoozecountdown_(0) {
  elements_[HEADER]       = Element{0, 0, kHeaderLength};
  elements_[BREATHING]    = Element{0, 19, 1};
  elements_[VOLUME]       = Element{1, 0, 4, "TV"};
  elements_[BPM]          = Element{1, 4, 3, "RR"};
  elements_[IE_RATIO]     = Element{1, 7, 5, "IE"};
  elements_[AC_TRIGGER]   = Element{1, 12, 5, "Ptr"};
  elements_[PEAK_PRES]    = Element{1, 17, 3, "peak"};
  elements_[PLATEAU_PRES] = Element{2, 17, 3, "plat"};
  elements_[PEEP_PRES]    = Element{3, 17, 3, "PEEP"};
  elements_[SNOOZE]       = Element{3, 1, 3};
  elements_[BELL]         = Element{3, 0, 1};
  elements_[FOOTER]       = Element{3, 7, kFooterLength};
}

void begin();

void addCustomCharacters();

void update();

void setAlarmText(const String& alarm);

void setUnconfirmedKnobAlarmText(const String& alarm);

void setSnoozeText(const int& countdown);

template <typename T>
void write(const DisplayKey& key, const T& value);

void writeBlank(const DisplayKey& key);

void writeHeader();

void writeFooter();

void writeSnoozeTime();

void writeVolume(const int& vol);

void writeBPM(const int& bpm);
```

```
  void writeIEratio(const float& ie);

  void writeACTrigger(const float& ac_trigger);

  void writePeakP(const int& peak);

  void writePlateauP(const int& plat);

  void writePEEP(const int& peep);

  void showPatientIcon();

  void showTimeIcon();

  void hideTimePatientIcon();

  void hideIcon(const int& row, const int& col);

  void showPeakIcon(const int& row, const int& col);

  void showPlateauIcon(const int& row, const int& col);

  void showPEEPIcon(const int& row, const int& col);

  void showIEiIcon(const int& row, const int& col);

  void showIE1Icon(const int& row, const int& col);

  void showBellIcon();

  void hideBellIcon();

  template <typename T>
  String toString(const DisplayKey& key, const T& value) const;

  inline String getLabel(const DisplayKey& key) const { return
  elements_[key].label; };

private:
  LiquidCrystal* lcd_;
  const float trigger_threshold_;
  TextAnimation animation_;
```

```
  TextAnimation animationfooter_;
  unsigned long snoozecountdown_;
  Element elements_[NUM_KEYS];

  template <typename T>
  void write(const int& row, const int& col, const T& printable);

  inline bool alarmsON() const { return (!animation_.empty() ||
  !animationfooter_.empty()); }
};

#define INSTANTIATE_WRITE(type) \
  template void Display::write(const DisplayKey& key, const type& value);
INSTANTIATE_WRITE(int)
INSTANTIATE_WRITE(float)
#undef INSTANTIATE_WRITE


}

#endif

#ifndef Indicators_h
#define Indicators_h


static const byte patientIcon[8] = {
    B01110,
    B01110,
    B00100,
    B11111,
    B11111,
    B11111,
    B01010,
    B01010};

static const byte timeIcon[8] = {
    B00000,
    B11111,
    B11111,
    B01110,
    B00100,
    B01110,
    B11111,
```

```
    B11111};

static const byte peakIcon[8] = {
    B01111,
    B10000,
    B10000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000};

static const byte plateauIcon[8] = {
    B00000,
    B10000,
    B10000,
    B01110,
    B00001,
    B00001,
    B00000,
    B00000};

static const byte peepIcon[8] = {
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B10000,
    B10000,
    B01111};

static const byte ieiIcon[8] = {
    B11100,
    B01011,
    B01011,
    B01000,
    B01011,
    B01011,
    B11100,
    B00000};

static const byte ie1Icon[8] = {
```

```
    B01000,
    B11011,
    B01011,
    B01000,
    B01011,
    B01011,
    B11100,
    B00000};

static const byte bellIcon[8] = {
    B00000,
    B00100,
    B01110,
    B01110,
    B01110,
    B11111,
    B00100,
    B00000};

    #endif

#include "Input.h"


namespace input {

template <typename T>
void Input<T>::begin(float (*read_fun)()) {
  read_fun_ = read_fun;
  set_value_ = discretize(read_fun_());
}

template <typename T>
T Input<T>::discretize(const float& raw_value) const {
  return round(raw_value / resolution_) * resolution_;
}

template <typename T>
void Input<T>::display(const T& value, const bool& blank) {
  const unsigned long time_now = millis();
  if (time_now - last_display_update_time_ < kDisplayUpdatePeriod) return;
  last_display_update_time_ = time_now;

  if (blank) {
```

```
    displ_->writeBlank(disp_key_);
  }
  else {
    displ_->write(disp_key_, value);
  }
}

template <typename T>
void Knob<T>::update() {
  this->set_value_ = discretize(read_fun_());
  this->display(read());
}

template <typename T>
void SafeKnob<T>::begin(T (*read_fun)()) {
  Input<T>::begin(read_fun);
  confirm_button_.begin();
}

template <typename T>
void SafeKnob<T>::update() {
  unconfirmed_value_ = discretize(read_fun_());
  if (abs(unconfirmed_value_ - this->set_value_) <= this->resolution_ / 2) {
    this->display(read());
    if (!confirmed_) {
      confirmed_ = true;
      alarms_->unconfirmedChange(false,getConfirmPrompt(),this->disp_key_);
    }
  }
  else if (confirm_button_.is_LOW()) {
    this->set_value_ = unconfirmed_value_;
  }
  else {
    const unsigned long time_now = millis();
    if (confirmed_) {
      time_changed_ = time_now;
      confirmed_ = false;
    }
    this->display(unconfirmed_value_, !pulse_.read());
    if (time_now - time_changed_ > kAlarmTime) {
      alarms_->unconfirmedChange(true, getConfirmPrompt(),this->disp_key_);
    }
  }
```

```
}

template <typename T>
String SafeKnob<T>::getConfirmPrompt() const {
  char buff[display::kWidth + 1];
  sprintf(buff, "%s->%s?", this->getLabel().c_str(),
  this->toString(unconfirmed_value_).c_str());
  String s(buff);
  return s;
}



}

#ifndef Input_h
#define Input_h

#include "Arduino.h"
#include "Alarms.h"
#include "Buttons.h"
#include "Display.h"
#include "Utilities.h"


namespace input {


using alarms::AlarmManager;
using display::Display;

template <typename T>
class Input {
  static const unsigned long kDisplayUpdatePeriod = 250;

public:
  Input(Display* displ, const display::DisplayKey& key, const T& resolution):
      displ_(displ),
      disp_key_(key),
      resolution_(resolution) {}

  void begin(float (*read_fun)());

  virtual void update() = 0;
```

```
  inline T read() const& { return set_value_; }

protected:
  float (*read_fun_)();
  Display* displ_;
  display::DisplayKey disp_key_;
  T resolution_;

  T set_value_;
  unsigned long last_display_update_time_ = 0;

  T discretize(const float& raw_value) const;

  void display(const T& value, const bool& blank = false);

  inline String toString(const T& val) const { return displ_->toString(disp_key_,
  val); }

  inline String getLabel() const { return displ_->getLabel(disp_key_); }
};

template <typename T>
class Knob : public Input<T> {
public:
  Knob(Display* displ, const display::DisplayKey& key, const T& resolution):
      Input<T>(displ, key, resolution) {}

  void update();
};

template class Knob<int>;
template class Knob<float>;

template <typename T>
class SafeKnob : public Input<T> {

  static const unsigned long kAlarmTime = 5 * 1000UL;

public:
  SafeKnob(Display* displ, const display::DisplayKey& key,
           const int& confirm_pin, AlarmManager* alarms, const T& resolution):
      Input<T>(displ, key, resolution),
      confirm_button_(confirm_pin),
```

```
      alarms_(alarms),
      pulse_(1000, 0.5) {}

  void begin(T (*read_fun)());

  void update();

private:
  buttons::DebouncedButton confirm_button_;
  AlarmManager* alarms_;
  utils::Pulse pulse_;
  T unconfirmed_value_;
  unsigned long time_changed_ = 0;
  bool confirmed_ = true;

  String getConfirmPrompt() const;
};

template class SafeKnob<int>;
template class SafeKnob<float>;


}


#endif

#include "Logging.h"

namespace logging {

template <typename T>
Var::Var(const String& label, T* var, const int& min_digits,
const int& float_precision):
    label_(label),
    min_digits_(min_digits),
    float_precision_(float_precision) {
  setPtr(var);
}

String Var::serialize() const {
  switch (type_) {
    case BOOL:
      return serialize(var_.b);
```

```
    case INT:
      return serialize(var_.i);
    case FLOAT:
      return serialize(var_.f);
    case DOUBLE:
      return serialize(var_.d);
  }
}

String Var::pad(String& s) {
  while (s.length() < min_digits_) {
    s = " " + s;
  }
  return s;
}

void Var::setPtr(const bool* var) {
  var_.b = var;
  type_ = BOOL;
}

void Var::setPtr(const int* var) {
  var_.i = var;
  type_ = INT;
}

void Var::setPtr(const float* var) {
  var_.f = var;
  type_ = FLOAT;
}

void Var::setPtr(const double* var) {
  var_.d = var;
  type_ = DOUBLE;
}

String Var::serialize(bool* var) const {
  String string_out((int)*var);
  return pad(string_out);
}

String Var::serialize(int* var) const {
  String string_out(*var);
```

```
  return pad(string_out);
}

String Var::serialize(float* var) const {
  String string_out(*var, float_precision_);
  return pad(string_out);
}

String Var::serialize(double* var) const {
  String string_out(*var, float_precision_);
  return pad(string_out);
}

Logger::Logger(bool log_to_serial, bool log_to_SD,
               bool serial_labels, const String delim):
    log_to_serial_(log_to_serial),
    log_to_SD_(log_to_SD),
    serial_labels_(serial_labels),
    delim_(delim) {}

template <typename T>
void Logger::addVar(const char var_name[], const T* var,
                    const int& min_digits, const int& float_precision) {
  vars_[num_vars_++] = Var(var_name, var, min_digits, float_precision);
}

void Logger::begin(const Stream* serial, const int& pin_select_SD) {
  stream_ = serial;

  if (log_to_SD_) {
    pinMode(pin_select_SD, OUTPUT);

    if (!SD.begin(pin_select_SD)) {
      return;
    }

    makeFile();
  }
}

void Logger::update() {
  if ((!log_to_serial_ && !log_to_SD_) || num_vars_ == 0) {
    return;
```

```
  }
  unsigned long time_now = millis();
  String line, line_with_labels;

  const bool need_line_without_labels = log_to_SD_ ||
  (log_to_serial_ && !serial_labels_);
  const bool need_line_with_labels = log_to_serial_ && serial_labels_;

  for (int i = 0; i < num_vars_; i++) {
    String word = vars_[i].serialize();
    if (i != num_vars_ - 1) {
      word += delim_;
    }
    if (need_line_without_labels) {
      line += word;
    }
    if (need_line_with_labels) {
      line_with_labels += vars_[i].label() + ": " + word;
    }
  }

  if (log_to_serial_) {
    stream_->println(serial_labels_ ? line_with_labels : line);
  }

  if (log_to_SD_) {
    if (!file_) {
      file_ = SD.open(filename_, FILE_WRITE);
    }
    if (file_) {
      file_.println(line);
    }
    if (time_now - last_save_ > kSavePeriod) {
      file_.close();
      last_save_ = time_now;
    }
  }
}

void Logger::makeFile() {
  int num;
  File number_file = SD.open("number.txt", FILE_READ);
  if (number_file) {
```

```
    num = number_file.parseInt();
    number_file.close();
  }

  SD.remove("number.txt");
  number_file = SD.open("number.txt", FILE_WRITE);
  if (number_file) {
    number_file.println(num + 1);
    number_file.close();
  }
  snprintf(filename_, sizeof(filename_), "DATA%03d.TXT", num);

  file_ = SD.open(filename_, FILE_WRITE);
  if (file_) {
    String line;
    for (int i = 0; i < num_vars_; i++) {
      line += vars_[i].label();
      if (i != num_vars_ - 1) {
        line += delim_;
      }
    }
    file_.println(line);
    file_.close();
    last_save_ = millis();
  }
}


}

#ifndef Logging_h
#define Logging_h

#include "Arduino.h"
#include <SD.h>
#include <SPI.h>


namespace logging {

class Var {
public:
  Var() = default;
```

```
  template <typename T>
  Var(const String& label, T* var, const int& min_digits,
  const int& float_precision);

  inline String label() const { return label_; }

  String serialize() const;

private:
  String label_;
  int min_digits_;
  int float_precision_;

  union {
    bool* b;
    int* i;
    float* f;
    double* d;
  } var_;

  enum Type {
    BOOL,
    INT,
    FLOAT,
    DOUBLE
  } type_;

  String pad(String& s);

  void setPtr(const bool* var);

  void setPtr(const int* var);

  void setPtr(const float* var);

  void setPtr(const double* var);

  String serialize(bool* var) const;

  String serialize(int* var) const;

  String serialize(float* var) const;
```

```
  String serialize(double* var) const;
};

class Logger {

  const unsigned long kSavePeriod = 1 * 1000UL;

  static const int kMaxVars = 20;

public:
  Logger(bool log_to_serial, bool log_to_SD,
         bool serial_labels = true, const String delim = "\t");

  template <typename T>
  void addVar(const char var_name[], const T* var,
              const int& min_digits = 1, const int& float_precision = 2);

  void begin(const Stream* serial, const int& pin_select_SD);

  void update();

private:
  const bool log_to_serial_, log_to_SD_, serial_labels_;
  const String delim_;

  Stream* stream_;
  char filename_[12] = "DATA000.TXT";
  File file_;

  unsigned long last_save_ = 0;
  Var vars_[kMaxVars];
  int num_vars_ = 0;

  void makeFile();
};

#define INSTANTIATE_ADDVAR(vartype) \
  template void Logger::addVar(const char var_name[], const vartype* var, \
                               const int& min_digits, const int& float_precision);
INSTANTIATE_ADDVAR(bool)
INSTANTIATE_ADDVAR(int)
INSTANTIATE_ADDVAR(float)
INSTANTIATE_ADDVAR(double)
```

```
#undef INSTANTIATE_ADDVAR


}

#endif

#ifndef Pressure_h
#define Pressure_h

#include "Arduino.h"

class Pressure {
public:
  Pressure(int pin):
    sense_pin_(pin),
    current_(0.0),
    current_peak_(0.0),
    peak_(0.0),
    plateau_(0.0),
    peep_(0.0) {}

  void read() {
    int V = analogRead(sense_pin_);

    float Pmin = -100.0;
    float Pmax = 100.0;
    float Vmax = 1024;
    float pres = (10 * V/Vmax - 1) * (Pmax-Pmin)/8. + Pmin;

    pres *= 1.01972;
    current_peak_ = max(current_peak_, pres);
    current_ = pres;
  }

  const float& get() {
    return current_;
  }

  void set_peak_and_reset() {
    peak_ = current_peak_;
    current_peak_ = 0.0;
  }
```

```
  void set_plateau() {
    plateau_ = get();
  }

  void set_peep() {
    peep_ = get();
  }

  const float& peak() { return peak_; }
  const float& plateau() { return plateau_; }
  const float& peep() { return peep_; }

private:
  int sense_pin_;
  float current_;
  float current_peak_;
  float peak_, plateau_, peep_;
};

#endif

#include "Utilities.h"

namespace utils {

Pulse::Pulse(const unsigned long& period, const float& duty,
const bool& random_offset):
    period_(period),
    on_duration_(duty * period),
    offset_(random_offset ? random(period) : 0) {}

bool Pulse::read() {
  return (millis() - offset_) % period_ < on_duration_;
}

float map(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

float ticks2volume(const float& vol_ticks) {
  return COEFFS.a * sqr(vol_ticks) + COEFFS.b * vol_ticks + COEFFS.c;
}

float volume2ticks(const float& vol_ml) {
```

```
  return (-COEFFS.b + sqrt(sqr(COEFFS.b) - 4 * COEFFS.a *
  (COEFFS.c - vol_ml))) / (2 * COEFFS.a);
}

float readVolume() {
  return map(analogRead(VOL_PIN), 0, ANALOG_PIN_MAX, VOL_MIN, VOL_MAX);
}

float readBpm() {
  return map(analogRead(BPM_PIN), 0, ANALOG_PIN_MAX, BPM_MIN, BPM_MAX);
}

float readIeRatio() {
  return map(analogRead(IE_PIN), 0, ANALOG_PIN_MAX, IE_MIN, IE_MAX);
}

float readAc() {
  return map(analogRead(AC_PIN), 0, ANALOG_PIN_MAX, AC_MIN - AC_RES, AC_MAX);
}

bool readEncoder(const RoboClaw& roboclaw, int& motorPosition) {
  uint8_t robot_status;
  bool valid;
  motorPosition = roboclaw.ReadEncM1(ROBOCLAW_ADDR, &robot_status, &valid);
  return valid;
}

void goToPosition(const RoboClaw& roboclaw, const long& pos, const long& vel,
const long& acc) {
    roboclaw.SpeedAccelDeccelPositionM1(ROBOCLAW_ADDR, acc, vel, acc, pos, 1);
}

void goToPositionByDur(const RoboClaw& roboclaw, const long& goal_pos,
const long& cur_pos, const float& dur) {
  if (dur <= 0) return;

  const long dist = abs(goal_pos - cur_pos);
  long vel = round(2*dist/dur);
  long acc = round(2*vel/dur);
  if (vel > VEL_MAX) {
    vel = VEL_MAX;
    const float acc_dur = dur - dist/vel;
    acc = acc_dur > 0 ? round(vel/acc_dur) : ACC_MAX;
```

```
    acc = min(ACC_MAX, acc);
  }

  goToPosition(roboclaw, goal_pos, vel, acc);
}

bool readMotorCurrent(const RoboClaw& roboclaw, int& motorCurrent) {
  int noSecondMotor;
  const bool valid = roboclaw.ReadCurrents(ROBOCLAW_ADDR, motorCurrent, noSecondMotor);
  return valid;
}



}

#ifndef Utilities_h
#define Utilities_h
#include "Arduino.h"
#include "src/thirdparty/RoboClaw/RoboClaw.h"
#include "cpp_utils.h"
#include "Constants.h"
namespace utils {

class Pulse {
public:
  Pulse(const unsigned long& period, const float& duty,
  const bool& random_offset = true);

  bool read();

private:
  const unsigned long period_;
  const unsigned long on_duration_;
  const unsigned long offset_;
};

float map(float x, float in_min, float in_max, float out_min, float out_max);

float ticks2volume(const float& vol_ticks);

float volume2ticks(const float& vol_ml);

inline float now() { return millis()*1e-3; }
```

```
inline bool homeSwitchPressed() { return digitalRead(HOME_PIN) == LOW; }

float readVolume();
float readBpm();
float readIeRatio();
float readAc();

bool readEncoder(const RoboClaw& roboclaw, int& motorPosition);

void goToPosition(const RoboClaw& roboclaw, const long& pos,
const long& vel, const long& acc);

void goToPositionByDur(const RoboClaw& roboclaw, const long& goal_pos,
const long& cur_pos, const float& dur);

bool readMotorCurrent(const RoboClaw& roboclaw, int& motorCurrent);


}


#endif


#ifndef CPP_UTILS_H_INCLUDED
#define CPP_UTILS_H_INCLUDED
#ifdef __cplusplus
#undef min
#undef max
#undef abs
#undef radians
#undef degrees
#undef sq

template<typename Tpa, typename Tpb>
inline auto min(const Tpa& a, const Tpb& b) -> decltype(a < b ? a : b) {
    return b < a ? b : a;
}

template<typename Tpa, typename Tpb>
inline auto max(const Tpa& a, const Tpb& b) -> decltype(b > a ? b : a) {
    return b > a ? b : a;
}
```

```
inline double abs(double x) {
    return __builtin_fabs(x);
}

inline float abs(float x) {
    return __builtin_fabsf(x);
}

inline long double abs(long double x) {
    return __builtin_fabsl(x);
}

inline long abs(long i) {
    return labs(i);
}

inline long long abs(long long x) {
    return x >= 0 ? x : -x;
}

inline float radians(float deg) {
    return deg * DEG_TO_RAD;
}

inline double radians(double deg) {
    return deg * DEG_TO_RAD;
}

inline float degrees(float deg) {
    return deg * RAD_TO_DEG;
}

inline double degrees(double deg) {
    return deg * RAD_TO_DEG;
}

template<typename Tp>
inline auto sqr(const Tp& x) -> decltype(x * x) {
    return x * x;
}

inline void * operator new(size_t size) { return malloc(size); }
inline void   operator delete(void* ptr) { free(ptr); }
```

```
#endif

#endif

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
```

```
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
```

```
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

ANEXO B

Diagramas Esquemáticos y PCB del circuito de control

B.1. Diagrama del circuito

El diagrama del circuito, como ya se ha mencionado en secciones anteriores, tiene como base un Arduino Mega, al cual se encuentran conectados todos los demás dispositivos electrónicos y su respectiva fuente de poder, el Arduino tiene conectados a los potenciómetros, botones, luces, alarma, switch de fin carrera, display, sensor, encoder y un slot micro SD. Todos los componentes se encuentran conectados como se muestra en la Figura B.1.

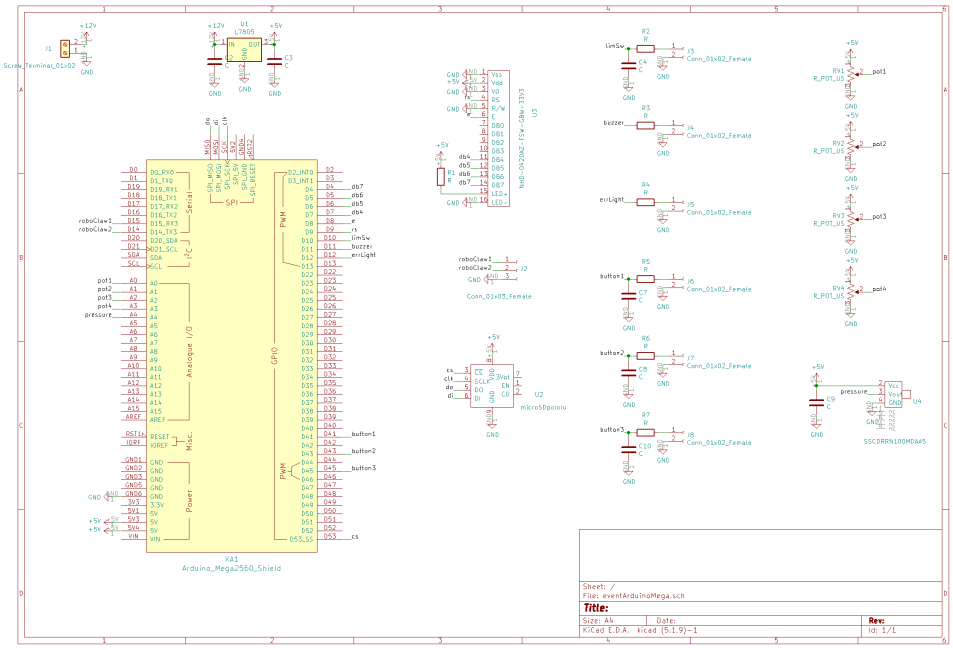


Figura B.1: Circuito de control esquemático del ventilador con base en un Arduino Mega.

B.2. Placa PCB del control

Partiendo del circuito esquemático, mostrado en la Figura B.1, se ha diseñado el respectivo PCB, teniendo particularmente en cuenta el tamaño del dispositivo, ya que al ser usado en situaciones en donde se requiere movilidad, el tamaño del mismo deberá ser lo más pequeño y robusto posible. El PCB del circuito puede observarse en la Figura B.2.

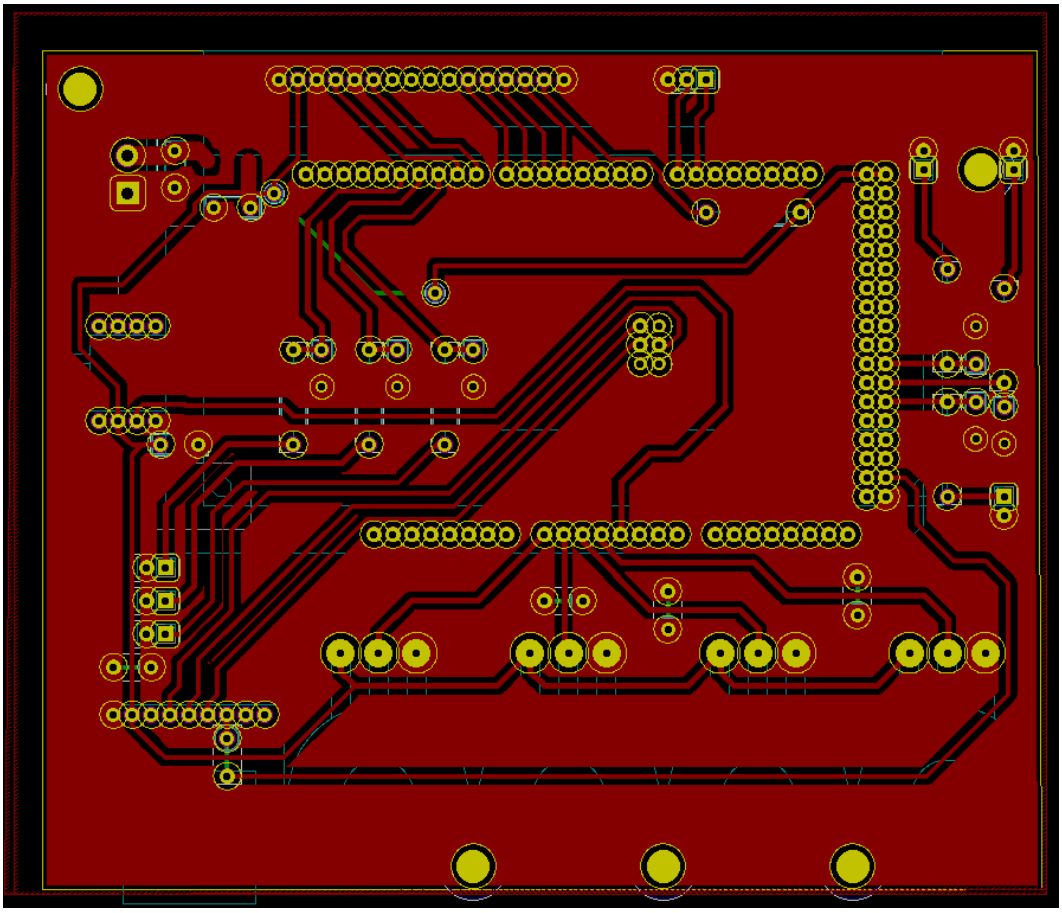


Figura B.2: Circuito de control PCB del ventilador con base en un Arduino Mega.

Anexo C

Especificaciones clave de ventilación

C.1. Especificaciones clave de ventilación

Esta página resume el conjunto mínimo de parámetros controlables y rangos recomendados, para ventilar a un paciente. Estos son más agresivos que la Guía de diseño usada como modelo. Esta guía es del ventilador de emergencia (EUV) de AAMI/CR501:2020. [15]

- **Frecuencia respiratoria (RR) (respiraciones por minuto):** entre 6 y 40. Tenga en cuenta que los RR bajos de 6 a 9 solo se aplican al control de asistencia.
- **Volumen corriente (TV) (volumen de aire empujado hacia el pulmón):** entre 200 y 800 ml según el peso del paciente.
- **Relación I/E (relación tiempo inspiratorio/espiración):** se recomienda comenzar alrededor de 1:2; mejor si se puede ajustar entre un rango de 1:1 - 1:4.
- **El control de asistencia se basa en una sensibilidad de activación:** cuando un paciente intenta inspirar, puede provocar una caída del orden de 2 a 7 H_2O, con respecto a la presión PEEP (no necesariamente igual a la presión atmosférica).
- La presión de las vías respiratorias debe controlarse continuamente. La presión máxima debe limitarse a 40 cm H_2O en cualquier momento.
- Presión Plateau debe limitarse a un máximo de 30 cm H_2O.
- El uso de una válvula mecánica de expulsión pasiva fijada en 40 cm H_2O es muy recomendable. Está integrado en la mayoría de los reanimadores manuales.
- El médico requiere lecturas de presión Plateau y PEEP.
- Se requiere una PEEP de 5 a 15 cm H_2O; muchos pacientes necesitan 10-15 cm H_2O.
- Las condiciones de falla deben generar una alarma y permitir la intervención manual del médico de manera inmediata

El rango se determina en función de la configuración del ventilador de varios pacientes con COVID-19 informada por las UCI del área de Boston, Estados Unidos. Este es un requisito mínimo establecido para uso de ventiladores de emergencia. Los equipos diseñados para un uso más regular, incluso si

están destinados a mercados emergentes, requerirán funciones adicionales para su uso regular.

La ventilación del aire de la habitación es mejor que la ausencia total de ventilación. La mezcla de oxígeno, aire y gas para ajustar la FiO_2 (Fracción Inspirada de Oxígeno) no es importante en un escenario de emergencia. Sin duda, es bueno tener esa capacidad y se puede implementar fácilmente con un mezclador de gas de oxígeno/aire que algunos hospitales ya tienen.

COVID-19 puede ser transmitido de manera aérea, por lo que se requiere filtración HEPA en la exhalación del paciente o entre la unidad de ventilación y el paciente (al final del tubo endotraqueal) para proteger al personal clínico de ciertas infecciones. Esto puede servir para proporcionar un intercambio de calor y humedad entre el aire exhalado y el inspirado. Los filtros HEPA en línea generalmente se pueden comprar junto con las bolsas de reanimación manual.

ANEXO D

Estudios realizados por el personal del MIT

D.1. Introducción

Este anexo tiene como objetivo mostrar los análisis realizados por el personal del MIT, como ya se ha descrito en anterioridad, ciertas pruebas de funcionamiento se salen de la experticie de los autores, sin embargo, adjuntar un análisis adicional sobre simulaciones y mediciones del funcionamiento del ventilador pudiesen ser de provecho para el lector.

Los resultados de estos análisis vienen dados a manera de formas de onda desde aproximaciones basadas en modelos de simulación y mediante la recolección de datos usando un simulador de respiración ASL 5000 conectado al ventilador.

D.1.1. ISO 80601-2-79:2018

Las configuraciones para las pruebas son inspiradas por la ISO 80601-2-79:2018 [30], la Tabla D.1, muestra los parámetros usados a través de este anexo, en donde las ocho primeras filas de la tabla son procedentes directamente de la ISO 80601-2-79:2018.

De particular interés es el entendimiento de:

- La forma de la señal de presión
- La señal de flujo y sus valores pico
- La señal del volumen tidal entregado
- Los parámetros en donde la presión excede los 40 $cm\ H_2O$.

Otros requerimientos generales de la ISO 80601-2-79:2018 para tipos respiración controlada por volumen durante las pruebas son:

1. El volumen entregado en respiraciones individuales no deben desviarse más del 35 %.
2. El promedio de volumen entregado sobre un intervalo de un minuto no debe desviarse de más del 25 %.

Test	Complianza (ml/hPa*) ± 10 %	Resistencia Lineal (hPa*/L/s) ± 10 %	Volumen (ml)	Frecuencia del Ventilador (breaths/min)	Tiempo de Inspiracion (s)	PEEP (hPa*)
1	50	5	500	20	1	5
2	50	20	500	12	1	10
3	20	5	500	20	1	5
4	20	20	500	20	1	10
5	20	20	300	20	1	5
6	20	50	300	12	1	10
7	10	50	300	20	1	10
8	10	20	200	20	1	5
9	10	20	200	25	1	5
10	10	20	200	30	1	5

Tabla D.1: Tabla de parámetros usada para este estudio

D.1.2. Perfiles de flujo

En el reporte de los desarrolladores de la MIT se discuten datos debidos a perfiles de flujo constante y de flujo triangular.

D.1.2.1. Flujo constante

En la ventilación por control de volumen, una aproximación común es la de entregar volumen a flujo fijo durante el tiempo de inspiración, esto requiere un que el tiempo se asemeje a una onda cuadrada lo máximo posible. La presión inspiratoria pico PIP para un flujo constante es esperada al final de la duración de la inhalación, cuando la máxima presión debido a la complianza es agregada a la presión constante debido al flujo constante que fluye a través de la resistencia de las vías respiratorias.

D.1.2.2. Flujo triangular

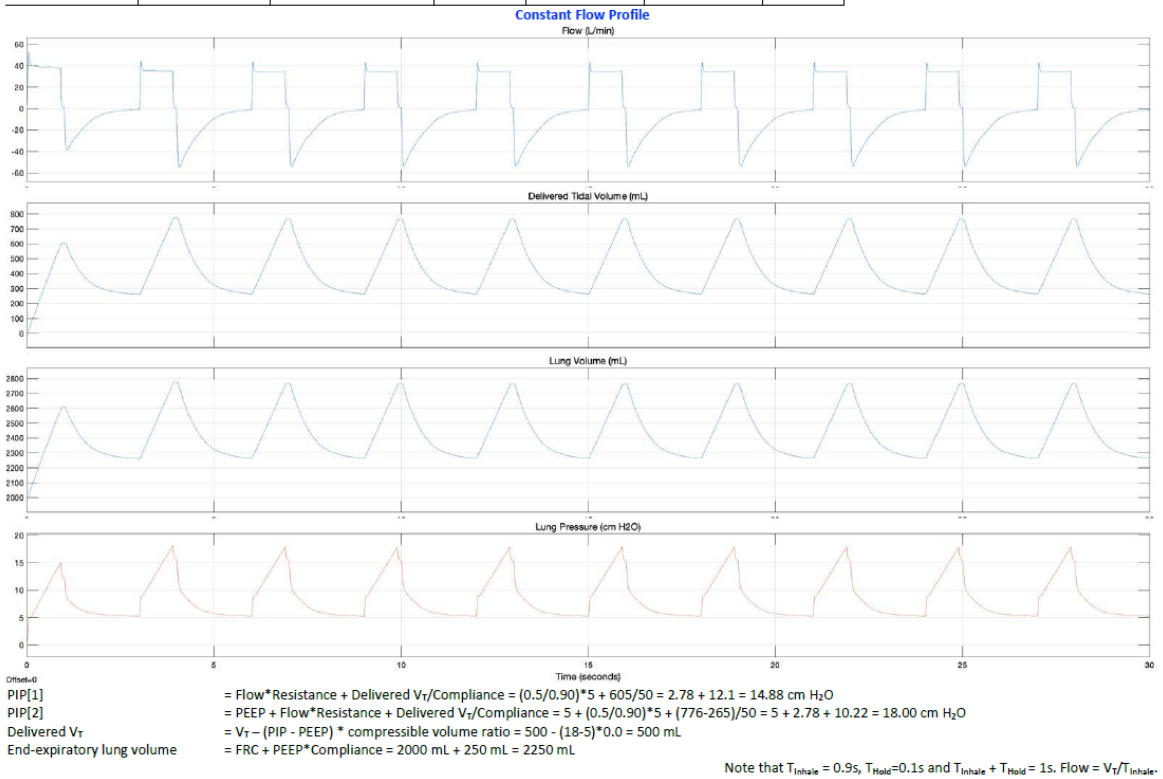
Un flujo triangular que cubre la misma area o flujo tidal entregado, como un flujo constante deberá requerir un flujo pico que sea el doble del de un perfil de flujo constante. En casos en donde pequeñas subidas y bajadas en tiempo, para conseguir una forma de onda constante, no puedan ser logrados, un flujo triangular es posible.

D.1.3. Formas de onda basados en modelo para pruebas realizadas por el personal de la MIT

Se hace uso de un modelo para simular el comportamiento, acorde a estas simulaciones basadas en modelos, la cuarta y séptima prueba exceden los 40 $cm\ H_2O$, cuando se usa un flujo constante, mientras que las pruebas cuarta, sexta y séptima exceden los 40 $cm\ H_2O$ cuando se usa un perfil de forma de onda triangular, además se provee cálculos preliminares para caracterizar las respuestas observadas basadas en modelo para el caso de flujo constante.

El modelo en Simulink ya ha sido mostrado en la Figura **??**. Todas las figuras que serán mostradas a continuación son propiedad de la MIT [6].

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H2O)
1	50	5	500	20	1 I/E =1:2	5



Offset=0

PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.5/0.90)*5 + 605/50 = 2.78 + 12.1 = 14.88 cm H_2O

PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.5/0.90)*5 + (776-265)/50 = 5 + 2.78 + 10.22 = 18.00 cm H_2O

Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 500 - (18-5)*0.0 = 500 mL

End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 250 mL = 2250 mL

Note that T_{Inhale} = 0.9s, T_{Hold}=0.1s and T_{Inhale} + T_{Hold} = 1s. Flow = V_T/T_{Inhale}.

Figura D.1: Resultado del experimento #1 en Simulink realizado por la MIT (1).

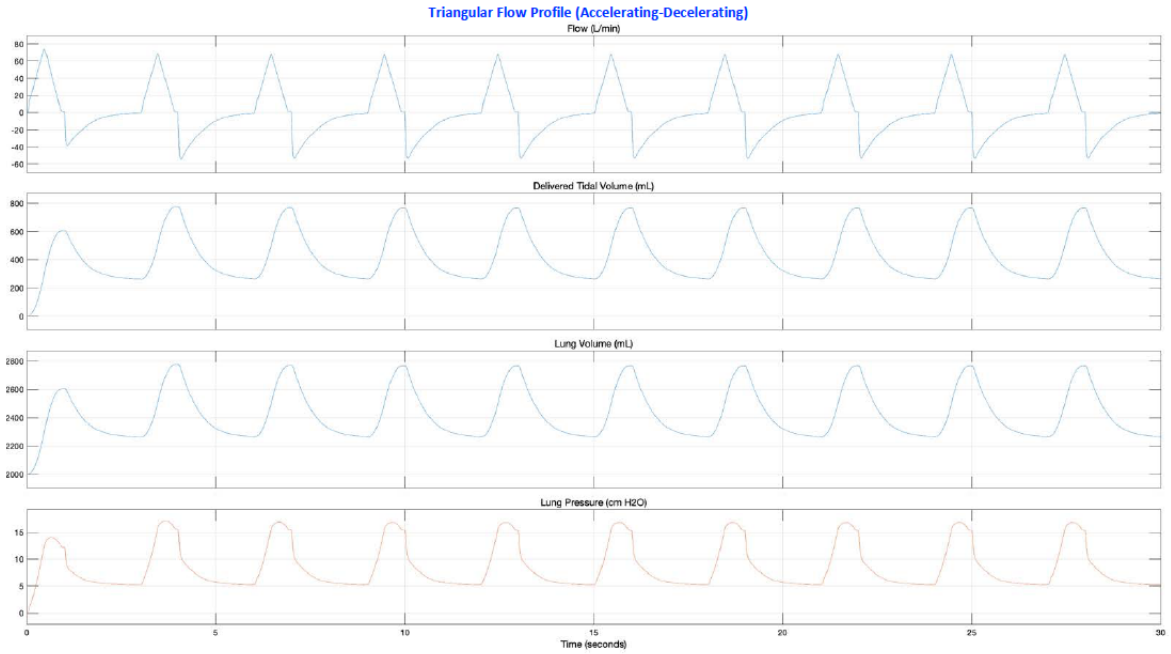
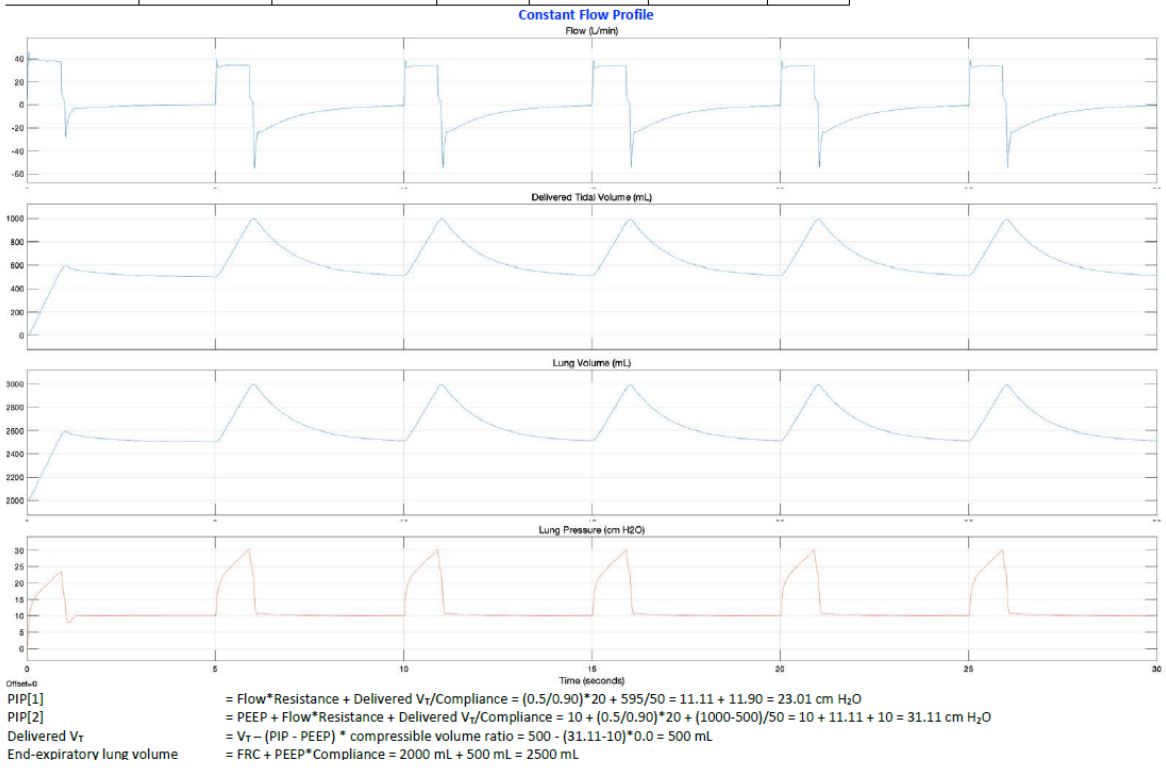


Figura D.2: Resultado del experimento #1 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
2	50	20	500	12	1 I/E =1:4	10



PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.5/0.90)*20 + 595/50 = 11.11 + 11.90 = 23.01 cm H_2O

PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 10 + (0.5/0.90)*20 + (1000-500)/50 = 10 + 11.11 + 10 = 31.11 cm H_2O

Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 500 - (31.11-10)*0.0 = 500 mL

End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 500 mL = 2500 mL

Figura D.3: Resultado del experimento #2 en Simulink realizado por la MIT (1).

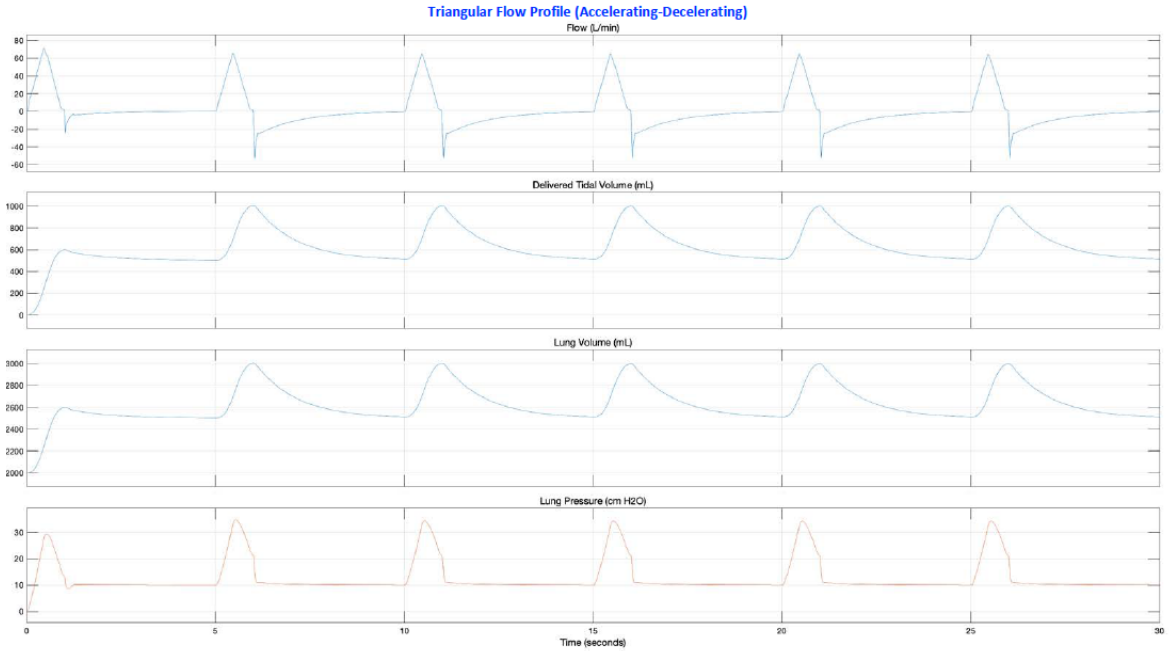
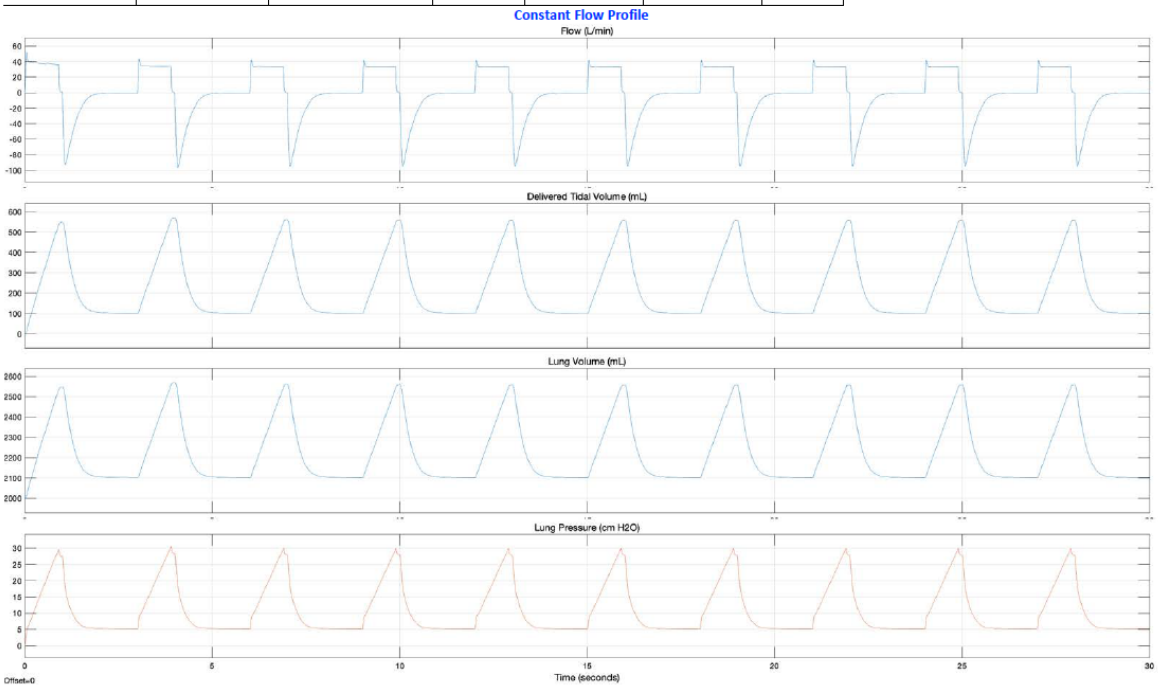


Figura D.4: Resultado del experimento #2 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
3	20	5	500	20	1 I/E =1:2	5



PIP[1]	= Flow*Resistance + Delivered V_T/Compliance = (0.5/0.90)*5 + 548/20 = 2.78 + 27.4 = 30.18 cm H_2O
PIP[2]	= PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.5/0.90)*5 + (569-100)/20 = 5 + 2.78 + 23.45 = 31.23 cm H_2O
Delivered V_T	= V_T – (PIP - PEEP) * compressible volume ratio = 500 - (31.23-5)*1.154 = 469.73 mL
End-expiratory lung volume	= FRC + PEEP*Compliance = 2000 mL + 100 mL = 2100 mL

Figura D.5: Resultado del experimento #3 en Simulink realizado por la MIT (1).

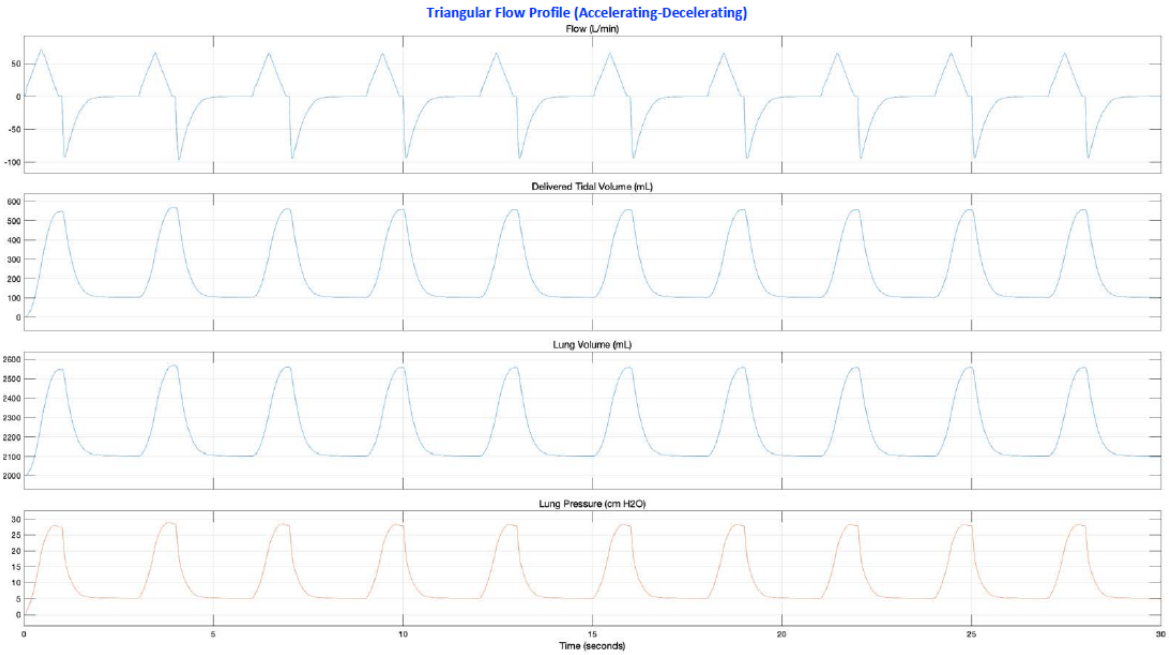
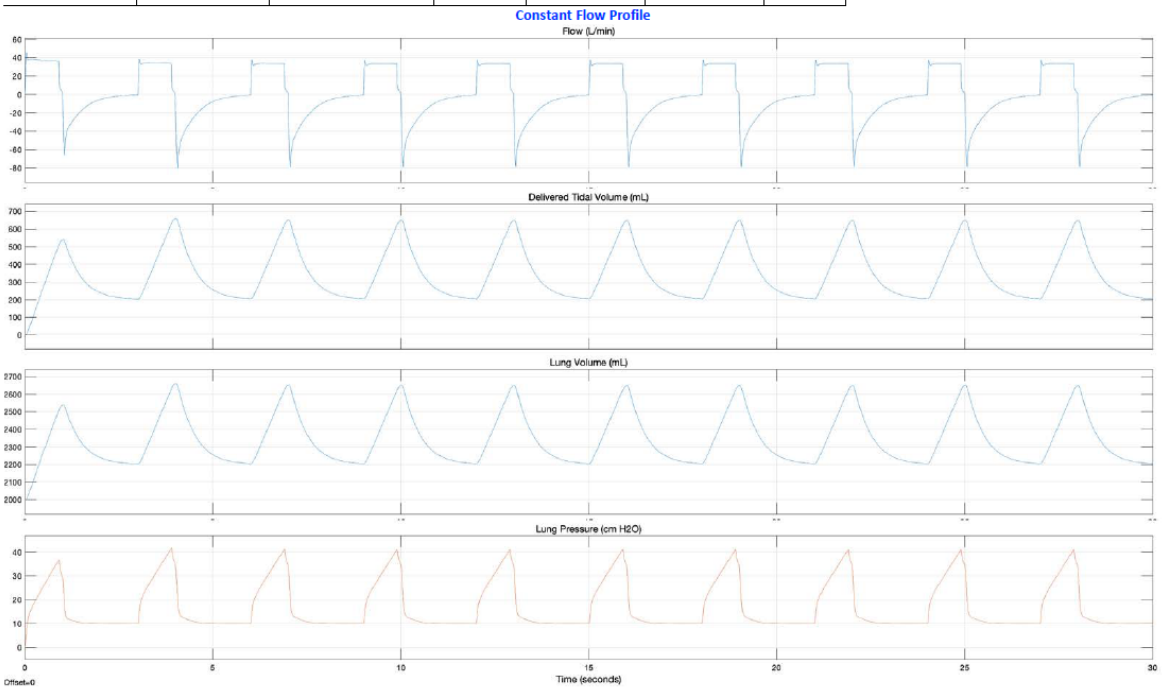


Figura D.6: Resultado del experimento #3 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
4	20	20	500	20	1 I/E =1:2	10



PIP[1]	= Flow*Resistance + Delivered V_T/Compliance = (0.5/0.90)*20 + 540/20 = 11.11 + 27 = 38.11 cm H_2O
PIP[2]	= PEEP + Flow*Resistance + Delivered V_T/Compliance = 10 + (0.5/0.90)*20 + (660-200)/20 = 10 + 11.11 + 23 = 44.11 cm H_2O
Delivered V_T	= V_T – (PIP - PEEP) * compressible volume ratio = 500 - (44.11-10)*1.154 = 460.64 mL
End-expiratory lung volume	= FRC + PEEP*Compliance = 2000 mL + 200 mL = 2200 mL

Figura D.7: Resultado del experimento #4 en Simulink realizado por la MIT (1).

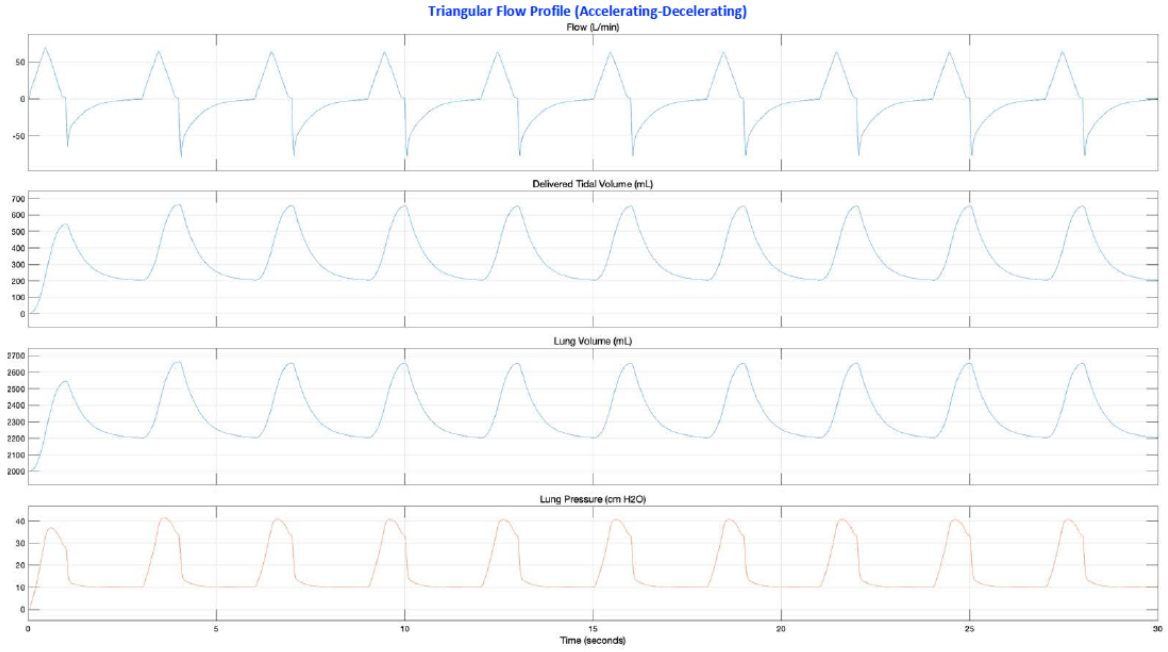
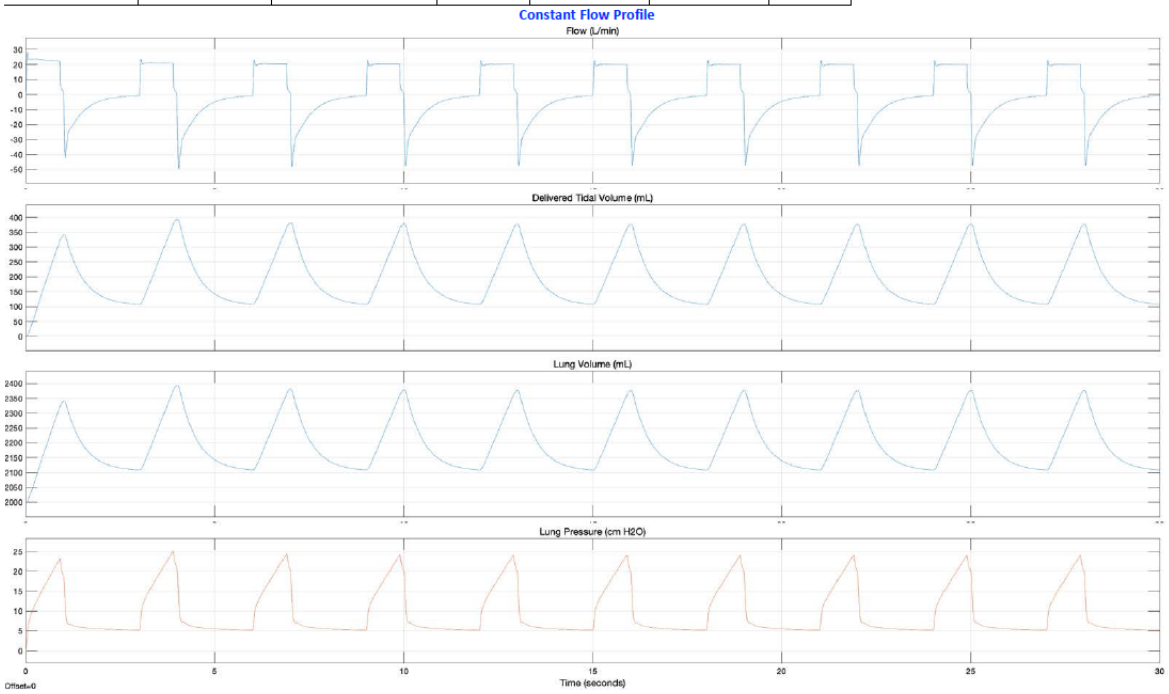


Figura D.8: Resultado del experimento #4 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
5	20	20	300	20	1 I/E =1:2	5



PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.3/0.90)*20 + 340/20 = 6.67 + 17 = 23.67 cm H_2O

PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.3/0.90)*20 + (393-108)/20 = 5 + 6.67 + 14.25 = 25.92 cm H_2O

Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 300 - (25.92-5)*1.154 = 275.86 mL

End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 100 mL = 2100 mL

Figura D.9: Resultado del experimento #5 en Simulink realizado por la MIT (1).

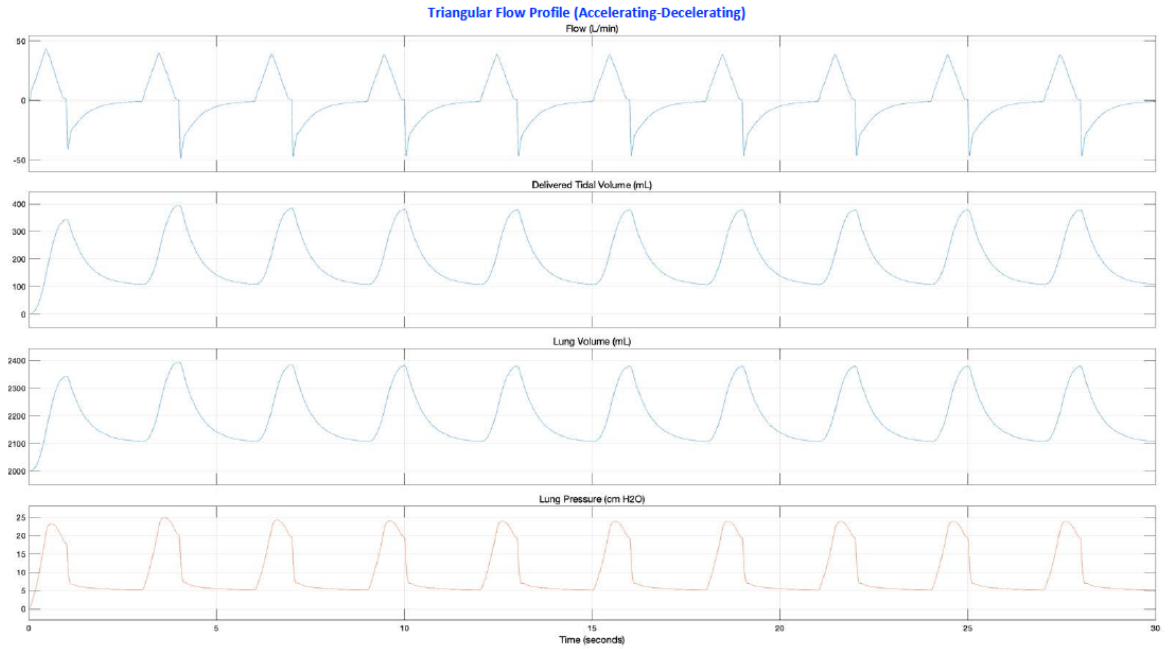
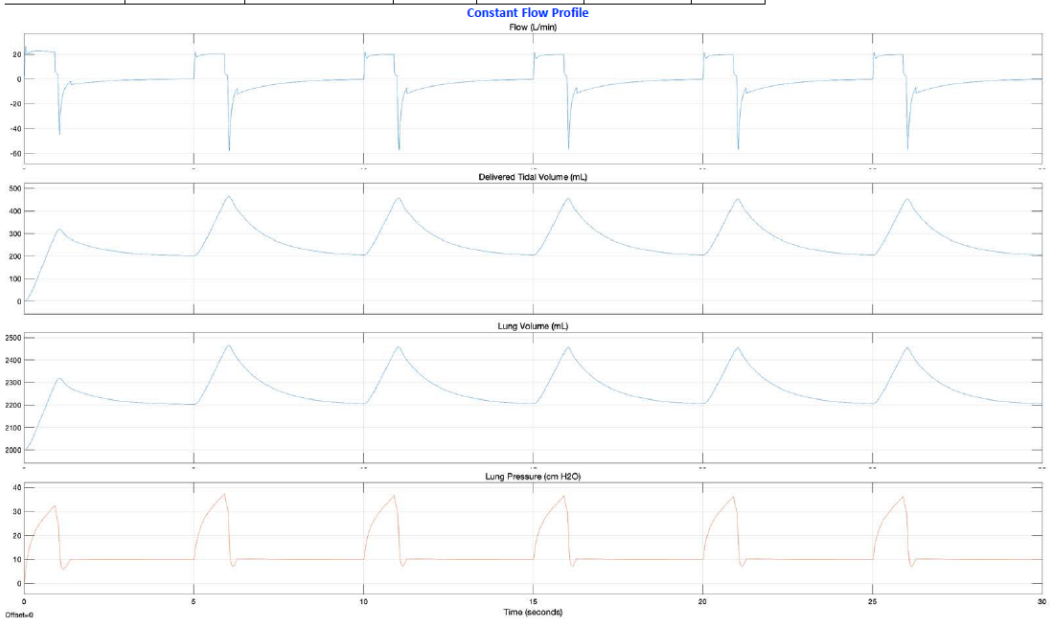


Figura D.10: Resultado del experimento #5 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
6	20	50	300	12	1 I/E =1:4	10



PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.3/0.90)*50 + 320/20 = 16.67 + 16 = 32.67 cm H_2O
PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 10 + (0.3/0.90)*50 + (465-202)/20 = 10 + 16.67 + 13.15 = 39.82 cm H_2O
Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 300 - (39.82 -10)*1.154 = 265.59 mL
End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 200 mL = 2200 mL

Figura D.11: Resultado del experimento #6 en Simulink realizado por la MIT (1).

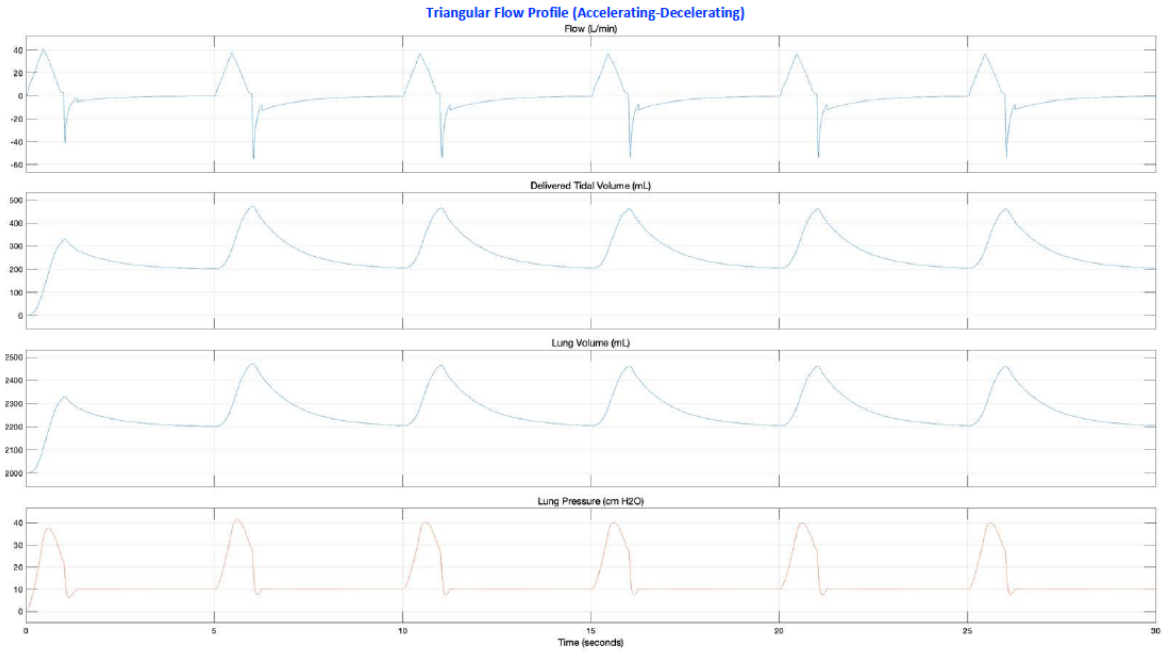
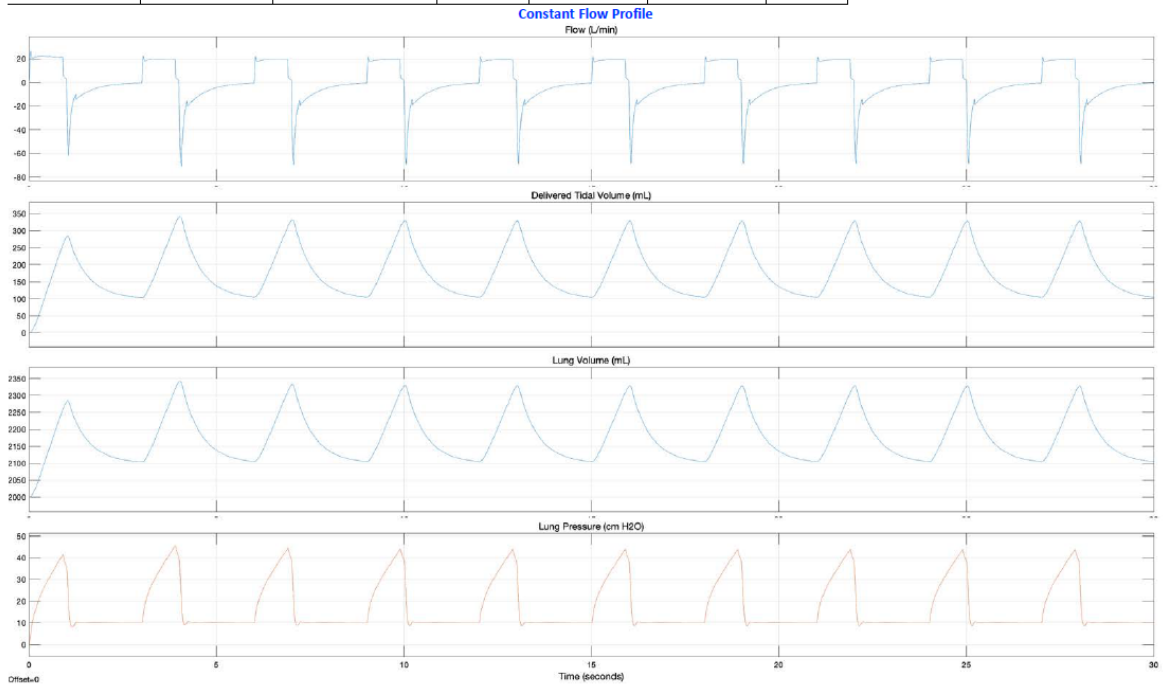


Figura D.12: Resultado del experimento #6 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
7	10	50	300	20	1 I/E =1:2	10



Offset=0

PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.3/0.90)*50 + 284/10 = 16.67 + 28.4 = 45.06 cm H_2O

PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 10 + (0.3/0.90)*50 + (341-104)/10 = 10 + 16.67 + 23.7 = 50.37 cm H_2O

Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 300 - (50.37 -10)*1.154 = 253.41 mL

End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 100 mL = 2100 mL

Figura D.13: Resultado del experimento #7 en Simulink realizado por la MIT (1).

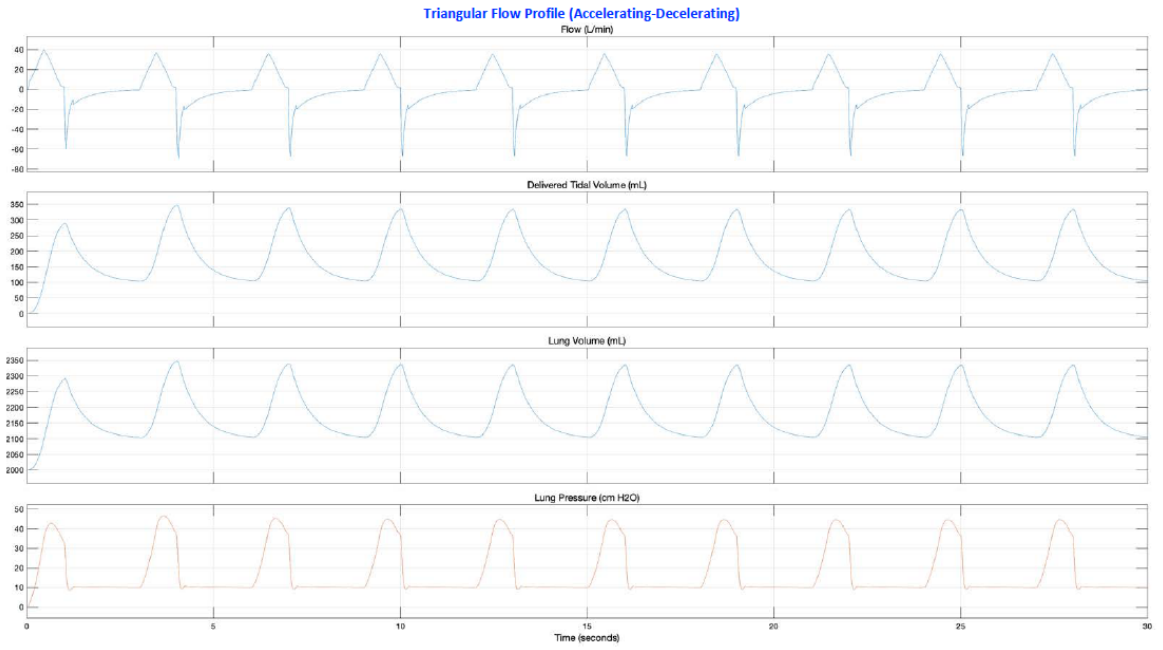
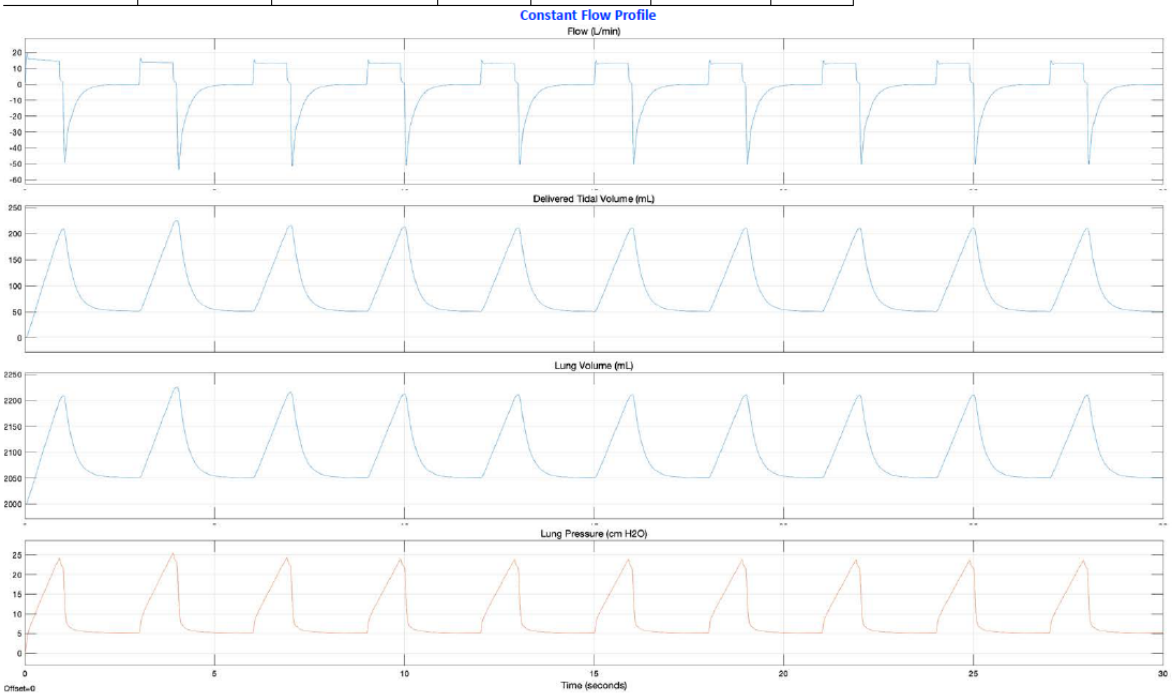


Figura D.14: Resultado del experimento #7 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
8	10	20	200	20	1 I/E =1:2	5



PIP[1]	= Flow*Resistance + Delivered V_T/Compliance = (0.2/0.90)*20 + 209/10 = 4 .44+ 20.9 = 25.34 cm H_2O
PIP[2]	= PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.2/0.90)*20 + (226-51)/10 = 5 + 4 .44 + 17.5 = 26.94 cm H_2O
Delivered V_T	= V_T – (PIP - PEEP) * compressible volume ratio = 200 - (26.94 -5)*1.154 = 174.68 mL
End-expiratory lung volume	= FRC + PEEP*Compliance = 2000 mL + 50 mL = 2050 mL

Figura D.15: Resultado del experimento #8 en Simulink realizado por la MIT (1).

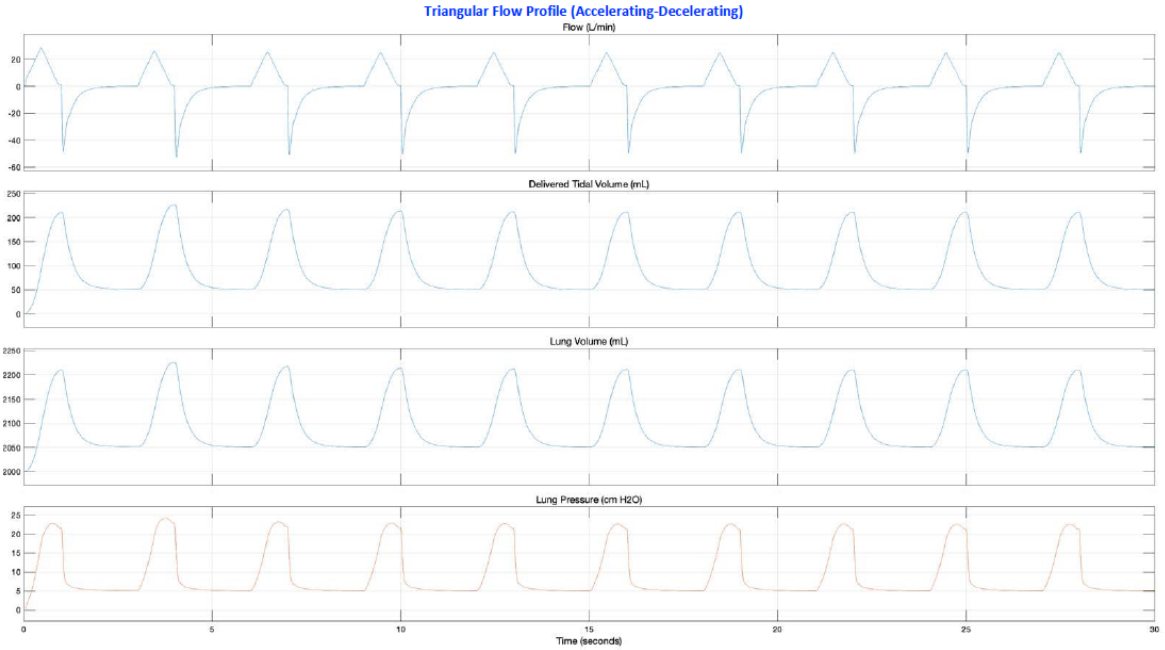
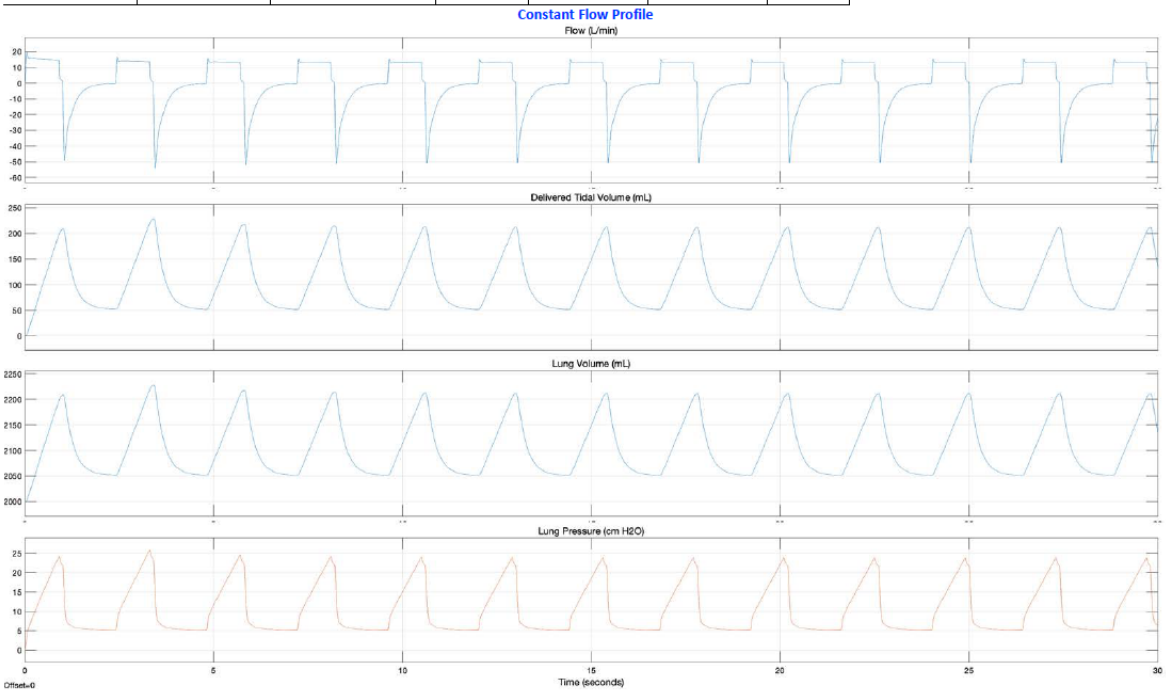


Figura D.16: Resultado del experimento #8 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
9	10	20	200	25	1 I/E =1:1.4	5



PIP[1] = Flow*Resistance + Delivered V_T/Compliance = (0.2/0.90)*20 + 209/10 = 4 .44+ 20.9 = 25.34 cm H_2O

PIP[2] = PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.2/0.90)*20 + (228-52)/10 = 5 + 4 .44 + 17.6 = 27.04 cm H_2O

Delivered V_T = V_T – (PIP - PEEP) * compressible volume ratio = 200 - (27.04 -5)*1.154 = 174.57 mL

End-expiratory lung volume = FRC + PEEP*Compliance = 2000 mL + 50 mL = 2050 mL

Figura D.17: Resultado del experimento #9 en Simulink realizado por la MIT (1).

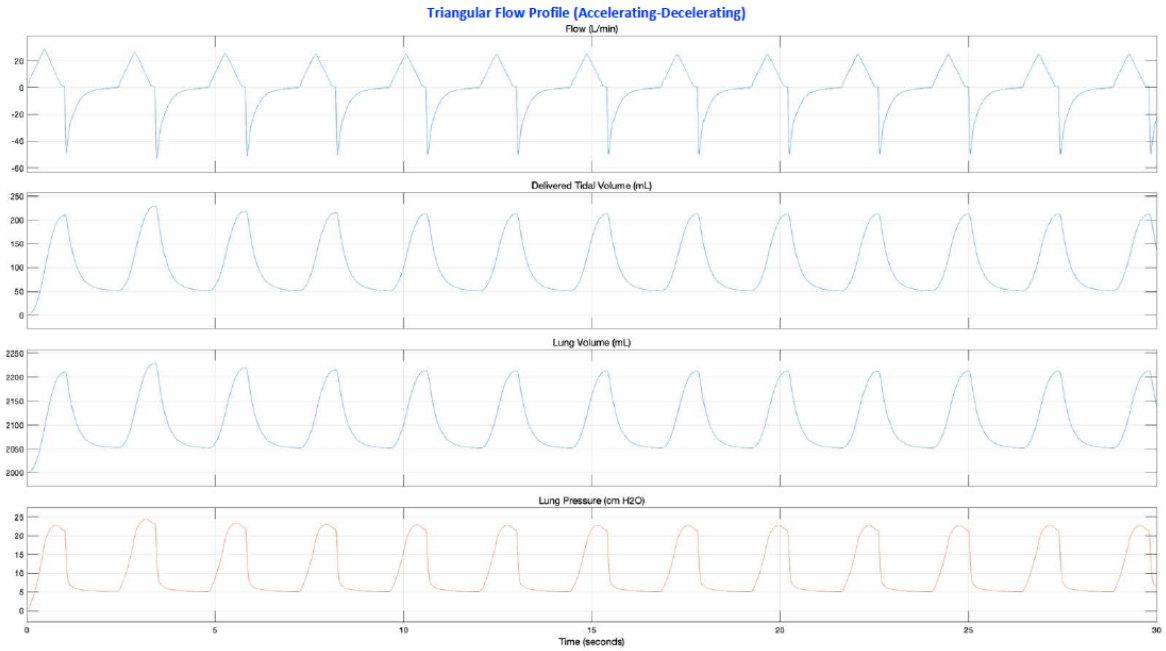
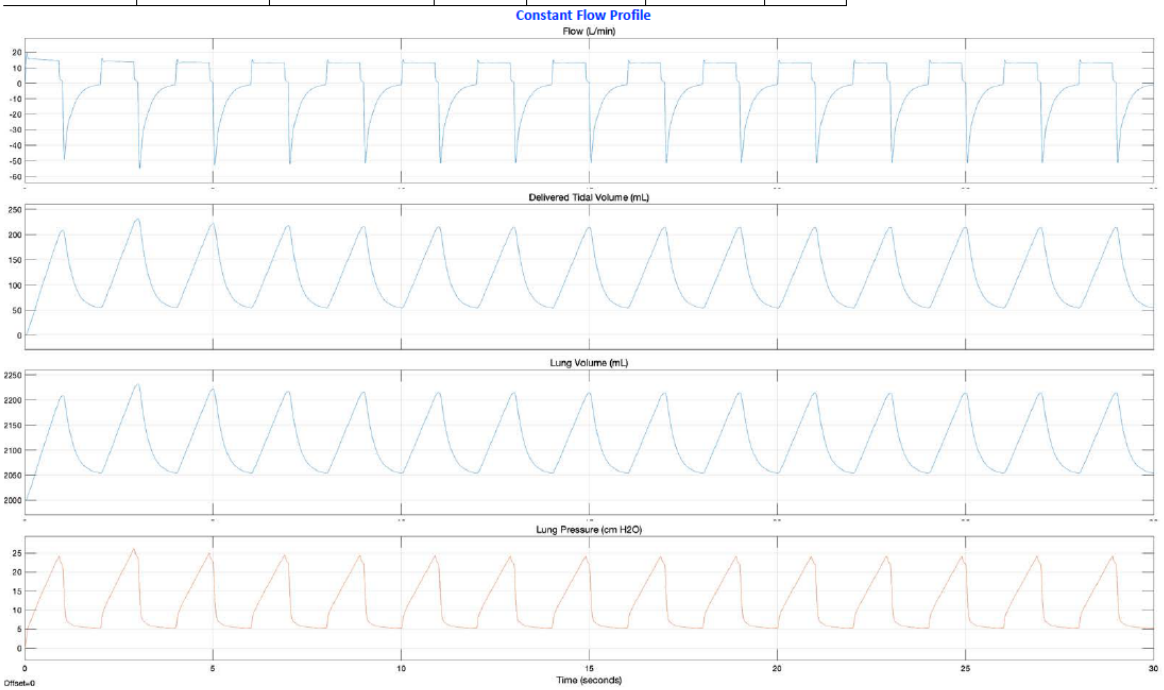


Figura D.18: Resultado del experimento #9 en Simulink realizado por la MIT (2).

Test Num.	Compliance (ml/cm H_2O)	Linear Resistance (cm H_2O/(L/s))	Volume (ml)	Ventilator Frequency[1] (breaths/min)	Inspiratory Time (s)	PEEP (cm H_2O)
10	10	20	200	30	1 I/E =1:1	5



PIP[1]	= Flow*Resistance + Delivered V_T/Compliance = (0.2/0.90)*20 + 209/10 = 4 .44+ 20.9 = 25.34 cm H_2O
PIP[2]	= PEEP + Flow*Resistance + Delivered V_T/Compliance = 5 + (0.2/0.90)*20 + (232-54)/10 = 5 + 4 .44 + 17.8 = 27.24 cm H_2O
Delivered V_T	= V_T – (PIP - PEEP) * compressible volume ratio = 200 - (27.24 -5)*1.154 = 174.34 mL
End-expiratory lung volume	= FRC + PEEP*Compliance = 2000 mL + 50 mL = 2050 mL

Figura D.19: Resultado del experimento #10 en Simulink realizado por la MIT (1).

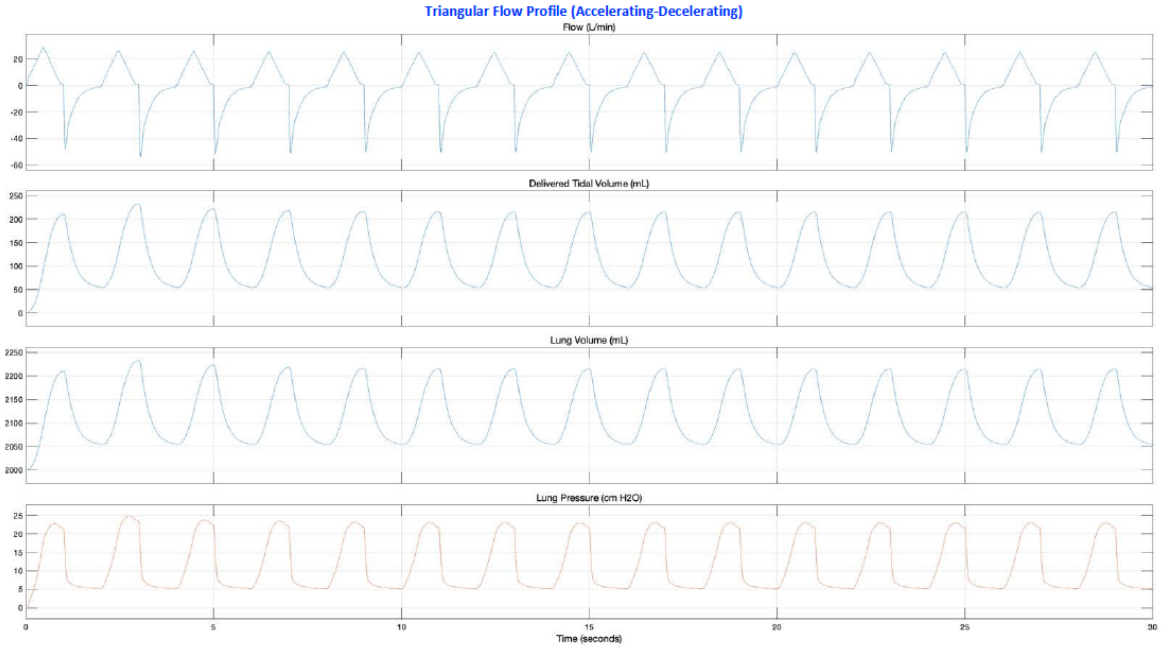


Figura D.20: Resultado del experimento #10 en Simulink realizado por la MIT (2).

ANEXO E

Listado y precio de los materiales

E.1. Listado de materiales electrónicos y su precio

En la Tabla E.1, se observa el listado de componentes electrónicos usados para la construcción del prototipo.

Nombre del componente	Fuente	Numero del componente	Cantidad	Precio	Total
PG188 1/2 Hex Gearmotor	AndyMark	am-3656	1	94	94
Arduino Mega 2560	Audio Electro Center	-	1	16,24	16,24
Pantalla LCD 20x4	Audio Electro Center	-	1	11,19	11,19
Buzzer	Audio Electro Center	-	1	0,67	0,67
RoboClaw Solo 30A	Pololu	3290	1	79,95	79,95
Modulo lector Micro SD	Audio Electro Center	-	1	3,48	3,48
Indicador LED con carcasa	Audio Electro Center	-	1	0,25	0,25
Sensor de Presión Honeywell	Digikey	SSCDRRN100MDAA5	1	35,74	35,74
Switch de Poder	Audio Electro Center	-	1	0,31	0,31
Fuente de poder 12V 150W	Digikey	LRS-150-12	1	75	75
Botones	Audio Electro Center	-	3	0,31	0,93
Boton de emergencia	Audio Electro Center	-	1	5	5
Cables varios	Audio Electro Center	-	1	10	10
Resistencia 180 Ohms 1/2 W	Audio Electro Center	-	1	0,1	0,1
Regulador de voltaje 7805	Digikey	-	1	0,89	0,89
Capacitor 1uF	Audio Electro Center	-	7	0,14	0,98
Potenciometros 10 KOhms	Audio Electro Center	-	4	0,45	1,8
Resistencia 10KOhms 1/4 W	Audio Electro Center	-	6	0,1	0,6
Diodo 1N5400	Audio Electro Center	-	1	0,09	0,09
				Subtotal	337,22
				IVA 12 %	40,47
				TOTAL	377,69

Tabla E.1: Listado de materiales electrónicos usados en la construcción del prototipo

E.2. Listado de materiales mecánicos y su precio

A continuación se muestra en la Tabla E.2 una lista de materiales del apartado mecánico del prototipo, así como su costo en el mercado local.

Nombre del componente	Fuente	Numero del componente	Cantidad	Costo	Total
Dedos para sujeción de bolsa Ambu	Impresión 3D	-	30	4	120
Brazos de acero inoxidable	Corte laser Smelektronik	-	2	25	50
Perfiles de aluminio 20x80	Perfiles CNC	-	2	15	30
Engranaje	AndyMark	am-4254	1	20	20
Soporte para bolsa Ambu	Impresión 3D	-	2	10	20
Soporte para motor	Impresion 3D	-	1	10	10
Soporte para brazos	Impresión 3D	-	1	8	8
Soporte universal en L	Perfiles CNC	-	20	0,50	10
Pernos allen cabeza plana M5x15	Perfiles CNC	-	20	0,0693	1,79
Rodamiento 6000	Perfiles CNC	-	4	0,8929	3,57
Tuerca T rectangular serie 20	Perfiles CNC	-	20	0,1607	3,21
Acero Transmisión 34	Perfiles CNC	-	50 cm	3,5714	1,79
Soporte de esquina serie 20	Perfiles CNC	-	20	0,8929	17,86
Tuerca hexagonal de seguridad M5	Perfiles CNC	-	0,0893	20	1,79
Perno allen cabeza plana M5x8	Perfiles CNC	-	0,0893	20	1,79
Switch Limite	Electro Audio Center	-	1	3,03	3,03
Collar de seguridad 10 mm	Perfiles CNC	-	4	0,8929	3,57
Acero plata 10mm	Perfiles CNC	-	20 cm	17,857	3,57
Case	Impresión 3D	-	2	50	100
				Subtotal	409,97
				IVA 12 %	49,2
				Total	459,17

Tabla E.2: Listado de materiales mecánicos usados en la construcción del prototipo

Según lo mostrado en las tablas E.1 y E.2, el costo del prototipo construido es de $836,86, que es un precio mucho menor a un ventilador de alta gama del mercado.

ANEXO F

Cálculos de torque y estrés

En este apartado se presentan los formularios usados para el cálculo de torque y estrés.

E-Vent torque and speed calculations

Purpose: This spreadsheet outputs the torque and speed required by the driven arm (in case the arm is driven directly) and the pinion. The radial load from the pinion on the gearbox shaft is also calculated. This also causes bending.

Updated 14 April 2020

Inputs

System physical parameters

pres	40	cm H2O	Pressure in bag
	3922.4	Pa	
area_x	90	mm	contact area dim. X
area_y	115	mm	contact area dim. Y
area	0.01035	m^2	Contact area on bag
lever_arm	120.00	mm	Pressure force is located in center of contact area
	4.724	in	
n_finger	48	teeth	Finger gear - number of teeth
n_pinion	30	teeth	Drive pinion - number of teeth
P	16	TPI	Diametral pitch for gear set
P.A.	14.5	deg	Pressure angle
eta_pres	2		Assumed pressure transmission efficiency
TV_max	800	mL	The maximum tidal volume possible
sweep	27	deg	The angle that one arm must sweep for max tidal volume
	0.471239	rad	

Patient parameters

T_V	600	mL	Tidal Volume
I:E	3		1:X - inspiration:expiration ratio
rate	30	BPM	Breaths per minute

We could also model compliance, which would result in increasing pressure as a function of volume delivered, for simplicity we focus on the end condition, when the arm is completing it's transit to maximum compression.

Outputs

tau_in	4.87	Nm	Input torque per finger required to drive bag
tau_finger	9.74	Nm	Driving torque on finger
ratio	1.6		Gear ratio finger/pinion
tau_total	19.49	Nm	Torque applied to the driven arm (individual arm x 2)
d_finger	3	in	Finger gear pitch diameter
	76.20	mm	Finger gear pitch diameter, metric
d_pinion	1.875	in	Pinion pitch diameter
	47.63	mm	Pinion pitch diameter, metric

What torque must the motor + gearbox supply to the pinion?

tau_pinion	12.18	Nm	Torque that must be supplied by gearbox to pinion

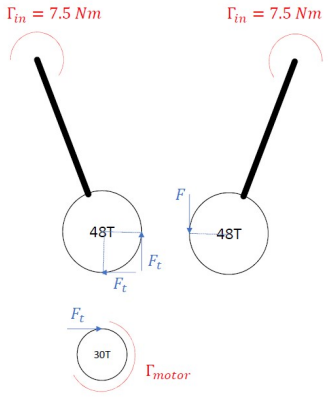
What is the radial loading on the gearbox shaft?

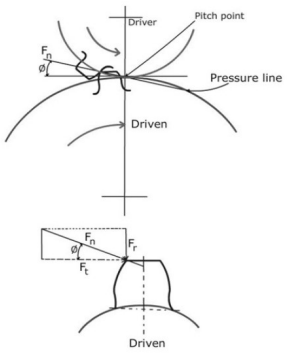
F_t	511.46	N	Force tangential to the pinion, function of torque and dia.
F_r	132.27	N	Gear seperation force
F_n	528.28	N	Total radial force on the gearbox shaft

What is maximum speed that is required of the gearbox?

I_time	0.50	s	Inspiration time
arm_sp	0.71	rad/s	Speed arm must move to achive selected tidal volume
	6.75	RPM	
pinion_sp	1.13	rad/s	Pinion speed during this move
	10.80	RPM	Speed that must be supplied by gearbox

Constants
25.4 mm/inch
98.06 Pa/cmH2O
1000 mm/m





https://www.engineersedge.com/gears/force_analysis_spur_gears_13977.htm

E-Vent gear stress and fatigue analysis

Purpose: This spreadsheet is designed to calculate the maximum bending stress experienced by gear teeth. This should be compared to stress – fatigue curves for a given candidate material.

Calculations: Design equations from Shigley & Mischke, Chapter 13
Calculations are for the pinion (small gear) which has smaller tooth faces and must transmit torque for both arms. This spreadsheet focuses on conservative values.
Updated 14 April 2020

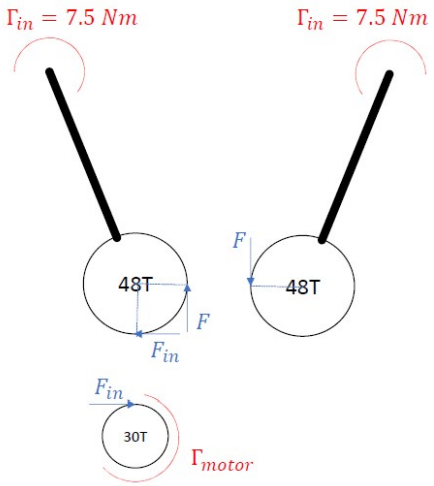
Input loads

tau_in	7.5	Nm	Input torque per finger required to drive bag This will later be multiplied since there are two fingers

Constants
25.4 mm/inch
6.895 MPa/kpsi

Gear description inputs

P	16	TPI	Diametral pitch for gear set
P.A.	14.5	deg	Pressure angle, not used in this spreadsheet
FW	0.250	in	Gear face width
	6.35	mm	Gear face width, metric
n_finger	48	teeth	Finger gear - number of teeth
n_pinion	30	teeth	Drive pinion - number of teeth
d_finger	3.000	in	Finger gear pitch diameter
	76.20	mm	Finger gear pitch diameter, metric
d_pinion	1.875	in	Pinion pitch diameter
	47.63	mm	Pinion pitch diameter, metric



Intermediate gear geometry calculations (AGMA standard)

Calculate tooth thickness (base at width of tooth)
Need to know the circular pitch

p	0.196	in	circular pitch (distance between teeth)
	4.99	mm	circular pitch, metric
t	0.098	in	tooth thickness, found at the pitch line
	2.49	mm	tooth thickness, metric

How long is each tooth? Add dedendum to addendum

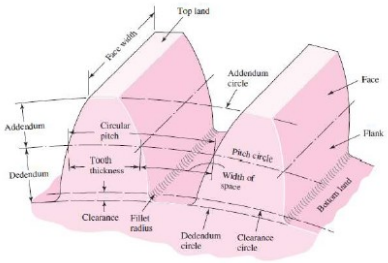
a	0.063	in	addendum
b	0.078	in	dedendum
L	0.141	in	tooth length (addendum+dedendum)
	3.57	mm	tooth length (metric)

Force required to produce required torque

The pinion must drive the finger with double the torque of a single finger
How much load does the gear tooth experience?
WORST CASE - Assume all load on a single tooth at the tip

tau_finger	15	Nm	Driving torque on finger
	15000	Nmm	(convert to N-mm)
r_finger	38.10	mm	Finger gear radius (half of gear diameter)
F_tip	394	N	Force at pinion tooth tip

Figure 13–5
Nomenclature of spur-gear teeth.



What stress does this force cause at the gear root?
Use beam bending calculations

M_max	1406.25	N mm	Bending moment on tooth
I_tooth	8.205	mm^4	Area moment of inertia of tooth "beam"
y_max	1.25	mm	Location along beam height of max bending stress
sigma_bend	213.7	MPa	Max bending stress on beam

There's a fillet at the tooth root - what stress concentration does this entail?

r_f	0.019	in	Full-depth tooth root fillet (assumes 20 deg P.A.)
	0.48	mm	Metric
r_f/t	0.191		Ratio of root fillet, to tooth thickness - needed for stress conc lookup
	1.5		Stress concentration factor - from Chart 3.7 in Peterson's Stress Concentration Factors (3rd ed.)
sigma_net	320.5	MPa	Max bending stress, with stress concentration factor
	46.5	kpsi	Same, in pound units

Fatigue considerations

First consider aluminum
Aluminium does not have an endurance limit, it will always fatigue, so we have to consult test data

UTS_Al	310	MPa	UTS of 6061-T6 Al (measured)
	0.4		Endurance limit derating for 500 million cycles (adjusts the alternating stress to reach 500 million cycles)
max. allow	124	MPa	Allowable stress to reach 500 million cycles, this is an approximation for infinite life
	18.0	kpsi	Same, in pound units

To use the plot below, find applied stress on the y-axis and read the life from the x-axis or, as an alternative, use the max. allowable stress for "infinite" life and stay below this level
For example, for the 7.5 N-m input (15 N-m total) the stress is 320 MPa. Life can be estimated as 10K cycles.
Since the load is not alternating (no significant load on out stroke), the life can be doubled to 20K cycles.
At 40 breaths per minute, damage would be expected to show up in under a day of cycles.
In conclusion, although this is a highly conservative estimate, Aluminium will not survive.

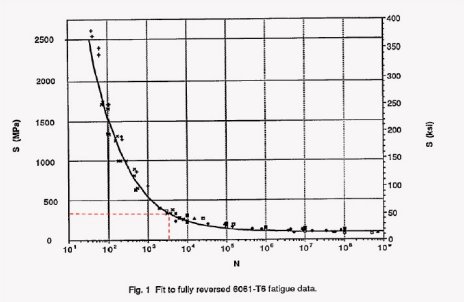
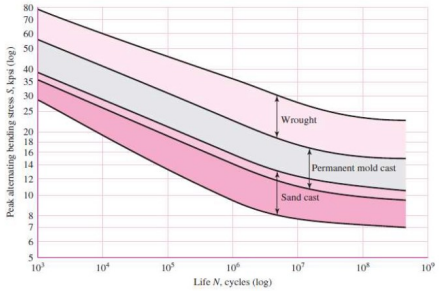


Fig. 1 Fit to fully reversed 6061-T6 fatigue data.

Fatigue data from G.T. Yahr, "Fatigue Design Curves for 6061-T6 Aluminum", Oak Ridge National Laboratory 1993

Figure 6–11

S-N bands for representative aluminum alloys, excluding wrought alloys with $S_{ut} < 38$ kpsi. *(From R. C. Juvinall,* Engineering Considerations of Stress, Strain and Strength. *Copyright © 1967 by The McGraw-Hill Companies, Inc. Reprinted by permission.)*



Next consider steel
The plot below shows fatigue limits for two different types of steel; stress values below these curves are unlikely to cause fatigue.
The alternating stress for non-reversed loads is half the peak stress. (If you imagine the load as a sinusoid - it's the amplitude of the load.)
Half of the peak stress is 160 MPa, which is at the endurance limit of A36 steel, so steel will probably serve in this application.

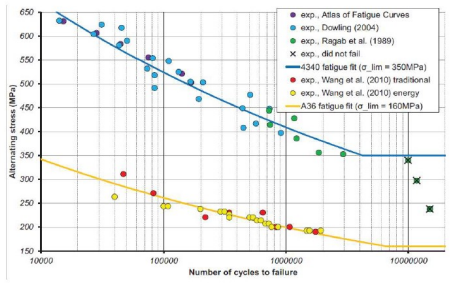


Figure 6: S-N curve fits of ASTM A36 steel based on HCF data by Wang et al. *[21] and AISI 4340 steel based on HCF data from Atlas of Fatigue Curves [22], Dowling [23] and Ragab* et al. *[24]*

Gorash, Y., & MacKenzie, D. (2017). On cyclic yield strength in definition of limits for characterisation of fatigue and creep behaviour. *Open Engineering*, *7* (1), 126-140.

Bibliografía

[1] "Clinical - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/clinical/

[2] R. Vargas, "Bag valve mask mathematical model," p. 6, 2021.

[3] "Mechanical - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/mechanical/

[4] "Plumbing - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/mechanical/plumbing/

[5] "High Level Control - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/controls/high-level-controls/

[6] E-vent Team, "Waveform Analysis for the MIT Emergency Ventilator," 2020.

[7] "Experiments And Results - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/testing-results/

[8] ResMed, "Manual de usuario Astral 100 - 150," p. 171, 2014.

[9] MAGNAMED, "Manual de Operación y Servicio OxyMag," 2014. [En línea]. Disponible: www.maanamed.com.br

[10] "Nuevo coronavirus 2019." [En línea]. Disponible: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019

[11] "¿Cómo decidir el uso de ventiladores para los pacientes por coronavirus en la UCI? | Cluster Salud | AméricaEconomía." [En línea]. Disponible: https://clustersalud.americaeconomia.com/insumos-y-servicios-hospitalarios/como-decidir-el-uso-de-ventiladores-para-los-pacientes-por-coronavirus-en-la-uci

[12] "Medtronic. (n.d.)." [En línea]. Disponible: https://www.medtronic.com/us-en/e/open-files.html

[13] "Medtronic Shares Ventilation Design Specifications to Accelerate Efforts to Increase Global Ventilator Production | Medtronic. (n.d.)." [En línea]. Disponible: https://newsroom.medtronic.com/news-releases/news-release-details/medtronic-shares-ventilation-design-specifications-accelerate/

[14] "MIT Emergency Ventilator | Design Toolbox." [En línea]. Disponible: https://emergency-vent.mit.edu/

[15] AAMI, "Emergency Use Ventilator (EUV) Design Guidance. AAMI Consensus Report," vol. 3, num. 2, pp. 54–67, 2020. [En línea]. Disponible: https://www.aami.org/docs/default-source/standardslibrary/200410_cr501-2020_rev1-2.pdf?sfvrsn=699e62b7_2%0Ahttp://repositorio.unan.edu.ni/2986/1/5624.pdf

[16] M. Perez, J. Gomez, y R. Dieguez, "Características clínico-epidemiológicas de la COVID-19," *Revista Habanera de Ciencias Médicas*, num. April, p. 15, 2020.

[17] M. de sanidad. y Centro de Coordinación de Alertas y Emergencias Sanitarias., "Información científica-técnica versión 17 Abril 2020," *17 de Abril*, pp. 1–54, 2020. [En línea]. Disponible: https://www.aemps.gob.es/

[18] A. Khoury, A. De Luca, F. S. Sall, L. Pazart, y G. Capellier, "Ventilation feedback device for manual ventilation in simulated respiratory arrest: A crossover manikin study," *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine*, vol. 27, num. 1, oct 2019.

[19] Organización Panamericana de la Salud, "Recomendaciones para la expansión de capacidades clínicas y despliegue de equipos médicos de emergencia - OPS/OMS | Organización Panamericana de la Salud," 2020. [En línea]. Disponible: https://www.paho.org/es/documentos/recomendaciones-para-expansion-capacidades-clinicas-despliegue-equipos-medicos

[20] T. Yoshida, M. B. Amato, B. P. Kavanagh, y Y. Fujino, "Impact of spontaneous breathing during mechanical ventilation in acute respiratory distress syndrome," *Current Opinion in Critical Care*, vol. 25, num. 2, pp. 192–198, apr 2019. [En línea]. Disponible: https://journals.lww.com/co-criticalcare/Fulltext/2019/04000/Impact_of_spontaneous_breathing_during_mechanical.17.aspx

[21] N. I. H. Nhlbi, A. Clinical, N. Mechanical, V. Protocol, A. Nicolini, C. Cilloniz, I. G. Piroddi, P. Faverio, N. K. Ghatehorde, y H. Regunath, "PBW and Tidal Volume for Females PBW and Tidal Volume for Males," *Community Acquired Infection*, vol. 2, num. 48, pp. 1–12, 2000. [En línea]. Disponible: http://www.ncbi.nlm.nih.gov/pubmed/29083689%0Ahttp://www.ardsnet.org/files/pbwtables_2005-02-02.pdf

[22] S. S. Batah y A. T. Fabro, "Pulmonary pathology of ARDS in COVID-19: A pathological review for clinicians," p. 106239, jan 2021.

[23] S. Díaz, "ENSAMBLE Y PROGRAMACIÓN DE UN PROTOTIPO DE RESPIRADOR ARTIFICIAL DE BAJO COSTO CON TRES MODOS DE OPERACIÓN," *Tesis de grado presentada como requisito para la obtención del título de Ingeniera Mecánica*, vol. 1, p. 50, 2015.

[24] "ISO - ISO 80601-2-79:2018 - Medical electrical equipment — Part 2-79: Particular requirements for basic safety and essential performance of ventilatory support equipment for ventilatory impairment." [En línea]. Disponible: https://www.iso.org/standard/68843.html

[25] H. Singh Johar y K. Yadav, "DRDO 's Portable Low - Cost Ventilator : " DEVEN "," *Transactions of the Indian National Academy of Engineering*, vol. 5, num. 2, pp. 365–371, 2020. [En línea]. Disponible: https://doi.org/10.1007/s41403-020-00143-5

[26] Z. Fang, A. I. Li, y H. Wang, "AmbuBox : A Fast-Deployable Low-Cost Ventilator for COVID-19 Emergent Care," 2020.

[27] T. K. Kaul y G. Mittal, "Mapleson's Breathing Systems," *Indian Journal of Anaesthesia*, vol. 57, num. 5, p. 507, sep 2013. [En línea]. Disponible: /pmc/articles/PMC3821268//pmc/articles/PMC3821268/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC3821268/

[28] "List of Alarms - MIT Emergency Ventilator." [En línea]. Disponible: https://emergency-vent.mit.edu/controls/list-of-alarms/

[29] "Paul Carrion. (2021, September 21). Construcción de un ventilador manual invasivo de asistencia mecánica para respiración asistida." [En línea]. Disponible: https://www.youtube.com/watch?v=OXJKeZAx_A4

[30] "ISO - ISO 80601-2-79:2018 - Medical electrical equipment — Part 2-79: Particular requirements for basic safety and essential performance of ventilatory support equipment for ventilatory impairment." [En línea]. Disponible: https://www.iso.org/standard/68843.html