



**SOFTWARE PARA PROTOTIPO ELECTRONICO DE VENTILADOR  
MECANICO CON PROTOCOLO DE INTERNET DE LAS COSAS  
QUE APORTE A LA RECUPERACIÓN DE LA CAPACIDAD  
RESPIRATORIA DE PACIENTES AFECTADOS POR COVID19**

**JULIAN ARMANDO DUQUE ALAYON**

**UNIVERSIDAD DE LOS LLANOS  
FACULTAD DE CIENCIAS BASICAS E INGENIERIA  
ESCUELA DE INGENIERÍAS  
PROGRAMA DE INGENIERÍA ELECTRÓNICA  
VILLAVICENCIO, COLOMBIA  
2022**

**DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ GRÁFICA CON  
PROTOCOLO DE INTERNET DE LAS COSAS QUE PERMITA LA  
VISUALIZACIÓN REMOTA DE LOS DATOS DE OPERACIÓN DE UN  
PROTOTIPO DE VENTILADOR MECÁNICO**

**JULIAN ARMANDO DUQUE ALAYON**


Trabajo de grado presentado como requisito parcial para optar al título de  
INGENIERO ELECTRÓNICO

Director:  
PhD. Camilo Torres Gómez

Codirector:  
M.Sc. Jairo David Cuero Ortega

Asesor:  
Ing. Bladimir Pineda

**UNIVERSIDAD DE LOS LLANOS  
FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA  
ESCUELA DE INGENIERÍA  
PROGRAMA DE INGENIERÍA ELECTRÓNICA  
VILLAVICENCIO, COLOMBIA  
2022**

	<b>UNIVERSIDAD DE LOS LLANOS</b>	<b>CÓDIGO: FO-DOC-97</b>	
		<b>VERSIÓN: 02</b>	<b>PÁGINA: 3 de 54</b>
	<b>PROCESO DOCENCIA</b>	<b>FECHA: 02/09/2016</b>	
	<b>FORMATO AUTORIZACION DE DERECHOS</b>	<b>VIGENCIA: 2016</b>	

## FACULTAD DE CIENCIAS BÁSICAS E INGENIERIA

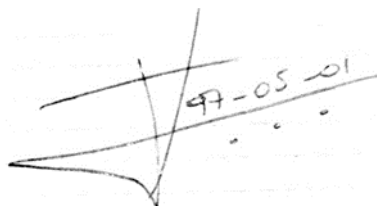
### AUTORIZACIÓN

Yo Julian Armando Duque Alayon mayor de edad, vecino de Villavicencio, Meta, identificado con la Cédula de Ciudadanía No. 1.121.945.009 de Villavicencio, actuando en nombre propio en mi calidad de autor del trabajo de EPI o trabajo de grado denominado: **DISEÑO E IMPLEMENTACION DE UNA INTERFAZ GRAFICA CON PROTOCOLO DE INTERNET DE LAS COSAS QUE PERMITA LA VISUALIZACION REMOTA DE LOS DATOS DE OPERACIÓN DE UN PROTOTIPO DE VENTILADOR MECANICO**, hago entrega del ejemplar y de sus anexos de ser el caso, en formato digital o electrónico (CD-ROM) y autorizo a la **UNIVERSIDAD DE LOS LLANOS**, para que en los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia, con la finalidad de que se utilice y use en todas sus formas, realice la reproducción, comunicación pública, edición y distribución, en formato impreso y digital, o formato conocido o por conocer de manera total y parcial de mi trabajo de grado o tesis.

**EL AUTOR – ESTUDIANTE**, Como autor, manifiesto que el trabajo de grado o tesis objeto de la presente autorización, es original y se realizó sin violar o usurpar derechos de autor de terceros; por tanto, la obra es de mi exclusiva autoría y poseo la titularidad sobre la misma; en caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, como autor, asumiré toda la responsabilidad, y saldré en defensa de los derechos aquí autorizados, para todos los efectos la Universidad actúa como un tercero de buena fe.

Para constancia, se firma el presente documento en dos (2) ejemplares del mismo valor y tenor en Villavicencio - Meta, a los 25 días del mes de julio de dos mil veintidós (2022).

EL AUTOR – ESTUDIANTE



Firma: \_\_\_\_\_

Nombre: Julian Armando Duque Alayon  
C.C. 1.121.945.009

**Nota De Aceptación:**

---

---

---

---

---

---

---

---



---

**ING. Camilo Torres Gómez**  
Jurado y director de trabajo de grado



---

**ING. Jairo David Cuero Ortega**  
Codirector

## **DEDICATORIA**

De la forma más sincera atribuyó este logro en mi vida a las personas que se han involucrado en la realización de este trabajo, sin embargo, merecen reconocimiento especial mi Madre que con su esfuerzo y dedicación me ayudaron a culminar mi carrera universitaria y me dieron el apoyo suficiente para no decaer cuando todo parecía complicado e imposible.

De igual forma, agradezco a mi director de Tesis, que gracias a sus consejos y correcciones hoy puedo culminar este trabajo. A los Profesores que me han visto crecer como persona, y gracias a sus conocimientos hoy puedo sentirme dichoso y contento.

## **AGRADECIMIENTOS**

Agradecimiento especial al Ingeniero Camilo Torres quien nos inspiró a realizar este proyecto enfocado a trabajar para encontrar alternativas más asequibles y recursivas ante una problemática de nuestra región, y quien nos apoyó durante todo el tiempo que pudo, al Ingeniero Jairo Cuero, por su dedicación, por todo el conocimiento que nos transmitió y su paciencia hacia nosotros y a todos los ingenieros, ingenieras y docentes que hicieron parte de nuestra formación profesional, por ofrecer sus conocimientos, tiempo y experiencias que nos han permitido crecer no solo como personas sino también como profesionales integrales, resaltando siempre la preocupación por el mejoramiento continuo de nuestra sociedad a través de la educación, aplicando el arte y técnica de los conocimientos científicos a la invención, diseño, perfeccionamiento y manejo de nuevos procedimientos que puedan llegar a existir no solo en la industria sino en la sociedad y otros campos de aplicación científicos, que es lo que caracteriza la palabra ingeniería.

De igual manera agradezco a mi madre quien me dio el apoyo emocional y sustento económico para llegar hasta donde estoy. Y finalmente a la Universidad de los Llanos por abrirnos sus puertas, por el apoyo que nos brindó durante las diferentes etapas de este proyecto.

## **TABLA DE CONTENIDO**

<b>DEDICATORIA</b>	<b>5</b>
<b>AGRADECIMIENTOS</b>	<b>6</b>
<b>TABLA DE CONTENIDO</b>	<b>5</b>
<b>LISTA DE FIGURAS</b>	<b>9</b>
<b>RESUMEN</b>	<b>11</b>
<b>ABSTRACT</b>	<b>12</b>
<b>1. INTRODUCCIÓN</b>	<b>13</b>
<b>2. OBJETIVOS</b>	<b>15</b>
<b>3. JUSTIFICACIÓN</b>	<b>16</b>
<b>4. METODOLOGÍA</b>	<b>17</b>
<b>5. MARCO REFERENCIAL</b>	<b>19</b>
<b>5.1 MARCO CONTEXTUAL</b>	<b>19</b>
<b>5.2 ESTADO DEL ARTE</b>	<b>19</b>
<b>5.3 MARCO TEÓRICO</b>	<b>21</b>
<b>5.3.1.1 MPX5010DP</b>	<b>21</b>
<b>5.3.1.2 Divisor de voltaje</b>	<b>23</b>
<b>5.3.1.3 Conexión fuente de alimentación</b>	<b>23</b>
<b>5.3.1.4 Tubo Venturi</b>	<b>24</b>
<b>5.3.2.1 ESP32</b>	<b>21</b>
<b>5.3.2.2 CONECTIVIDAD INALÁMBRICA</b>	<b>26</b>
<b>5.3.2.3 CONVERTOR ANALÓGICO DIGITAL</b>	<b>26</b>
<b>5.3.2.4 CONVERTOR DIGITAL ANALÓGICO</b>	<b>27</b>
<b>5.3.3 PYTHON</b>	<b>27</b>
<b>5.3.4 QT DESIGNER</b>	<b>28</b>
<b>5.3.5 PYQT5</b>	<b>29</b>
<b>5.3.6 PROTOTIPO DE VENTILADOR MECANICO</b>	<b>29</b>
<b>6. DESARROLLO DEL PROYECTO</b>	<b>31</b>
<b>6.1 Arduino Script</b>	<b>37</b>

6.2 HTML Script:	37
6.3 Python Script:	38
6.4 GUI Código	39
6.5 MAIN CODE	40
7. RESULTADOS Y ANÁLISIS	46
7.1 RESULTADOS	46
7.1.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL PROTOTIPO DE UN VENTILADOR MECANCIO	46
7.2 ANÁLISIS	49
7.2.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL PROTOTIPO DE VENTILADOR MECANICO	49
8. CONCLUSIONES	51
9. REFERENCIAS	53



## LISTA DE FIGURAS

Figure 1. Sensor MPX5010DP [39] .....	22
Figure 2. <i>Salida vs presión diferencial</i> [39].....	22
Figure 3. Divisor de voltaje resistivo de varias resistencias en serie [40].....	23
Figure 4. Divisor de voltaje de dos resistencias. [40].....	23
Figure 5. Conexión a fuente de alimentación MPX5010DP. [39] .....	24
Figure 6. Tubo Venturi en vista lateral. [41].....	25
Figure 7. <i>Tarjeta programable ESP32</i> . [40] .....	26
Figure 8. Esquema de la conectividad inalámbrica de la ESP32. [40] .....	26
Figure 9. Señal DAC. [40] .....	27
Figure 10. Logo de Python. [41].....	28
Figure 11. <i>Logo de QT</i> [42] .....	29
Figure 12. <i>Motor limpiaparabrisas</i> . .....	29
Figure 13. Prototipo inicial del respirador mecánico. ....	30
Figure 14. Modulo Puente H IBT-2.....	31
Figure 15. tarjeta ESP32 DEVKIT V1. [40] .....	32
Figure 16. Circuito de adquisición de datos.....	32
Figure 17. <i>Montaje sistema de captura de datos</i> . ....	33
Figure 18. Diagrama estructura organizacional ESP32. ....	33
Figure 19. Código Arduino Sketch Librerías. ....	34
Figure 20. Código Arduino Sketch variables. ....	34
Figure 21. <i>Código Arduino Sketch Funciones</i> . ....	35
Figure 22. <i>Código Arduino Sketch Setup</i> . ....	35
Figure 23. <i>Código Arduino Sketch Setup</i> . ....	36
Figure 24. Código Arduino Void Loop. ....	36
Figure 25. Código HTML Librería. ....	37
Figure 26. Código HTML Instancias de cada gráfica. ....	37
Figure 27. Código HTML Setup gráfica. ....	38
Figure 28. Código HTML Actualización gráfica.....	38
Figure 29. <i>Gui Front QtDesigner Page Home</i> . ....	39
Figure 30. <i>Gui Front QtDesigner Page Graph</i> . ....	40
Figure 31. Código Python Main code Importación de módulos.....	41
Figure 32. Código Python Main código creación de hilos. ....	42
Figure 33. Código Python Main código creación de hilos. ....	43
Figure 34. Código Python Main código creación de hilos .....	43
Figure 35. Código Python Main code Configuración inicial.....	43
Figure 36. Código Python Main code function graph properties. ....	44
Figure 37. Código Python Main code function value. ....	44
Figure 38. Código Python Main code function update_graph. ....	44
Figure 39. Código Python Main code función reboot_graph.....	44

Figure 40. Código Python Main code función monitor_mode. ....	45
Figure 41. Código Python Main code función actualPage. ....	45
Figure 42. Código Python Main code función ePrompt.....	45
Figure 43. Código Python Main code función menu_buttons.....	45
Figure 44. Código Python Main code función principal.....	45
Figure 45. Monitor Serial Arduino IDE, IP ESP32.....	46
Figure 46. Interfaz de usuario diseño final página de inicio. ....	47
Figure 47. <i>Monitor Python muestreo</i> . ....	47
Figure 48. HTML aplicación en red local. ....	48

## RESUMEN

El análisis, tratamiento y monitoreo permanente que se da a los datos proporcionados por los dispositivos electrónicos que componen un sistema determinado a través de la instrumentación virtual es de gran interés porque le permite al usuario contar con un soporte en cuanto a su funcionamiento, comportamiento y rendimiento. Para lograr dichos alcances se hace uso de un software con el fin de lograr sencillez, versatilidad, en cuanto a su uso y sin ningún tipo de restricciones, es decir, que sea completamente libre tanto el lenguaje de programación como la tarjeta de programación.

Por ende en este proyecto denominado “DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ GRÁFICA CON PROTOCOLO DE INTERNET DE LAS COSAS QUE PERMITA LA VISUALIZACIÓN REMOTA DE LOS DATOS DE OPERACIÓN DE UN PROTOTIPO DE VENTILADOR MECÁNICO”, el cual está orientado a construir una interfaz virtual para monitoreo y control de un respirador mecánico de bajo costo que permita no solo hacer uso de los datos capturados por los sensores del prototipo de forma local, sino realizar una consulta de los datos obtenidos de forma remota haciendo una transferencia de datos a una página web que los renderiza con el fin de dar mayor libertad de consulta a los trabajadores de la salud siendo estos capaces de acceder a la información desde cualquier dispositivo que esté conectado a la misma red del ventilador mecánico.

Para la ejecución de este sistema virtual (interfaz) se implementa el lenguaje de programación Python por su versatilidad en construcción de interfaces y tratamiento de datos, en cuanto al hardware (micro controladores) que permite la adquisición de datos analógicos que brindan los sensores se emplea la ESP32 y la Raspberry Pi 3b+, son tarjetas con características muy especiales que van desde los periféricos con los que cuenta hasta sus múltiples lenguajes de programación que acepta, la app móvil donde se pueden visualizar las mismas características del comportamiento y funcionamiento de los dispositivos se desarrolló con la ayuda de tecnologías como HTML5, CSS3 y JavaScript.

**Palabras clave:** Ventilador mecánico, micro controlador, conexión remota.

## ABSTRACT

The analysis, treatment and permanent monitoring that is given to the data provided by the electronic devices that make up a given system through virtual instrumentation is of great importance since it allows the user to have support in terms of its operation, behaviour and performance. To achieve these scopes, software and hardware are used where versatility, ease of use and without any restrictions are achieved, that is, both the programming language and the programming card are completely free.

Therefore, in this project called "DESIGN AND IMPLEMENTATION OF A GRAPHICAL INTERFACE WITH INTERNET OF THINGS PROTOCOL THAT ALLOWS REMOTE VIEWING OF THE OPERATION DATA OF A MECHANICAL FAN PROTOTYPE", which is aimed at building a virtual interface for monitoring and control of a low-cost mechanical respirator that allows not only to make use of the data captured by the prototype sensors locally, but also to query the data obtained remotely by transferring the data to a web page that rendered in order to give greater freedom of consultation to health workers, being able to access the information from any device that is connected to the same network as the mechanical ventilator.

For the execution of this virtual system (interface), the Python programming language is implemented due to its versatility in the construction of interfaces and data processing, in terms of hardware (microcontrollers) that allows the acquisition of analogue data provided by the sensors. the ESP32 and the Raspberry Pi 3b+, are cards with very special characteristics that go from the peripherals that it has to its multiple programming languages that it accepts, the mobile app where the same characteristics of the behaviour and operation of the devices can be visualized. developed with the help of technologies such as HTML5, CSS3 and JavaScript.

**Keywords:** Mechanical fan, microcontroller, remote connection.

## 1. INTRODUCCIÓN

A principios del año 2020, la Organización Mundial de la Salud (OMS) declaró la emergencia mundial a causa del rápido avance de las infecciones por el COVID-19. De inmediato, los países alertaron a sus sistemas sanitarios con el fin de prevenir y mitigar el impacto de una infección grave y altamente contagiosa y de la que no se conocía mucho. Los esfuerzos se enfocaron entonces a la detección temprana de los casos presentados y al manejo oportuno de los pacientes que desarrollaban las formas más graves de la enfermedad, especialmente pacientes mayores de 70 años o con comorbilidades asociadas por su alta letalidad (Nova Sepúlveda, 2020).

A nivel global la cantidad de pacientes que requerían asistencia ventilatoria superó el número de camas disponibles en la unidad de cuidados intensivos (UCI). Como respuesta a esto, un porcentaje de camas generales se convirtieron en camas UCI y los hospitales generales en hospitales de cuidados críticos (Heredia & otros, 2021). Debido a esta problemática y al crecimiento exponencial del número de casos en la expansión de la pandemia y dada la saturación de demanda en el mercado mundial de aparatos médicos; una de las acciones llevadas a cabo en muchos países fue activar la fabricación de ventiladores mecánicos de emergencia (Farre & otros, 2020).

Hoy en día se tiene un manejo moderado de las crisis hospitalarias presentadas en los picos de la pandemia gracias a las recientes vacunas y protocolos de bioseguridad, sin embargo, en todas las unidades médicas y en especial en partes del país alejadas se ve escasez de camas UCI, respiradores o tanques de oxígeno para el correcto trato de enfermedades respiratorias o manejo de síntomas presentados por el COVID 19 (Félix & Palate, 2021) al punto de sobrepasar las unidades de ventiladores mecánicos disponibles para uso e incluso utilizando unidades de reserva y tras la compra de nuevos equipos en un país con limitados recursos en salud, generando potenciales fallas graves en la atención de esta población por ausencia de dicha tecnología (Rada & Patiño, 2021).

Hasta el momento no existe cura específica para el COVID-19, sin embargo, la ventilación mecánica es una de las principales estrategias para contrarrestar los peligrosos efectos de la insuficiencia respiratoria observada en esta enfermedad ya que corresponde a un método de soporte ventilatorio, a través del cual se reemplaza la función ventilatoria del pulmón (Arellano, 2006).

Esta es utilizada hasta que la condición del paciente mejore y ha sido una medida paliativa y de manejo, ya que solo restablece el intercambio de gases mientras transcurre la evolución natural de la enfermedad en un solo paciente (Vásquez & otros, 2020) y gracias a los avances tecnológicos brinda la oportunidad de suministrar un soporte avanzado de vida eficiente a los pacientes que se encuentran con un Síndrome de Dificultad Respiratoria Aguda.

## 1.1 PLANTEAMIENTO DEL PROBLEMA

En las últimas décadas el desarrollo tecnológico respecto a la biomedicina se ha vuelto de gran importancia para adaptarse a los requerimientos frente ante cualquier enfermedad ya que esto ayuda a futuro para la economía y para mejorar la calidad de vida, debido a la magnitud de esto y a la capacidad de investigación e innovación que existe, se reconoce la necesidad a nivel global de la formación y entrenamiento en todos los niveles respecto a este tema. [1] claramente se han venido desarrollando tecnologías para la cual ofrecen accesibilidad a los diferentes tipos de instrumentos biomédicos existentes hasta en los niveles más básicos, lo que lleva a reconocer como la educación debe avanzar a la par con la tecnología destacando como esto hace parte de la educación, para la inclusión de herramientas para el desarrollo de personas capacitadas para enfrentar los retos tecnológicos actuales, [2] existen instituciones dedicadas a construir estrategias para fortalecer la transferencia de este conocimiento por medio de recursos físicos como es el caso del La universidad de los Llanos, consciente de su responsabilidad con la región apoya la iniciativa de desarrollar un sistema de ventilación mecánica que contribuya al sistema de salud de la región. Adicionalmente se agrega al equipo propuesto un protocolo de internet de las cosas que permite al equipo contar con un sistema ciber físico para el envío de información a la nube y pueda ser leído y analizado por el profesional de la salud en una APP. El problema recae en que dichos dispositivos generalmente tienen un precio en el mercado muy elevado lo que conlleva a que hospitales que no cuentan con la financiación no puedan tener acceso a estos recursos y por lo tanto disminuye la esperanza de vida de dicha población. [3]

Es aquí donde la implementación de la instrumentación virtual crearía soluciones que permitan adquirir, analizar y presentar datos de los dispositivos aprovechando al máximo las capacidades de cálculo y comunicación de un computador moderno lo cual les concede a los interesados en manejar los equipos, aprender de ellos sin ser especialistas en software o electrónica mejorando así los procesos de enseñanza y aprendizaje de las nuevas tecnologías. [4]

Otro factor del problema a tratar; ocurre que en Colombia existe corrupción lo que conlleva en la carencia de las tecnologías suficientes para suplir las necesidades que pueden llegar a ser básicas y esenciales, sobre todo en zonas alejadas del país donde los conflictos armados muchas veces impiden el progreso debilitando aún más la capacidad para suplir dicho objetivo. [5]

Por tanto, la convergencia de saberes y conocimientos del grupo de investigación con líneas de investigación en Bioingeniería y Automatización han permitido conformar un equipo idóneo para la solución a la problemática planteada. Por todo lo anterior, se propone una alternativa tecnológica que aporta a la mitigación de las problemáticas sanitarias y de salud pública ocasionada por el COVID-19.

## **2. OBJETIVOS**

### **2.1 OBJETIVO GENERAL**

Construir un prototipo de instrumentación virtual útil para el monitoreo de las señales eléctricas generadas por el ventilador mecánico.

### **2.2 OBJETIVOS ESPECÍFICOS**

1. Identificar el comportamiento de los parámetros eléctricos presentes en los sensores utilizados en el prototipo de ventilador mecánico.
2. Elaborar una interfaz hardware-software que permita la adquisición y visualización de las variables eléctricas del prototipo de ventilador mecánico.
3. Construir una app móvil y página web que brinde al usuario el monitoreo de los diferentes datos eléctricos del prototipo de ventilador mecánico de manera inalámbrica a través de la herramienta internet.

### 3. JUSTIFICACIÓN

Para conseguir un alto rendimiento y eficiencia durante el proceso de monitoreo y control, se han venido desarrollando mediante la ayuda del internet de las cosas que junto con tecnologías como Python brindan un avance y comodidad, para que remotamente se obtengan resultados satisfactorios. Dicha combinación proporciona los principios claves esenciales para una comprensión integral, no solo de las capacidades de la parte mecánica del sistema, sino también de la parte del software, mecánica e ingeniería.

El internet de las cosas ofrece a los usuarios de la misma una red colectiva de dispositivos conectados a la nube, así como la conectividad entre los propios dispositivos. Gracias a la llegada de los chips de ordenador de bajo coste y a las telecomunicaciones de gran ancho de banda, ahora tenemos miles de millones de dispositivos conectados a Internet.

Tras una búsqueda minuciosa, se encontró que los prototipos de ventiladores mecánicos de bajo costo no suelen tener una interfaz de usuario, sin embargo, para los que si presentan pueden tener una interfaz poco comprensible, esto trae complicaciones al operador. Si los ventiladores mecánicos no cuentan con un soporte virtual que apoye la utilización de una interfaz de manera integral, con lo cual no es capaz de brindar la experiencia deseada.

En cuanto a la finalidad de bajo costo es ampliar la capacidad de utilizar los dispositivos de la mejor manera, se busca diseñar e implementar un software que permita el procesamiento, análisis y monitoreo de las variables eléctricas, las cuales se logran tomar de los dispositivos que componen dicho prototipo. Para alcanzar este objetivo se necesita la adición de un elemento virtual que cuente con la visualización de una interfaz que brinda al usuario la capacidad de monitorear, adquirir y comunicar datos, y aparte añada una experiencia más agradable y fluida durante el uso de la herramienta.

En segunda instancia la situación actual que vive Colombia con lo que respecta a la gran brecha tecnológica que existe y la alta demanda de respiradores, es propicio implementar una herramienta que permita de una manera más sencilla fomentar la implementación de dispositivos auxiliares que puedan suplir de manera momentánea la alta demanda de respiradores. Dando así origen a una búsqueda de carácter prioritario a tecnologías enfocadas principalmente en la generación de prototipos de ventiladores mecánicos de bajo costo de producción los cuales tengan la capacidad no solo suplir la alta demanda sino de una mayor comodidad para el operador ya que este podrá continuar haciendo sus deberes y ante cualquier eventualidad o si desea hacer un chequeo general podrá acceder a las señales y valores en tiempo real del ventilador sin necesidad de estar frente a él, siendo así ideales para el desarrollo y alcance del objetivo principal.



## **4. METODOLOGÍA**

La metodología de este proyecto es de carácter cuantitativo experimental de tipo interactivo, planeada y formada en fases, para llevar el control del tiempo, con el fin de cumplir las metas establecidas.

### **4.1 FASE I**

#### **IDENTIFICACIÓN:**

- Consulta de manual de funcionamiento y principios tanto eléctricos, físicos y químicos de los sensores a utilizar en el prototipo de ventilador mecánico.
- Consulta de información teórica sobre el funcionamiento de los ventiladores mecánicos.
- Medir las salidas eléctricas de los dispositivos.

### **4.2 FASE II**

#### **CONSTRUCCIÓN INTERFAZ HARDWARE-SOFTWARE Y APLICACIÓN MÓVIL:**

- Identificar las características que debe cumplir el programa.
- Designar la tarjeta de programación que ejecutará la adquisición y tratamiento de los datos proporcionados por cada dispositivo del Kit.
- Seleccionar un software libre apropiado para cumplir los objetivos.
- Diseñar el programa para monitoreo y visualización.

### **4.3 FASE III**

#### **PRUEBAS EN HOSPITAL:**

- Comparar resultados obtenidos del prototipo con los de un respirador comercial.
- Caracterizar las señales mediante el uso del VT PLUS HF Gas Flow Analyzer.

### **4.4 FASE IV**

#### **ANÁLISIS DE DATOS Y GRÁFICAS OBTENIDAS:**

- Análisis de los datos obtenidos con el fin de asegurar un correcto funcionamiento en la adquisición y procesamiento de la información.

### **4.5 FASE V**

#### **CONSTRUCCIÓN DEL DOCUMENTO FINAL:**

- Elaboración del documento final a entregar una vez concluido el trabajo.
- Conclusiones.



## **5. MARCO REFERENCIAL**

### **5.1 MARCO CONTEXTUAL**

El Meta es el departamento más importante en la región Orinoquía, zona donde la actividad económica en gran parte es agrícola y ganadera. Limita por el norte con los departamentos de Casanare y Cundinamarca; por el oriente con Vichada, Guainía y Guaviare; por el occidente con Cundinamarca, el Distrito Capital y el departamento del Huila; y por el sur con Caquetá. Su capital es la ciudad de Villavicencio que es el centro comercial más importante de los Llanos Orientales. Está ubicada en el Piedemonte de la Cordillera Oriental, al Noroccidente del departamento del Meta, en la margen izquierda del río Guatiquía. Presenta un clima cálido y muy húmedo, con temperaturas medias de 28° C y 30°C. [6]

Dentro de la infraestructura educativa universitaria con la que cuenta el municipio de Villavicencio se encuentra la Universidad de los Llanos, la única de carácter público en la Orinoquía Colombiana, cuenta con dos sedes principales, una urbana llamada sede San Antonio, ubicada en el barrio Barzal Alto y una rural llamada sede Barcelona, ubicada en el kilómetro 12 Vía a Puerto López, Vda. Barcelona. [7]

La Universidad de los Llanos se caracteriza por ser una de las universidades que cuenta con una gran cantidad de grupos de investigación que desarrollan e implementan proyectos que benefician el progreso económico, social, ambiental, tecnológico y cultural de la región y el país. Lo que compete a la Facultad de Ciencias Básicas e Ingeniería donde se encuentra el programa de ingeniería electrónica se conforma con grupos de investigación, los cuales son MACRYPT y EYSI, donde su principal enfoque de investigación y desarrollo son temas de automatización, bioingeniería, sistemas de controles analógicos y digitales, uso racional de la energía, energías renovables, entre otros temas. [8]

### **5.2 ESTADO DEL ARTE**

A principios del año 2020, la Organización Mundial de la Salud (OMS) declaró la emergencia mundial a causa del rápido avance de las infecciones por el COVID-19. De inmediato, los países alertaron a sus sistemas sanitarios con el fin de prevenir y mitigar el impacto de una infección grave y altamente contagiosa y de la que no se conocía mucho. Los esfuerzos se enfocaron entonces a la detección temprana de los casos presentados y al manejo oportuno de los pacientes que desarrollaban las formas más graves de la enfermedad, especialmente pacientes mayores de 70 años o con comorbilidades asociadas por su alta letalidad (Nova Sepúlveda, 2020).

A nivel global la cantidad de pacientes que requerían asistencia ventilatoria superó el número de camas disponibles en la unidad de cuidados intensivos (UCI). Como respuesta a esto, un porcentaje de camas generales se convirtieron en camas UCI y

los hospitales generales en hospitales de cuidados críticos (Heredia & otros, 2021). Debido a esta problemática y al crecimiento exponencial del número de casos en la expansión de la pandemia y dada la saturación de demanda en el mercado mundial de

aparatos médicos; una de las acciones llevadas a cabo en muchos países fue activar la fabricación de ventiladores mecánicos de emergencia (Farre & otros, 2020).

Hoy en día se tiene un manejo moderado de las crisis hospitalarias presentadas en los picos de la pandemia gracias a las recientes vacunas y protocolos de bioseguridad, sin embargo, en todas las unidades médicas y en especial en partes del país alejadas se ve escasez de camas UCI, respiradores o tanques de oxígeno para el correcto trato de enfermedades respiratorias o manejo de síntomas presentados por el COVID 19 (Félix & Palate, 2021) al punto de sobrepasar las unidades de ventiladores mecánicos disponibles para uso e incluso utilizando unidades de reserva y tras la compra de nuevos equipos en un país con limitados recursos en salud, generando potenciales fallas graves en la atención de esta población por ausencia de dicha tecnología (Rada & Patiño, 2021).

Hasta el momento no existe un tratamiento o cura específica para el COVID-19, sin embargo, la ventilación mecánica es una de las principales estrategias para contrarrestar los peligrosos efectos de la insuficiencia respiratoria observada en esta enfermedad ya que corresponde a un método de soporte ventilatorio, a través del cual se reemplaza la función ventilatoria del pulmón (Arellano, 2006).

Esta es utilizada hasta que la condición del paciente mejore y ha sido una medida paliativa y de manejo, ya que solo restablece el intercambio de gases mientras transcurre la evolución natural de la enfermedad en un solo paciente (Vásquez & otros, 2020) y gracias a los avances tecnológicos brinda la oportunidad de suministrar un soporte avanzado de vida eficiente a los pacientes que se encuentran con un Síndrome de Dificultad Respiratoria Aguda.

En este contexto, numerosos grupos de investigación y universidades alrededor del mundo comenzaron la tarea de desarrollar unidades de ventilación mecánica para aligerar los retos en materia logística y económica dado que el valor de este dispositivo en pesos colombianos es de alrededor de cien (100) millones. En España, un grupo de investigación compuesto por médicos, ingenieros del entorno industrial y con la colaboración de diversas empresas, ha desarrollado un sistema de ventilación mecánica invasiva que puede ser distribuido de manera generalizada y a bajo coste; lo han llamado Ventijet y utiliza un modo de ventilación de flujo continuo, que permite al paciente inspirar en todo momento limitando la incidencia de asincronías inspiratorias (Parrilla-Gomez & otros, 2020).

En Argentina se desarrolló un ventilador mecánico no invasivo (VMNI) de bajo costo, denominado IARespira, trabaja con un control de presión con tres modos de ventilación posibles: CPAP (presión positiva continua en las vías respiratorias), BiPAP (presión positiva de dos niveles) y Asistida/Controlada. Además, propuso una solución a la dificultad de acceso a insumos críticos con elementos disponibles principalmente en el mercado local. El diseño se orientó para una rápida fabricación y pronta disponibilidad en los centros de salud de la Argentina (Salibe & otros, 2021).

La universidad de los Llanos, consciente de su responsabilidad con la región apoya la iniciativa de desarrollar un sistema de ventilación mecánica que contribuya al sistema de salud de la región. Adicionalmente se agrega al equipo propuesto un protocolo de internet de las cosas que permite al equipo contar con un sistema ciber físico para el envío de información a la nube y pueda ser leído y analizado por el profesional de la salud en una APP.

Por tanto, la convergencia de saberes y conocimientos de 2 grupos de investigación con líneas de investigación en Bioingeniería y Automatización han permitido conformar un equipo idóneo para la solución a la problemática planteada. Por todo lo anterior, se propone una alternativa tecnológica que aporta a la mitigación de las problemáticas sanitarias y de salud pública ocasionada por el COVID-19.

## **5.3 MARCO TEÓRICO**

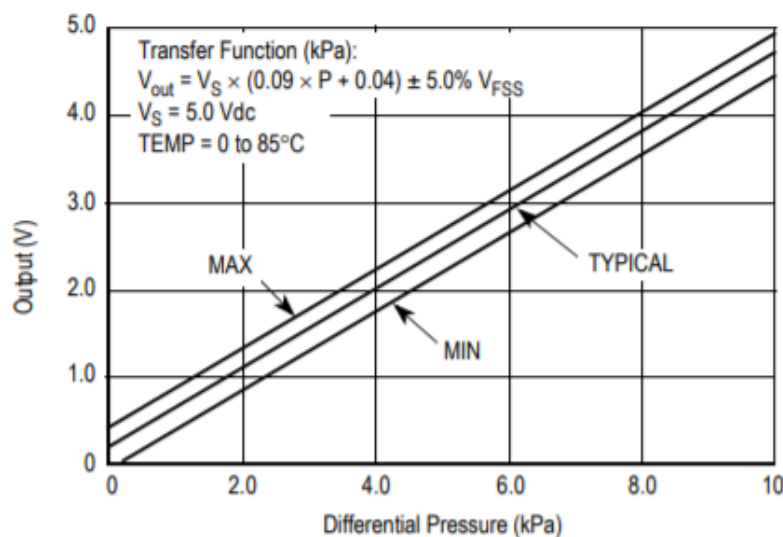
### **5.3.1.1 MPX5010DP**

El MPX5010DP es un sensor de presión de silicio integrado de doble puerto en un encapsulado SIP de 6 pines. Este transductor piezorresistivo es un sensor de presión de silicio monolítico de última generación, destinado a una gran variedad de aplicaciones. Es ideal para sistemas basados en microprocesador o microcontrolador. Este transductor combina técnicas avanzadas de micromecanizado, metalización de película delgada y procesamiento bipolar para ofrecer una señal de salida analógica precisa de alto nivel que es proporcional a la presión aplicada. El puerto axial ha sido modificado para acomodar tubos de grado industrial.



**Figure 1. Sensor MPX5010DP [39]**

Su salida un voltaje de 0v – 5v lo hace ideal para microcontroladores, a diferencia del MPX10DP que tiene un funcionamiento hasta 50mV lo que requerirá de un amplificador operacional para tratar su señal, puede medir una presión de 10 kPa. El sensor entrega una salida lineal sobre todo su rango de funcionamiento como se muestra en la figura extraída de la hoja de datos.



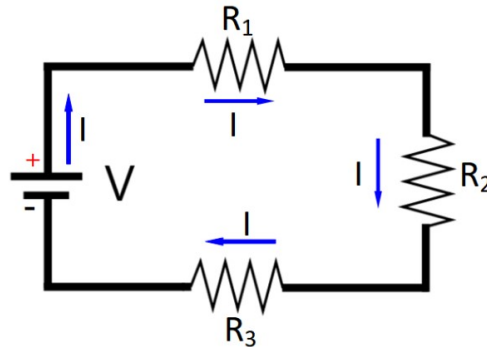
**Figure 2. Salida vs presión diferencial [39]**

Con este sensor nos encargaremos de leer la presión y el flujo generado por el prototipo, como ya mencionado anteriormente el sensor funciona de 0v a 5v lo que genera una mayor facilidad al momento de que sea conectado a un microcontrolador,

por esta razón se debe implementar un divisor de voltaje debido al microcontrolador que se implementara en el desarrollo del proyecto.

### 5.3.1.2 Divisor de voltaje

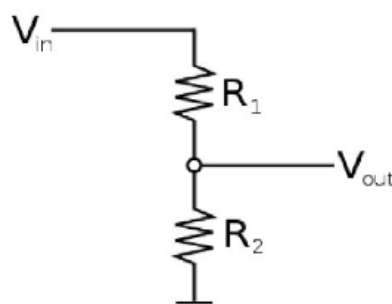
El divisor de voltaje o divisor de tensión consiste en una asociación de resistencias o impedancias en serie conectadas a una fuente. De esta manera el voltaje  $V$  suministrado por la fuente –voltaje de entrada- se reparte proporcionalmente en cada elemento, según la ley de Ohm.



**Figure 3. Divisor de voltaje resistivo de varias resistencias en serie [40]**

En un arreglo de 2 impedancias, comúnmente resistencias que dividen el voltaje y la corriente de salida. La división es proporcional a las resistencias involucradas en el divisor. Un divisor de voltaje se configura para tener una salida de potencial determinada, esta se puede calcular con una simple ecuación o formula.

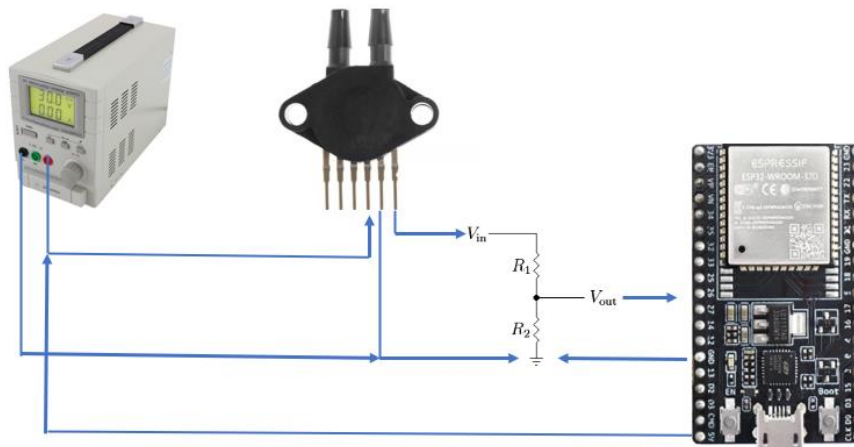
$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$



**Figure 4. Divisor de voltaje de dos resistencias. [40]**

### 5.3.1.3 Conexión fuente de alimentación

Para la conexión de los sensores es necesario alimentar con 5v, es necesario que estos no se tomen del microcontrolador sino de una fuente externa a este ya que al detectar las señales el microcontrolador genera ruido indeseado al sensor.



**Figure 5. Conexión a fuente de alimentación MPX5010DP. [39]**

El sensor debe compartir la conexión a tierra con el microcontrolador y la fuente de alimentación, en la siguiente imagen se puede ver una descripción de como conectar la alimentación del sensor y como conectar el sensor al microcontrolador mediante el divisor de voltaje.

Aunque en el esquema he utilizado una fuente de alimentación regulable de laboratorio, se puede utilizar una batería o cualquier otra fuente de alimentación.

Lo importante es que el pin 3 del sensor esté conectado al positivo de la fuente, el pin 2 se conecta tierra de la fuente. Por último, el terminal de la señal directo a la entrada del divisor de voltaje ya que este se encargará de bajar el voltaje para que el microcontrolador no sufra averías.

#### **5.3.1.4 Tubo Venturi**

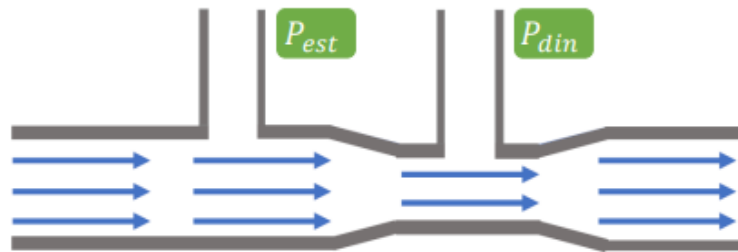
Este dispositivo es diseñado para medir la velocidad de un fluido en el interior de un conducto. Tiene una sección más estrecha, en la que el fluido experimenta una disminución de presión de acuerdo con el efecto Venturi.

Cuando un fluido que circula por el interior de un conducto cerrado pasa por un estrechamiento de dicho conducto, su velocidad aumenta y su presión disminuye. El fenómeno es conocido como efecto Venturi, por ser este el nombre del físico italiano que verificó su existencia.

A partir del conocimiento del efecto descrito por Venturi se desarrolló el tubo del mismo nombre, que es un aparato que permite medir la velocidad y el caudal de un fluido partiendo de la diferencia de presiones entre dos puntos del conducto.



$$Q_v = A_1 \cdot \sqrt{\frac{2 \cdot (P_{est} - P_{din})}{\rho \cdot \left(\left(\frac{A_1}{A_2}\right)^2 - 1\right)}}$$



**Figure 6. Tubo Venturi en vista lateral. [41]**

### 5.3.2.1 ESP32

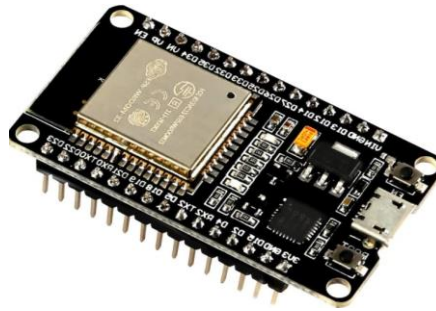
ESP32 es una serie de SoC (por sus siglas en inglés, *System on Chip*) y módulos de bajo costo y bajo consumo de energía creado por *Espressif Systems*.

Esta nueva familia es la sucesora del conocido ESP8266 y su característica más notable es que, además de Wi-Fi, también soporta Bluetooth.

Los ESP32 poseen un alto nivel de integración. En su pequeño encapsulado se incluyen:

- Interruptores de antena.
- Balun de RF.
- Amplificador de potencia.
- Amplificador de recepción de bajo ruido.
- Filtros y módulos de administración de energía.

Además de todo eso, logra un consumo de energía muy bajo a través de funciones de ahorro de energía que incluyen sincronización de reloj y múltiples modos de operación. Todo esto lo convierte en la herramienta ideal para tus proyectos energizados con baterías o aplicaciones IoT.

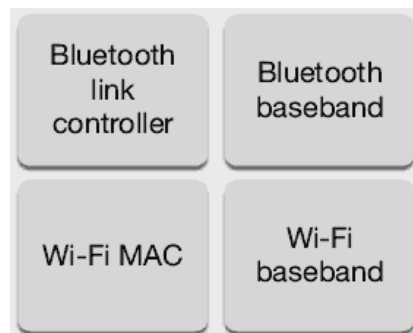


**Figure 7. Tarjeta programable ESP32. [40]**

### 5.3.2.2 CONECTIVIDAD INALÁMBRICA

El chip cuenta con conectividad WiFi, siendo compatible con 802.11 b/g/n en la banda de los 2.4GHz, alcanzando velocidades de hasta 150 Mbits/s.

También incluye comunicación Bluetooth compatible con Bluetooth v4.2 y *Bluetooth Low Energy* (BLE).



**Figure 8. Esquema de la conectividad inalámbrica de la ESP32. [40]**

### 5.3.2.3 CONVERTOR ANALÓGICO DIGITAL

Algunos de los pines también pueden ser utilizados para interactuar con sensores analógicos, es decir, como si fueran los pines analógicos de una placa Arduino.

Para esto el ESP32 cuenta con un convertor analógico digital de 12-bits y 18 canales, es decir, que puedes tomar lecturas de hasta 18 sensores analógicos.

Esto te permite desarrollar aplicaciones conectadas muy compactas, incluso cuando se empleen varios sensores analógicos.

#### 5.3.2.4 CONVERSOR DIGITAL ANALÓGICO

En la mayoría de las placas Arduino se utilizan señales PWM para generar voltajes analógicos. El ESP32 cuenta con dos conversores digital analógico.



**Figure 9. Señal DAC. [40]**

Esto permite generar dos señales de voltaje analógicas puras. Dichos conversores, pueden ser utilizados para:

- Controlar un circuito analógico.
- Manipular la intensidad de un LED.
- Agregar un pequeño amplificador y un altavoz a tu proyecto para reproducir una canción.

#### 5.3.3 PYTHON

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años, ya que facilita trabajar con inteligencia artificial, big data, machine learning y data science, entre muchos otros campos en auge. [41]



**PYTHON**

**Figure 10. Logo de Python. [41]**

### **5.3.4 QT DESIGNER**

Qt Designer es una herramienta de Qt que le proporciona una interfaz de usuario de lo que ve es lo que obtiene (WYSIWYG) para crear una GUI para sus aplicaciones PyQt de manera productiva y eficiente.

Con esta herramienta, se puede crear una GUI arrastrando y soltando objetos QWidget en un espacio vacío. Después de eso, puede organizarlos en una GUI coherente utilizando diferentes administradores de diseño.

Qt Designer también le permite obtener una vista previa de sus GUI usando diferentes estilos y resoluciones, conectar señales y ranuras, crear menús y barras de herramientas, y más.

Qt Designer es independiente de la plataforma y el lenguaje de programación. No produce código en ningún lenguaje de programación en particular, pero crea archivos .ui. Estos archivos son archivos XML con descripciones detalladas de cómo generar GUI basadas en Qt.

Puede traducir el contenido de los archivos .ui a código Python con PyQt5, que es una herramienta de línea de comandos que viene con PyQt. Entonces puede usar este código Python en sus aplicaciones GUI. También puede leer archivos .ui directamente y cargar su contenido para generar la GUI asociada. [42]



**Figure 11. Logo de QT [42]**

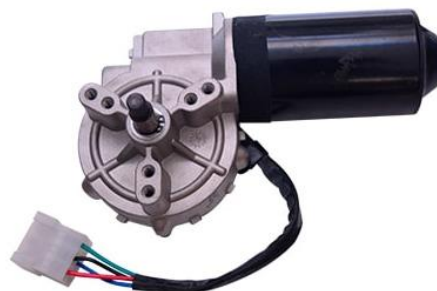
### **5.3.5 PYQT5**

Es un binding de la biblioteca gráfica QT para el lenguaje de programación Python. PyQT5 nos permite crear interfaces gráficas con python de manera rápida y sencilla, la legibilidad del código de Python hace que sea una tarea sumamente sencilla realizar interfaces gráficas, además que también posee una interfaz de diseño para crear nuestras interfaces gráficas.

La flexibilidad que tiene esta biblioteca, es que podemos diseñar por completo nuestras interfaces y luego comenzar a programar, esto es un punto muy importante. [43]

### **5.3.6 PROTOTIPO DE VENTILADOR MECANICO**

Para la construcción del respirador mecánico, teniendo en cuenta que una de las ideas principales del proyecto es de reducir el precio de producción se utilizó un motor reciclado de parabrisas de automóvil.



**Figure 12. Motor limpiaparabrisas.**

Para encapsular todo el sistema se utilizó inicialmente acrílico de un alto grosor el cual se planea ya en una fase final cambiar por lamina cold rolled calibre 18, el sistema

que proporcionará el aire será un resucitador manual de marca Ambu. Se implemento el uso de piezas impresas en PLA las cuales se diseñaron como pinzas las cuales atrapan al resucitador manual haciendo las veces de las manos del operador y una segunda impresión la cual se acopla al final de una de las garras la cual es diseñada para actuar como disco codificado para el encoder del motor. Por último, se utilizará una pantalla táctil de 7 pulgadas la cual se podrá visualizar la interfaz de usuario.



**Figure 13. Prototipo inicial del respirador mecánico.**

## 6. DESARROLLO DEL PROYECTO

Para cumplir con el objetivo general del proyecto se inició identificando las características del funcionamiento del prototipo de ventilador mecánico. Al momento de seleccionar el microcontrolador se tuvo en cuenta dos factores cruciales, mantener un presupuesto bajo y que este presentara conectividad a la red de manera integrada.

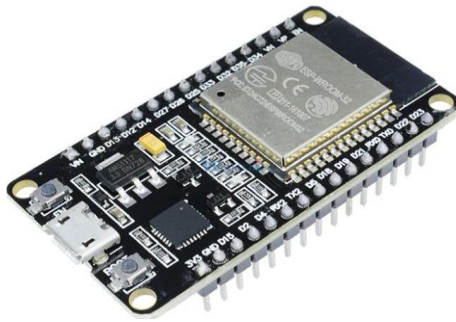
El circuito de adquisición de datos se encarga de tomar los valores de posición del motor, flujo y presión del aire, estas son las variables fundamentales de la realización del proyecto. Para el control del motor se utilizó un módulo Puente H IBT-2, se decidió utilizar este driver de alta potencia dado a su costo-beneficio, a su compatibilidad con el microcontrolador utilizado, su protección contra sobrevoltaje, su rango de operación de bajo voltaje entre otros beneficios. pero sobre todo su alta fidelidad y su alta tolerancia en altas corrientes, dado a que este driver soporta una corriente máxima de 47 Amperios.



**Figure 14. Modulo Puente H IBT-2**

Se utilizaron dos sensores ópticos Im393, el primero se utilizó para poder conocer siempre cuál será la posición inicial de la pinza y el segundo sensor se utilizó para leer la posición en la que se encuentra la pinza la cual está encargada de la compresión y descompresión del respirador mecánico a través de un encoder acoplado al chasis del ventilador mecánico. Este leerá los pasos que se han realizado dependiendo de un conteo en el número de muescas que nos proporciona el encoder.

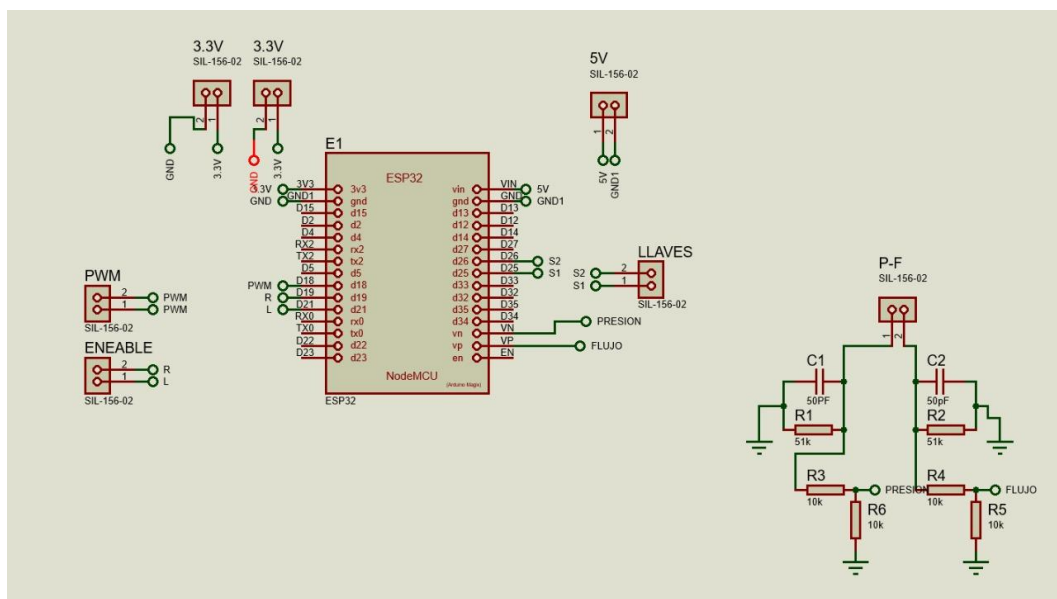
Para la comunicación de los datos con el software de visualización se utilizó una tarjeta de desarrollo ESP32 DEVKIT V1. La cual posee las características que se requieren, Wi-fi de 2.4 MHz a 150 Mbits/s, doble núcleo para mayor capacidad de procesamiento, protocolo de comunicación I2C y versatilidad de uso con su ambiente de programación que incluye lenguajes como JavaScript, MicroPython, Arduino IDE, etc.



**Figure 15. tarjeta ESP32 DEVKIT V1. [40]**

La tarjeta anteriormente mencionada se alimenta con 3.3V por intermedio del puerto microUSB al equipo.

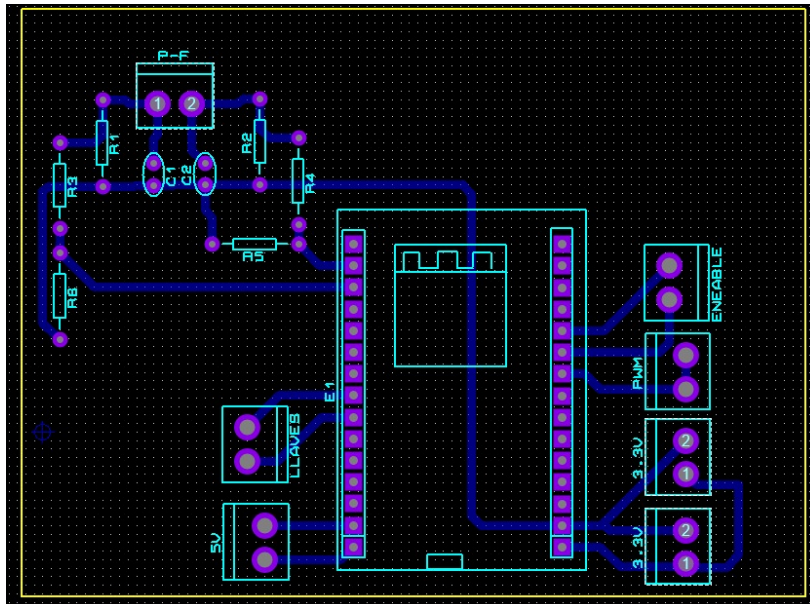
El circuito implementado para ejecutar la adquisición de los datos en tiempo real es el siguiente:



**Figure 16. Circuito de adquisición de datos.**

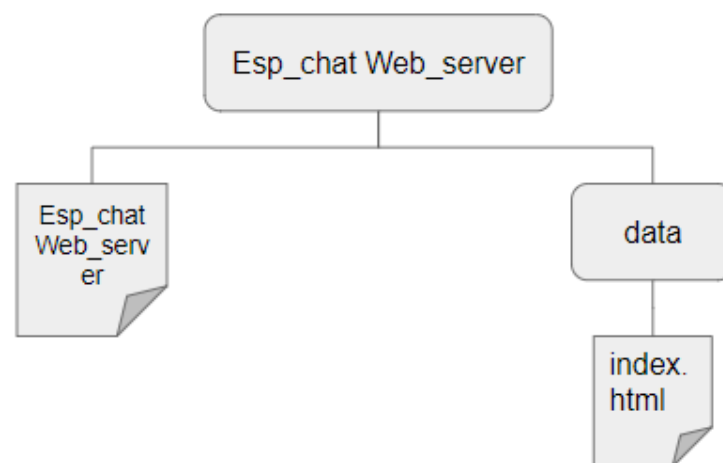
Como resultado del prototipo del circuito diseñado en esquemático, se decidió grabar el circuito en una baquelita:





**Figure 17. Montaje sistema de captura de datos.**

Se diseñó el código de la tarjeta en el IDE de Arduino, con el fin de utilizar la librería ESPAsyncWebServer.h para la creación de un servidor web asíncrono, asimismo el código HTML de la página estará también en la memoria de la ESP32 contando con gráficas para la visualización en tiempo real de las variables capturadas. El archivo HTML debe alojarse dentro de la memoria de la ESP, la cual a su vez está dentro de la ruta del Arduino sketch.



**Figure 18. Diagrama estructura organizacional ESP32.**

## 6.1 Arduino Script:

Primero se incluyen las librerías necesarias para la ESP32. Con el fin de poder ejecutar las funciones que se llevarán a cabo dentro del código.

```
// Librerías Requeridas
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <Adafruit_INA219.h>
#include <U8g2lib.h>
#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#else
#include <SoftwareSPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif
```

**Figure 19. Código Arduino Sketch Librerías.**

Se ingresan manualmente las credenciales de la red necesarias para conectar y las credenciales para la conexión MQTT junto con las variables que este usara.

```
3 // Replace the next variables with your SSID/Password combination
4 const char* ssid = "EPI";
5 const char* password = "ventilador";
6 // Add your MQTT Broker IP address, example:
7 //const char* mqtt_server = "192.168.1.144";
8 const char* mqtt_server = "10.42.0.1";
9 WiFiClient espClient;
10 PubSubClient client(espClient);
11 long lastMsg = 0;
12 char msg[50];
13 int value = 0;
14 String messageTemp;
```

**Figure 20. Código Arduino Sketch variables.**

Se crea el objeto de AsyncWebServer en el puerto 80, se inicializan las variables estáticas del cálculo de los voltajes y se hace declara el uso del primer núcleo de la ESP32 junto a los pines de PWM para el control del motor, las variables de control y por último se inicializa constantes para el cálculo del flujo y los pines que se encargaran de las lecturas ADC de los sensores.

```

16 TaskHandle_t Tarea1;
17 //////////////////////////////////////////////////
18 const int sensor1=25; // Llave optica 1 //rojo
19 const int sensor2=26; // Llave optica 2 //verde
20 int cont=0;
21 int EstadoActual=0;
22 int EstadoAnterior=0;
23 //////////////////////////////////////////////////
24 #define RPWM 33 // Entrada RPWM del driver
25 #define LPWM 27 // Entrada LPWM del driver
26 #define PWM 32 // Entrada PWM_Enable del driver
27 //////////////////////////////////////////////////
28 float TI = 0; //tiempo inspiratorio
29 float TE = 0; //tiempo espiratorio
30 float TI_aux = 0; //tiempo inspiratorio auxiliar
31 float TE_aux = 0; //tiempo espiratorio auxiliar
32 float TP=0; // tiempo de pausa
33 //////////////////////////////////////////////////
34 float contVolumen = 0;
35 float contVolumen_aux=0;
36 float aux_cont = 3;
37 int volumen=0;
38 //////////////////////////////////////////////////
39 #define ADCFLUJO 36 //ESP32 pin GIOP3 (ADC0) connected to Venturi
40 #define ADCPRESION 39 // ESP32 pin GIOP36 (ADC3)
41 const float ADC_mV = 0.8056640625; // convesion multiplier from ESP32 ADC value to voltage in mV
42 const float SensorOffset = 200.0; // in mV taken from datasheet; 244mv medido con multmetro
43 const float sensitivity = 4.413; // in mV/mmH2O taken from datasheet
44 const float mmh20_cmH2O = 10; // divide by this figure to convert mmH2O to cmH2O
45 const float mmh20_kpa = 0.00981; // convesion multiplier from mmH2O to kPa

```

**Figure 21. Código Arduino Sketch Funciones.**

Se inicializa el puerto serial, se inicializan los pines, se inicializa que el primer núcleo va a trabajar independiente del segundo y el WiFi se conecta con las credenciales ingresadas.

```

48 void setup() {
49
50     Serial.begin(115200); //iniciar puerto serie a 115200 baudios}
51     ledcAttachPin(PWM, 0);
52     ledcSetup(0, 10000, 8);
53     pinMode(sensor1, INPUT);
54     pinMode(sensor2, INPUT);
55     pinMode(RPWM, OUTPUT);
56     pinMode(LPWM, OUTPUT);
57     xTaskCreatePinnedToCore(
58         loop_tarea1, /* Funcion de la tarea */
59         "Tarea1", /* Nombre de la tarea */
60         10000, /* Tamaño de la pila */
61         NULL, /* Parametros de entrada */
62         0, /* Prioridad de la tarea */
63         &Tarea1, /* objeto TaskHandle_t. */
64         0); /* Nucleo donde se correra */
65
66     setup_wifi();
67     client.setServer(mqtt_server, 1883);
68     client.setCallback(callback);
69 }

```

**Figure 22. Código Arduino Sketch Setup.**

Luego se manejan los request para la tarjeta ESP, cuando se recibe una solicitud en la URL se envía el texto HTML que se guarda en SPIFFS con el nombre index.html,

cuando se recibe una solicitud sobre las URL de voltaje o corriente se llaman a las funciones que devuelven las lecturas del módulo y finalmente se inicia el server.

```
// Ruta conexion con servidor / pagina web
server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html");
});
server.on("/voltage", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", voltage().c_str());
});
server.on("/current", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", current().c_str());
});
// Start server
server.begin();
}
```

**Figure 23. Código Arduino Sketch Setup.**

En el Void Loop hacemos la lectura constante de los sensores del encoder, garantizamos que la conexión con el MQTT este siempre funcionando, se evita el uso de delays mediante el método reservado de millis, se publica por medio del tópico 'esp32/dato' cualquier valor pertinente a mostrar, se lee constantemente los pasos para el correcto calculo del volumen producido en el Ambu y por ultimo tenemos una funcion que detiene al motor.

```
193 void loop(){
194
195     if (!client.connected() && digitalRead(sensor2)== HIGH){
196         reconnect();
197     }
198     client.loop();
199     long now =millis();
200     if(now - lastMsg > 1000){
201         lastMsg=now;
202         client.publish("esp32/dato","12");
203     }
204     while(cont != volumen)
205     {
206         lecturaEncoder();
207         motor_cw();
208     }
209     motor_stop();
210     while((digitalRead(sensor2))!=HIGH){
211         motor_ccw();
212     }
213     cont=0;
214 }
```

**Figure 24. Código Arduino Void Loop.**

## 6.2 HTML Script:

Este Script es utilizado con la finalidad de poder construir la aplicación móvil a través de una dirección web, en la cual visualizamos la información de los datos o valores generados por los dispositivos que conforman el Kit Horizon. Cabe destacar que una de sus ventajas es que se puede obtener esta información desde un ordenador o un móvil al ser una dirección web.

Inicialmente se necesita incluir la librería highcharts. Con la finalidad de poder dar a las gráficas una animación dentro de la dirección web.

```
<script src="https://code.highcharts.com/highcharts.js"></script>
```

**Figure 25. Código HTML Librería.**

Se requiere crear un <div> para cada gráfica con un único identificador. Este “div” se crea con el objetivo de generar secciones o agrupar contenidos. De igual manera se utiliza comúnmente en el desarrollo de gráficas o interfaces con propósitos estilísticos, en conjunto con los atributos style y class.

```
<body>
<h2>Respirador mecancio</h2>
<div id = "Chart Presion" class = "container"></div>
<div id = "Chart Flujo" class = "container"></div>
</body>
```

**Figure 26. Código HTML Instancias de cada gráfica.**

Se crean las gráficas, las cuales se configuran y se agrega la data en lenguaje javascript, incluyendo títulos, labels y axis.

```

var chartV = new Highcharts.Chart({
  chart:{ renderTo : 'chart-voltage' },
  title: { text: 'Voltage Graph' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
    dataLabels: { enabled: true }
  },
  series: { color: '#059e8a' }
},
  xAxis: { type: 'datetime',
  dateTimeLabelFormats: { second: '%H:%M:%S' }
},
  yAxis: {
    title: { text: 'Voltage' }
  },
  credits: { enabled: false }
});

```

**Figure 27. Código HTML Setup gráfica.**

Después en la función setInterval se actualizan los puntos de la gráfica, leyendo el request cada 500 milisegundos, y asignando el instante de tiempo en el eje horizontal y la variable corriente o voltaje en el vertical, se hace de la misma manera para ambas gráficas.

```

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var x = (new Date()).getTime(),
      y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartV.series[0].data.length > 40) {
        chartV.series[0].addPoint([x, y], true, true, true);
      } else {
        chartV.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/voltage", true);
  xhttp.send();
}, 500 );

```

**Figure 28. Código HTML Actualización gráfica.**

### 6.3 Python Script:

El código que se construyó en Python es el programa principal del proyecto, el cual desde el equipo recibe las lecturas de la Esp32 y las gráficas en tiempo real para su fácil lectura y análisis, entre otras funciones tenemos la habilidad de pausar o

reproducir la señal, presenta también la opción de entrar en un modo desarrollador donde puede variar las constantes de control y por último puede ingresar las variables de entrada para el arranque del ventilador mecánico.

## 6.4 GUI

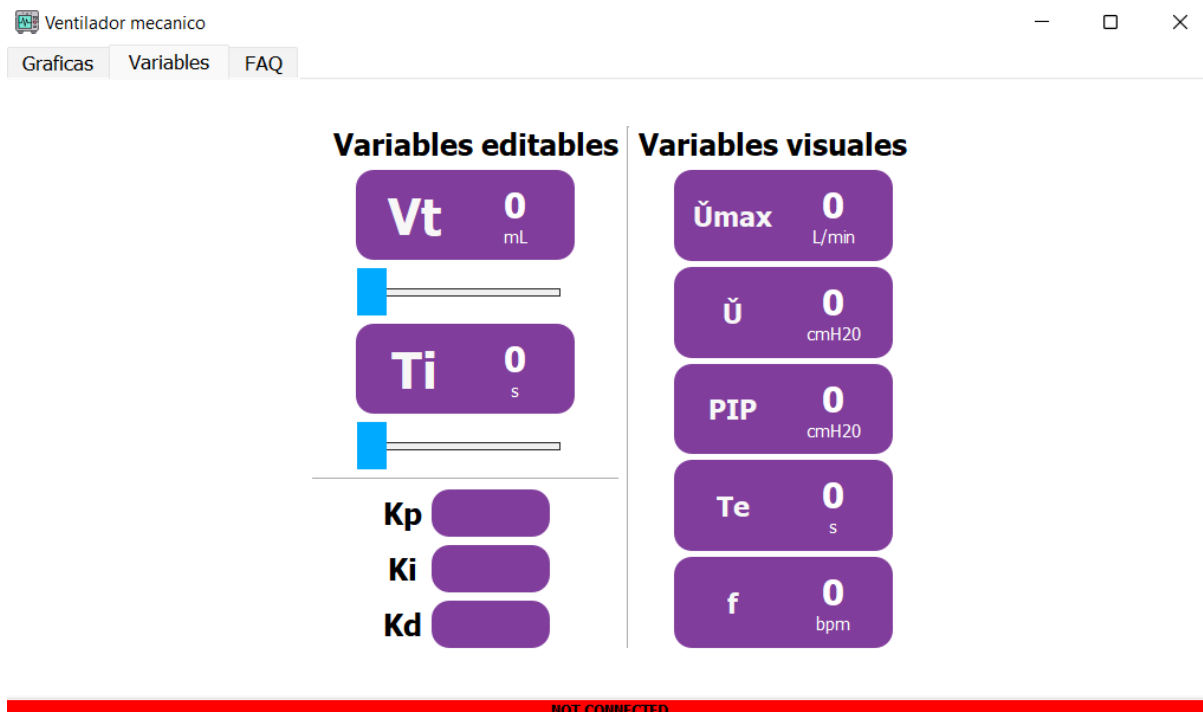
El software empleado para este desarrollo fue Qt Designer, un programa de fuente abierta para la creación de GUI's en Python.

La aplicación se divide en 3 páginas principales, La página de con la cual inicio el programa es la de gráficas, la del medio es la pestaña variables donde se puede ingresar las dos variables a controlar o variar las constantes del control y por ultimo tenemos la pestaña de FAQ por sus siglas en ingles de frequently asked questions donde se podrá encontrar una descripción del funcionamiento del programa y en la parte del fondo se encontrara una barra que puede cambiar de color, posee tres colores los cuales indican si esta o no conectado al servidor que habilita la comunicación entre la Raspberry y el Ventilador.



Figure 29. Gui Front QtDesigner Page Home.

En la pestaña variable podemos observar que las dos únicas variables necesarias para que el prototipo del ventilador inicie es el volumen tidal,  $V_t$  y el tiempo inspiratorio,  $T_i$ , estas dos variables se envían por medio de MQTT al microcontrolador quien se encarga del movimiento del motor. Para las constantes de control no es necesario igualarlas a algún valor ya que vienen por defecto, la modificación de estas se deja al criterio de uso.



**Figure 30. Gui Front QtDesigner Page Graph.**

## 6.7 MAIN CODE

El módulo principal del programa se realizan todos los procesos que lo componen, la lectura de las variables de datos de la ESP32, el motor visual de la aplicación y las funciones que posee, los cuales se presentarán en segmentos posteriores, primero se traen los módulos necesarios, luego se ejecuta el objeto aplicación como instancia de PyQt5, se define un mainprogram class y es inicializado. Allí se recibe el código GUI que contiene los elementos visuales en la GUI y el pyqtgraph que posee la configuración de la aplicación y la gráfica.

Se crea el hilo de lectura del puerto para la comunicación MQTT que va estar en loop con la función update graph mientras se esté corriendo la gráfica, y a su vez se fija la pestaña inicial de la GUI y se espera instrucción del usuario. Desde update graph se generan las alertas de error si oportunas y se hace el guardado de la información para proseguir con su envío en forma de reporte.

Inicialmente se incluyen los siguientes módulos externos requeridos para el funcionamiento del programa, conformados por: PyQt5 incorporando QWidgets para la generación de las gráficas, QThread y pyqt5signal para la lectura de las variables en paralelo con la actualización del widget que es el programa en sí, también PyQt5.uic para la lectura del archivo .ui.

Aparte se incluyen los módulos “time”, “datetime”, “numpy”, “random”, “paho”, para cumplir las demás tareas.



```

from statistics import variance
import string
import sys, time, os, traceback
from tkinter import Variable
from random import randint
import numpy as np

from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg

from PyQt5.QtWidgets import QApplication, QMainWindow
from PyQt5.uic import loadUi
from PyQt5 import QtGui
from PyQt5 import QtCore
from PyQt5.QtCore import QThread, pyqtSignal, pyqtSlot

from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt

from datetime import date

```

**Figure 31. Código Python Main code Importación de módulos.**

A continuación, se encuentra la clase WorkerThread, la cual permite correr la lectura de los puertos en un hilo diferente al loop principal que corresponde a la actualización de la interfaz, en este hilo extra se selecciona los tópicos de las cuales el código siempre estará leyendo y se guarda en una instancia denominada “ser”.

```

class WorkerThread(QThread):

    signal_input = pyqtSignal(str)

    def __init__(self, parent=None):
        QThread.__init__(self)

    def run(self):
        while(1):
            try:
                client = self.connect_mqtt()
                self.subscribe(client)
                client.loop_forever()
            except os.error:
                #print("not connected to network")
                self.signal_input.emit("1")

    def connect_mqtt(self):
        def on_connect(client, userdata, flags, rc):
            if rc == 0:
                self.signal_input.emit("0")
                #print("Connected to MQTT Broker!")

```

**Figure 32. Código Python Main código creación de hilos.**

Antes de iniciar la conexión al servidor se deja declarado los tópicos y se pone en variables la dirección IP donde debe conectarse, como se a mencionado anteriormente en los tópicos podemos encontrar las variables a leer de los sensores como las constantes del controlador si se desean modificar, en este caso el bróker, la raspberry va a cumplir con una función bidireccional de suscripción y publicación de los tópicos.

```

broker = '10.42.0.1'          JulianDuque97,
port = 1883
topic_1 = "esp32/presion"
topic_2 = "esp32/flujo"
topic_3 = 'esp32/volumen'
topic_4 = 'esp32/ti'
topic_5 = 'esp32/volumen_tidal'
topic_6 = 'esp32/setpoint'
topic_7 = 'esp32/kp'
topic_8 = 'esp32/ki'
topic_9 = 'esp32/kd'
client_id = 'Raspberry'
client = mqtt_client.Client(client_id)

```

**Figure 33. Código Python Main código creación de hilos.**

```

def subscribe(self,client: mqtt_client):
    def on_message(client_userdata, msg):
        if msg.topic == (parameter) msg: Any
            mssg = str(msg.payload.decode())+'p'
            self.signal_input.emit(mssg)
            #print(mssg)

        if msg.topic == 'esp32/flujo':
            mssg = str(msg.payload.decode())+'f'
            self.signal_input.emit(mssg)
            #print(mssg)

```

**Figure 34. Código Python Main código creación de hilos**

A continuación, se encuentra la clase Mainprogram donde en la función **Init** se configura el inicio del programa, incluyendo la carga del archivo UI, declaración de variables, conexiones a funciones, inicialización de objetos y parámetros visuales.

```

class Mainprogram(QMainWindow):
    def __init__(self): ...

```

**Figure 35. Código Python Main code Configuración inicial.**

La función **graph\_properties** contiene la parametrización de la gráfica que se está mostrando donde solo dos parámetros varían entre equipos, el color de ciertas características y la escala en el tiempo.

```
def graph_properties(self, linecolor, xmax): ...
```

Figure 36. Código Python Main code function graph properties.

La función **value** funciona como puente entre la adquisición de las variables y su visualización como parte de la gráfica, donde se reciben la variable 1, variable 2 con estos datos se actualiza el siguiente frame de la gráfica en ejecución.

```
def value(self, value1, value2, port_en): ...
```

Figure 37. Código Python Main code function value.

La función **update graph** actualiza la gráfica que se está mostrando, agregando el nuevo dato recibido a la lista de datos que se están cargando con la tasa de adquisición de la ESP32. Luego se recrea la gráfica con la nueva lista.

Como parte de la función de actualizar la gráfica en esta función también se encuentra una generación de alertas en la parte inferior de la pantalla ante novedades que se presenten como un puerto desconectado.

```
def update_graph(self, port_en): ...
```

Figure 38. Código Python Main code function update\_graph.

La función **reboot\_graph** se utiliza para resetear la gráfica borrando los datos actuales y restableciendo sus características, la cual es usada en ciertas funciones del programa.

```
def reboot_graph(self): ...
```

Figure 39. Código Python Main code función reboot\_graph.

La función **monitor\_mode** es una función de backend para el manejo de algunos procesos de back necesarios en el programa y la aparición de algunos frames entre el cambio de secciones en la interfaz

```
def monitor_mode(self, state, page): ...
```

**Figure 40. Código Python Main code función monitor\_mode.**

La función **actualPage** define las características visuales y de funcionamiento de las 3 gráficas existentes (Presion, Flujo y Volumen), hace visibles las características necesarias en cada instancia y selecciona la gráfica escogida.

```
def actualPage(self, page): ...
```

**Figure 41. Código Python Main code función actualPage.**

La función **ePrompt** es la que se encarga del proceso de generar un aviso por error o alerta en la visual de la interfaz, parametrizando el tipo de alerta y el texto.

```
def ePrompt(self, error, bcolor): ...
```

**Figure 42. Código Python Main code función ePrompt.**

La función **menu\_buttons** contiene el proceso de inicio y pausa de la gráfica recibiendo respuesta del botón de play, incluyendo la captura de tiempo en funcionamiento.

```
def menu_buttons(self, play):|...
```

**Figure 43. Código Python Main code función menu\_buttons.**

Una vez se corre el programa en el orden que se muestra, se crea la aplicación como instancia de Qapplication luego la ventana con Mainprogram. Una vez se completa el init de la clase se muestra en pantalla la ventana, y quedan corriendo los 2 hilos de PyQt5, el de actualización de la visual y el creado para la adquisición de datos.

```
app = QApplication([])
window = Mainprogram()
window.show()
app.exec_()
```

**Figure 44. Código Python Main code función principal.**

## 7. RESULTADOS Y ANÁLISIS

### 7.1 RESULTADOS

#### 7.1.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL PROTOTIPO DE UN VENTILADOR MECANICO

Con el sistema de instrumentación construido se realizaron pruebas con el ventilador mecánico, al implementar el prototipo del circuito diseñado conectándolo a un a la raspberry para hacer la alimentación y teniendo una red Wifi estable presente el diseño se conecta de manera efectiva al puerto serial transmitiendo los datos que recibe del sensor y a la red local para la aplicación móvil generando la ip que se utilizara desde el receptor.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8
192.168.20.33
0.00±0.30
0.00±0.30
```

Figure 45. Monitor Serial Arduino IDE, IP ESP32.

El diseño final de la interfaz de usuario que se desarrolló permite escoger la señal que se desee analizar, personalizando la experiencia para los experimentos que se quieran hacer con dicho equipo, tal selección se realiza desde el inicio del programa y abre todas las características que la aplicación ofrece para empezar a trabajar.



Figure 46. Interfaz de usuario diseño final página de inicio.

Al iniciar el programa podemos observar que somos recibidos por todas las variables que se calculan y se censan al tiempo mas 5 botones cada uno con una funcion especifican, tenemos los tres botones de la parte superior que nos permite cambiar de vista y luego tenemos en la parte inferior dos botones uno que corresponde a Play y Stop, este lo que hace como su nombre lo indica es que cuando el programa esta en funcionamiento tiene la capacidad de detener la grafica en un punto deseado y al tiempo detiene todo el sistema y por ultimo tenemos el botón inferior derecho que nos permite como ya mencionamos cambiar de señal

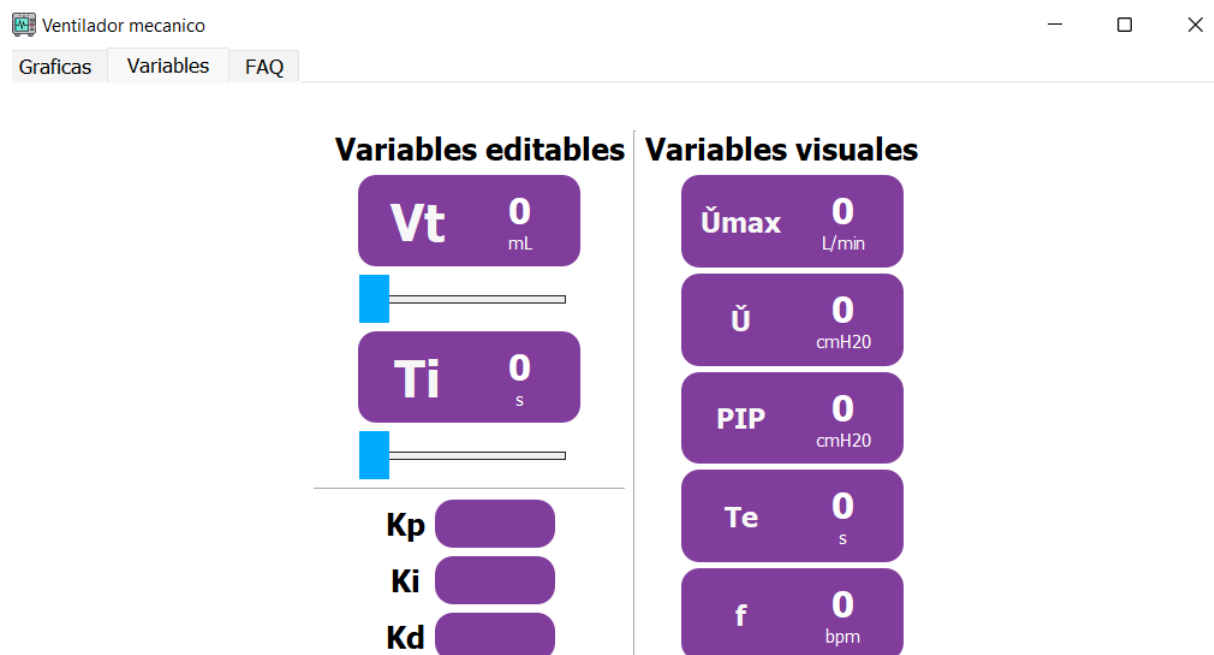


Figure 47. Monitor Python muestreo.

Para utilizar la aplicación móvil se utiliza la dirección IP que corresponde a la tarjeta la cual se puede ver en la respuesta vía serial al inicializar la ESP32 como en la figura 45, se escribe en el navegador de cualquier equipo conectado a la red y se podrán visualizar las gráficas en tiempo real, como en el caso realizado en la figura 67 correspondiente a la turbina de viento siendo activada manualmente.



**Figure 48. HTML aplicación en red local.**



## 7.2 ANÁLISIS

### 7.2.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL PROTOTIPO DE VENTILADOR MECANICO

Para la identificación de los parámetros eléctricos de los sensores se consultó la hoja de características y se hicieron las mediciones con multímetro y con el VTPlus que el Hospital tenía, de los valores encontrados se puede concluir que el sensor que se requiere para la adquisición de las mediciones debe tener un rango de medición de voltaje de 0 a mínimo 5V y la corriente de 0 a 100mA.

Para un mayor entendimiento de los procesos que comprenden cada uno de los equipos en su funcionamiento se realizó la consulta de los principios eléctricos y biomédicos detrás de cada elemento, de lo cual se encontró que la generación a esta escala, se espera con mucho ruido por lo cual se necesita una buena resolución en la medición de los sensores, y no se necesita de alguna construcción especial para que el sensor trabaje junto a los equipos.

Para la construcción del software y hardware del sistema de instrumentación virtual los requisitos que se plantearon como una necesidad para cumplir con el requerimiento del proyecto fueron: funcionar en tiempo real, contener material educativo interactivo dentro de la aplicación, la capacidad de guardar datos base y tener una manera de monitorear los datos de forma remota. Para cumplir estas, se utilizó Python el cual es de fuente abierta lo que permitió manejar el programa en tiempo real, donde las posibilidades de personalización en la creación de aplicaciones permite una creatividad plena lo cual facilitó la creación de una documentación y experimentos, además el amplio catálogo de módulos de fuente abierta encontrados en la red de Python permiten añadir funciones al programa.

Para el acceso remoto se había pensado inicialmente tener un servidor en la nube que permite total libertad en el monitoreo desde cualquier lugar, sin embargo, con las complicaciones que representa un servidor siempre activo y el constante seguimiento que requiere, se optó por una solución más local ajustándose mejor al funcionamiento del proyecto permitiendo revisar los datos en una red local, lo cual se implementó y ha dado buenos resultados.

La tarjeta que se eligió para el sistema de instrumentación fue la ESP32, la cual cumple con las características requeridas, incluyendo una alta velocidad de procesamiento, una buena capacidad de memoria lo cual evitará saturación en la lectura de los datos, además cuenta con una tarjeta de Wifi 2.4. La ESP32 funciona a 3.3V y se programó en lenguaje de Arduino IDE. Viendo el desempeño del sistema construido con esta tarjeta se puede recomendar esta para trabajos similares, destacando también su versatilidad, simpleza y robustez.

Para la interfaz de usuario que se haría en el computador, basado en experiencias previas y con los objetivos que se buscaban cumplir, destacando la necesidad de que utilice un software de uso libre, y que esté enfocado a desarrollos en el área de la ingeniería, se desarrolló el programa en Python, durante el proceso de construcción surgieron diversos retos, el primero fue la necesidad de correr dos procesos de forma paralela, lo cual se logró manejando threading en el flujo del código, lo que hace referencia a ejecutar procesos alternos durante tiempos de espera del proceso principal, y así aprovechar el tiempo más eficientemente. También se presentó un reto en la limitación que posee el intérprete de Python en el procesamiento, se identificó que a medida que se añadían recursos en la interfaz y animación interactiva con el usuario se reducía el rendimiento del programa lo que llevó a la necesidad de optimizar las funciones y alcanzar un equilibrio en la cantidad de elementos que podíamos manejar manteniendo un buen rendimiento en lo principal de la GUI que es la gráfica funcionando, finalmente se determinó que en la parte visual que posee la interfaz se logró un diseño moderno agradable y de fácil uso para facilitar la aceptación del programa por parte del usuario. Al terminar el código completo que compone la interfaz, se pudo determinar que la solución que se dio por medio de Python dio solución efectiva al objetivo sin embargo la portabilidad del programa no es la ideal, representado en errores que se pueden presentar ejecutando el programa en otros equipos.

## 8. CONCLUSIONES

- Para el correcto funcionamiento del sistema desarrollado se debe tener en cuenta la velocidad de comunicación que posee la tarjeta programable para transmitir de manera efectiva los datos al sistema de instrumentación virtual sin llegar a saturarse.
- El desarrollo y construcción del prototipo de instrumentación virtual para el prototipo de ventilador mecánico permite la visualización al usuario de las diferentes variables con las que cuenta un sistema como este, además del manejo que se le puede dar a la información adquirida por el sistema de los diferentes parámetros generados por cada uno de los elementos que componen el ventilador mecánico, y así darle tratamiento a los datos para la toma de decisiones en un posible proyecto a mayor escala.
- Al momento de adquirir los diferentes elementos a utilizar dentro del sistema se debe atender muy bien cada una de las características eléctricas y electrónicas en las que funcionan de manera óptima sin llegar a sufrir algún tipo de afectación.
- Las herramientas de desarrollo incorporadas dentro del ambiente Python para la construcción de interfaces de usuario, como lo son: PyQt5 y Qt Designer. Poseen una gran capacidad y versatilidad en recursos para dar solución a las necesidades que se presentaron durante la construcción del proyecto. Cabe resaltar que son de uso abierto.
- El software Python es un ambiente de desarrollo de programación que en la actualidad cuenta con una gran gama de información en la internet, esto la hace muy amigable al usuario ya que permite una gran facilidad de corrección de errores o inconvenientes que se pueden presentar durante el avance de un proyecto determinado.
- El diseño del prototipo de ventilador mecánico es una herramienta versátil a la hora de conocer el funcionamiento de las diferentes nuevas tecnologías que existen en la actualidad para hacer un aproximamiento a un ventilador mecánico de bajo costo de una manera fácil y amigable con el operador final.
- La tarjeta de programación ESP32 cuenta con una gran gama de características como lo son: WiFi, bluetooth, excelente calidad de comunicación en diferentes protocolos, excelente capacidad de memoria, una

muy buena capacidad en bits dentro del DAC y la versatilidad que posee en diferentes lenguajes de programación para escribir sobre ella.

- Cabe destacar la gran importancia que tiene el seguir enfocando proyectos que estén dirigidos hacia la transmisión de información y conocimiento de cómo el mundo ha venido evolucionando en temas tan importantes como lo son la tecnología en la generación de prototipos que su producción sean de bajo costo.
- Contar con una estructura organizada de tal manera que permita un flujo correcto dentro del ambiente de programación. Es algo que facilitará la construcción y entendimiento del código en general y la adición de posibles modificaciones o secciones que se necesiten en algún momento dado.

## 9. REFERENCIAS

- [1] Luis Enrique FAUROUX, Jorge Esteban ETEROVIC (1), Omar Jorge DEGAETANI (1), "THE VENTURI EFFECT AND ITS INCIDENCE IN THE PERFORMANCE OF MICRO-HYDRAULIC TURBINES," *Departamento de Ingeniería e Investigaciones Tecnológicas*.
- [2] M. YEPES, "Desafíos en ciencia, tecnología e innovación en tiempos de coronavirus," CORDOBA, 2020.
- [3] A. Patricia, G. Torres, J. Edgar, C. Montaña, and P. Rocha, "SIMVENT," pp. 1271–1277, 2021.
- [4] C. Quiñones and M. Bernal, "Diseño de un ventilador mecánico de bajo costo para el tratamiento de pacientes con insuficiencia respiratoria," 2011.
- [5] P. Jissette and R. Sanchez, "Diseño de Ventilador Mecánico Inteligente para pacientes con COVID-19 en UCI de hospitales," no. May, 209.
- [6] Oscar Heredia, Xiomara Chunga, Lewis De La Cruz, Mirko Zimic, "Diseño y evaluación de un ventilador mecánico," Unidad de Bioinformática y Biología Molecular.
- [7] FLOR, Omar C, SUAREZ, Franyelit M, "Evaluaciones matemáticas de los mecanismos para ventiladores artificiales emergentes," pp. 1271–1277.
- [8] <http://fcbi.unillanos.edu.co/fcbi/ci/>
- [9] T. Final and D. E. G. En, "IARespira: experiencias en el diseño y desarrollo de un Ventilador Mecánico No Invasivo para COVID-19," 201.
- [10] V. Alfonso and B. Ballesteros, "Low Cost Volume Sensors for Mechanical Ventilators" no. 2019, pp. 388–397.
- [11] A. Q. Ramírez and J. J. Páez, "Automatización y control de redes de distribución de agua"
- [12] Giovanni García-Castro , David Latorre-Galeano "PROTOCOLO DE PRUEBA DE VENTILADOR MECÁNICO PARA ATENCIÓN DE PACIENTES CON COVID-19 EN MODELO PORCINO" 206.
- [13] V. A. Ballesteros-ballesteros and A. P. Gallego-torres, "Prototipo de ventilador mecánico de emergencia tipo presión Portal de revistas: <http://revistas.utp.ac.pa> positivo intermitente en respuesta a la pandemia del COVID-19," vol. 28, no. 52, pp. 0–2, 201.
- [14] A. Patricia and G. Torres, "Reespirator: Prueba de concepto y validación preclínica de un nuevo modelo de ventilador mecánico," no. 21, pp. 111–120, 204.
- [15] N. A. Ávila and V. T. Gómez, "Respirador Mecánico VENT19," 2020.
- [16] V. H. Gómez et al., "Ventilador mecánico no invasivo para atención domiciliaria" pp. 17–23.
- [17] C. Guillermo, A. Aldana, O. Leonardo, and F. Ramos, "Since January 2020 Elsevier has created a COVID-19 resource centre with free information in

English and Mandarin on the novel coronavirus COVID- 19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.,” 208.

- [18] C. Alexander and C. Cerquera, “Diseño e Implementación de Prototipo de ventilador mecánico de bajo costo utilizando una placa programable y software libre para monitoreo de parámetros mínimos necesarios para dar soporte vital a pacientes con deficiencia respiratoria aguda aplicando Internet de las Cosas (IoT)” 2021.
- [19] S. C. Castillo, J. Sebastián, R. Aguilar, and E. Francisco, “Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated.”
- [20] L. Cilleruelo, “DIVISOR DE TENSION INDUCTIVO AUTOCALIBRADO DE ALTA PRECISION” pp. 1–18, 2014.
- [21] <http://www.16deabril.sld.cu/rev/246/AO/7-ventilacion%20mecanica.html>.
- [22] <https://programarfacil.com/blog/arduino-blog/sensor-ina219-arduino-esp8266-esp32/>
- [23] <https://programarfacil.com/esp8266/esp32/>
- [24] <https://www.becas-santander.com/es/blog/python-que-es.html>
- [25] <https://realpython.com/qt-designer-python/>
- [26] <https://unipython.com/pyqt5-interfaces-graficas-con-python/>