

**DISENO 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**

**JULIAN ARMANDO DUQUE ALAYON**

**UNIVERSIDAD DE LOS LLANOS**

**FACULTAD DE CIENCIAS BASICAS E INGENIERIA**

**ESCUELA DE INGENIERÍAS**

**PROGRAMA DE INGENIERÍA ELECTRÓNICA**

**VILLAVICENCIO, COLOMBIA**

**2022**

**DISENO 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**

**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

Asesora:

Ing. Bladimir Pineda

**UNIVERSIDAD DE LOS LLANOS**

**FACULTAD DE CIENCIAS BASICAS E INGENIERIA**

**ESCUELA DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA ELECTRÓNICA**

**VILLAVICENCIO, COLOMBIA**

**2022**

# **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**

# 

[**DEDICATORIA** 2](#_Toc94116042)

[**AGRADECIMIENTOS** 3](#_Toc94116043)

[**TABLA DE CONTENIDO** 5](#_Toc94116044)

[LISTA DE FIGURAS 6](#_Toc94116045)

[**RESUMEN** 7](#_Toc94116046)

[**ABSTRACT** 8](#_Toc94116047)

[**1. INTRODUCCIÓN** 9](#_Toc94116048)

[**2. OBJETIVOS** 11](#_Toc94116049)

[**3. JUSTIFICACIÓN** 12](#_Toc94116050)

[**4. METODOLOGÍA** 13](#_Toc94116051)

[**5. MARCO REFERENCIAL** 14](#_Toc94116052)

[**5.1 MARCO CONTEXTUAL** 14](#_Toc94116053)

[**5.2 ESTADO DEL ARTE** 15](#_Toc94116054)

[**5.3 MARCO TEÓRICO** 23](#_Toc94116055)

[**5.3.1.1 INA219** 23](#_Toc94116056)

[**5.3.1.2 Módulo sensor INA219** 25](#_Toc94116057)

[**5.3.1.3 Conexión fuente de alimentación** 25](#_Toc94116058)

[**5.3.1.4 Resistencia shunt** 26](#_Toc94116059)

[**5.3.1.5 Pines sensor INA219** 27](#_Toc94116060)

[**5.3.2.1 ESP32** 27](#_Toc94116061)

[**5.3.2.2 CONECTIVIDAD INALÁMBRICA** 28](#_Toc94116062)

[**5.3.2.3 CONVERSOR ANALÓGICO DIGITAL** 29](#_Toc94116063)

[**5.3.2.4 CONVERSOR DIGITAL ANALÓGICO** 29](#_Toc94116064)

[**5.3.2.5 COMUNICACIÓN I2C** 30](#_Toc94116065)

[**5.3.3 PYTHON** 30](#_Toc94116066)

[**5.3.4 QT DESIGNER** 31](#_Toc94116067)

[**5.3.5 PYQT5** 32](#_Toc94116068)

[**5.3.6 KIT DE CIENCIA DE ENERGÍAS RENOVABLES 2.2 HORIZON FCJJ-37** 32](#_Toc94116069)

[34](#_Toc94116070)

[**6. DESARROLLO DEL PROYECTO** 34](#_Toc94116071)

[**6.2 HTML Script:** 40](#_Toc94116072)

[**6.3 Python Script:** 42](#_Toc94116073)

[**6.4 MplWidget Code** 43](#_Toc94116074)

[**6.5 GUI Código** 44](#_Toc94116075)

[**6.6 PAGE\_CONFIG CÓDIGO** 48](#_Toc94116076)

[**6.7 MAIN CODE** 48](#_Toc94116077)

[**7. RESULTADOS Y ANÁLISIS** 56](#_Toc94116078)

[**7.1 RESULTADOS** 56](#_Toc94116079)

[**7.1.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL KIT HORIZON** 56](#_Toc94116080)

[**7.1.2 SOCIALIZACIÓN DEL PROYECTO A ESTUDIANTES DE LA INSTITUCIÓN EDUCATIVA SAN ANTONIO DEL ARIARI** 61](#_Toc94116081)

[**7.2 ANÁLISIS** 63](#_Toc94116082)

[**7.2.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL KIT HORIZON** 63](#_Toc94116083)

[**7.2.2 SOCIALIZACIÓN DEL PROYECTO A ESTUDIANTES DE LA INSTITUCIÓN EDUCATIVA SAN ANTONIO DEL ARIARI** 64](#_Toc94116084)

[**8. CONCLUSIONES** 65](#_Toc94116085)

[**9. REFERENCIAS** 67](#_Toc94116086)

# 

LISTA DE FIGURAS

[**Figura 1 Esquema eléctrico base del INA 219 [39] 25**](#_Toc94119092)

[**Figura 2. *Sensor INA219* [39] 26**](#_Toc94119093)

[**Figura 3. *Partes del módulo INA219.* [39] 26**](#_Toc94119094)

[**Figura 4. Conexión a fuente de alimentación INA219. [39] 27**](#_Toc94119095)

[**Figura 5. Pines de conexión INA219. [39]** 28](#_Toc94119096)

[**Figura 6. *Tarjeta programable ESP32.* [40]** 29](#_Toc94119097)

[**Figura 7. Esquema de la conectividad inalámbrica de la ESP32. [40]** 30](#_Toc94119098)

[**Figura 8. Señal DAC. [40]** 31](#_Toc94119099)

[**Figura 9. Logo de Python. [41]** 32](#_Toc94119100)

[**Figura 10. *Logo de QT* [42]** 33](#_Toc94119101)

[**Figura 11. KIT HORIZON MODELO FCJJ-37. [44]** 34](#_Toc94119102)

[**Figura 12. *Experimento 5.* [44]** 35](#_Toc94119103)

[**Figura 13. *Experimento 2.* [44]** 35](#_Toc94119104)

[**Figura 14. INA219 módulo [39]** 36](#_Toc94119105)

[**Figura 15. tarjeta ESP32 DEVKIT V1. [40]** 36](#_Toc94119106)

[**Figura 16. Circuito de adquisición de datos.** 37](#_Toc94119107)

[**Figura 17*.* *Montaje sistema de captura de datos.*** 37](#_Toc94119108)

[**Figura 18. Diagrama estructura organizacional ESP32.** 38](#_Toc94119109)

[**Figura 19. Código Arduino Sketch Librerías.** 38](#_Toc94119110)

[**Figura 20. Código Arduino Sketch variables.** 39](#_Toc94119111)

[**Figura 21*. Código Arduino Sketch Funciones.*** 39](#_Toc94119112)

[**Figura 22. *Código Arduino Sketch Setup.*** 40](#_Toc94119113)

[**Figura 23*. Código Arduino Sketch Setup.*** 40](#_Toc94119114)

[**Figura 24. Código Arduino Void Loop.** 41](#_Toc94119115)

[**Figura 25. Código HTML Librería.** 41](#_Toc94119116)

[**Figura 26. Código HTML Instancias de cada gráfica.** 42](#_Toc94119117)

[**Figura 27. Código HTML Setup gráfica.** 42](#_Toc94119118)

[**Figura 28. Código HTML Actualización gráfica.** 43](#_Toc94119119)

[**Figura 29. Diagrama estructura código en python.** 44](#_Toc94119120)

[**Figura 30. Código Python MplWidget configuration figura para las gráficas.** 44](#_Toc94119121)

[**Figura 31. *Gui Front QtDesigner Page Home.*** 45](#_Toc94119122)

[**Figura 32. *Gui Front QtDesigner Page Graph.*** 46](#_Toc94119123)

[**Figura 33. *Gui Front QtDesigner Page Doc.*** 47](#_Toc94119124)

[**Figura 34. *Gui Front QtDesigner Sección Lateral.* 48**](#_Toc94119125)

[**Figura 35. *Código Front archivo en formato .ui* 48**](#_Toc94119126)

[**Figura 36. *Código Python page\_config selector de páginas.*** 49](#_Toc94119127)

[**Figura 37. *Diagrama de flujo Main.py.*** 50](#_Toc94119128)

[**Figura 38. *Código Python Main code Importación de módulos.*** 51](#_Toc94119129)

[**Figura 39. *Código Python Main código creación de hilos, identificación de puerto serial.*** 51](#_Toc94119130)

[**Figura 40. *Código Python Main código creación de hilos, lectura del puerto serial.*** 52](#_Toc94119131)

[**Figura 41. Código Python Main código creación de hilos, error en puerto.** 52](#_Toc94119132)

[**Figura 42. Código Python Main code Configuración inicial.** 52](#_Toc94119133)

[**Figura 43. Código Python Main code function graph properties.** 52](#_Toc94119134)

[**Figura 44. Código Python Main code function value.** 53](#_Toc94119135)

[**Figura 45. *Código Python Main code function update\_graph.*** 53](#_Toc94119136)

[**Figura 46. *Código Python Main code función reboot\_graph.*** 53](#_Toc94119137)

[**Figura 47. *Código Python Main code funcion osc\_volt y osc\_range.*** 54](#_Toc94119138)

[**Figura 48. Código Python Main code función adjust.** 54](#_Toc94119139)

[**Figura 49. Código Python Main code función Rightmenu** 54](#_Toc94119140)

[**Figura 50. Código Python Main code función monitor\_mode.** 54](#_Toc94119141)

[**Figura 51. Código Python Main code función actualPage.** 54](#_Toc94119142)

[**Figura 52. Código Python Main code función ePrompt.** 55](#_Toc94119143)

[**Figura 53. *Código Python Main code función customBar.*** 55](#_Toc94119144)

[**Figura 54. Código Python Main code función menu\_buttons.** 55](#_Toc94119145)

[**Figura 55. Código Python Main código función auto\_save.** 55](#_Toc94119146)

[**Figura 56. *Código Python Main code función save\_data.*** 55](#_Toc94119147)

[**Figura 57. *Código Python Main code función load\_combobox.*** 56](#_Toc94119148)

[**Figura 58. *Código Python Main code función table.*** 56](#_Toc94119149)

[**Figura 59. Código Python Main code función load\_data.** 56](#_Toc94119150)

[**Figura 60. Código Python Main code función sendmail.** 56](#_Toc94119151)

[**Figura 61. *Código Python Main code función principal.*** 57](#_Toc94119152)

[**Figura 62. *Monitor Serial Arduino IDE, IP ESP32.*** 57](#_Toc94119153)

[**Figura 63. *Interfaz de usuario diseño final página de inicio.*** 58](#_Toc94119154)

[**Figura 64. *Monitor Python muestreo Panel solar.*** 58](#_Toc94119155)

[**Figura 65. *Registro de captura de datos en archivo xlsx.*** 59](#_Toc94119156)

[**Figura 66. Reporte vía correo electrónico de una captura de datos.** 60](#_Toc94119157)

[**Figura 67. *Gráfica voltaje del panel solar en el tiempo, respuesta escalón.*** 60](#_Toc94119158)

[**Figura 68. *Gráfica voltaje de la turbina de viento en el tiempo, respuesta a alteraciones manuales.*** 61](#_Toc94119159)

[**Figura 69. HTML aplicación en red local*.*** 62](#_Toc94119160)

[**Figura 70. Socialización del proyecto a los estudiantes de la institución educativa San Antonio del Ariari.** 63](#_Toc94119161)

# **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 interese 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 y hardware con el fin de lograr sencilles, 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 “DISENO 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”, 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 HTLM5, 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 finales.

# **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**

## **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.

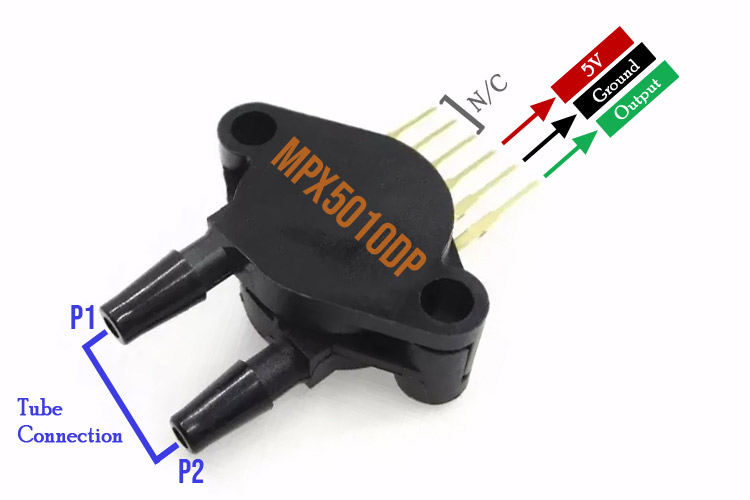


Figura 1 Sensor MPX5010DP [39]

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.

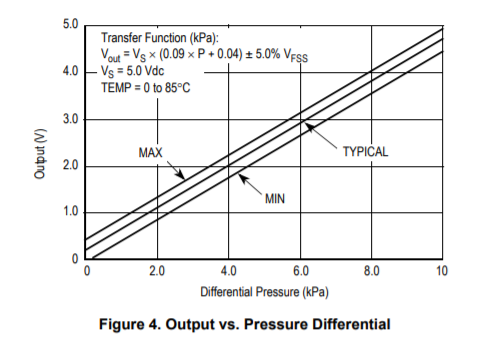


Figura 2. *Salida vs presión diferencial* [39]

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.

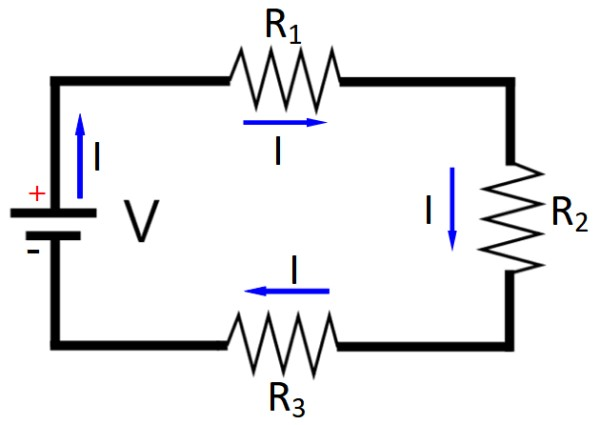


Figura 3. *Divisor de voltaje resistivo de varias resistencias en serie* [40]

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.

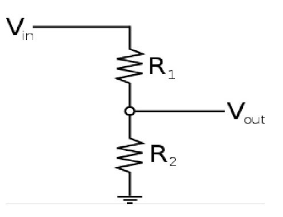


Figura 4. Divisor de voltaje de dos resistencias. [40]

#### **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, 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.



Figura 5. Conexión a fuente de alimentación MPX5010DP. [39]

Aunque en el esquema he utilizado una [fuente de alimentación regulable de laboratorio](https://amzn.to/3fCAqiI), se puede utilizar una batería o cualquier otra fuente de alimentación.

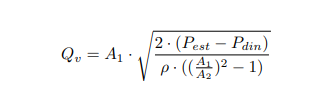
Lo importante es que el pin 3 del sensor esté conectado al positivo de la fuente, el pin 2 se conecta tierra de la fuente. Por último, el terminal de la señal directo a la entrada del divisor de voltaje ya que este se encargará de bajar el voltaje para que el microcontrolador no sufra averías.

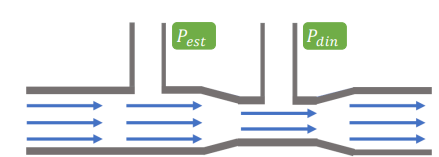
#### **5.3.1.4 Tubo Venturi**

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.





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

### **5.3.2.1 ESP32**

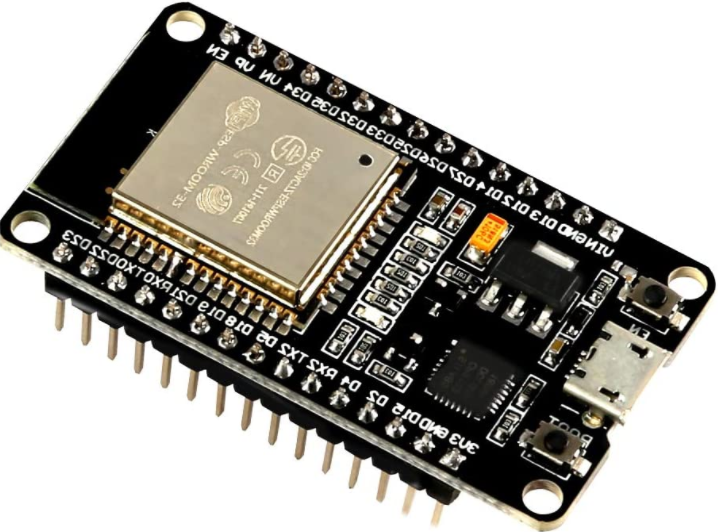
ESP32 es una serie de SoC (por sus siglas en inglés, *System on Chip*) y módulos de bajo costo y bajo consumo de energía creado por [*Espressif Systems*](https://www.espressif.com/).

Esta nueva familia es la sucesora del conocido [ESP8266](https://programarfacil.com/esp8266/proyectos-con-esp8266-iot/) 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.



**Figura 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).

****

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

#### **5.3.2.3 CONVERSOR 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 conversor 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](https://programarfacil.com/blog/arduino-blog/pwm-con-arduino-analogico/). El ESP32 cuenta con dos conversores digital analógico.

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.



**Figura 9. Señal DAC. [40]**

#### **5.3.2.5 COMUNICACIÓN I2C**

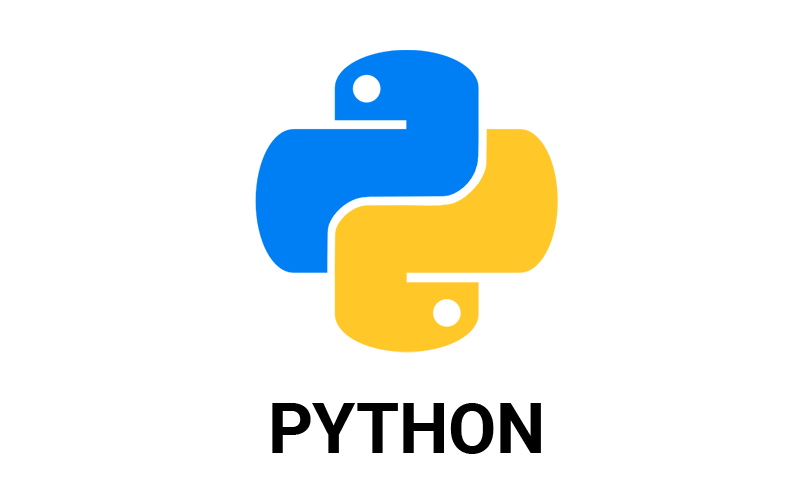
Los ESP32 cuentan con dos interfaces [I2C o TWI](https://programarfacil.com/blog/arduino-blog/comunicacion-i2c-con-arduino/) que admiten los modos de operación maestro y esclavo. Entre sus características destacan:

* Modo estándar (100 Kbits/s).
* Modo rápido (400 Kbits/s)
* Direccionamiento de 7 y 10 bits. [40]

### **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]



**Figura 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]



**Figura 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 MECANCIO**

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.



**Figura 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.

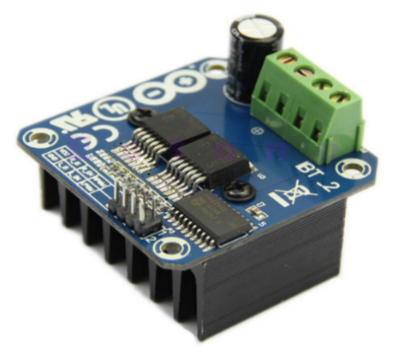


**Figura 13. *Prototipo inicial del respirador mecánico.***

# **6. DESARROLLO DEL PROYECTO**

Para cumplir con el objetivo general del proyecto se inicio 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.

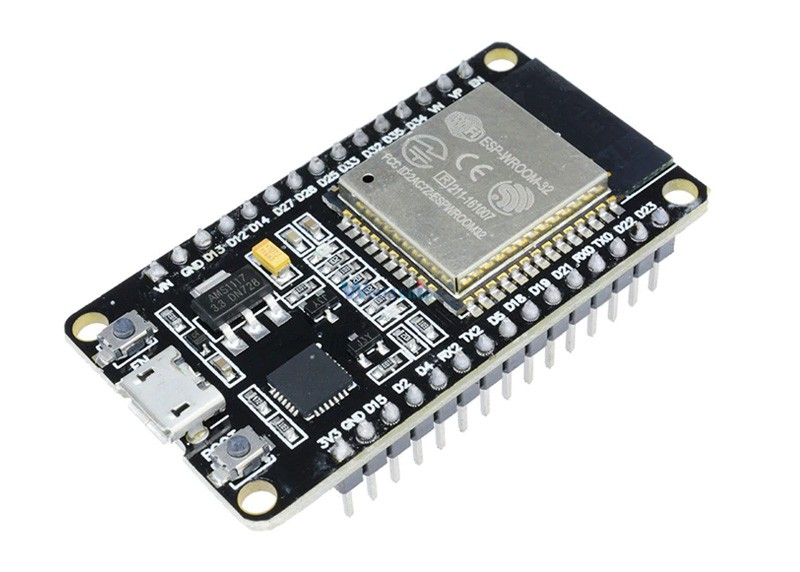
El circuito de adquisición de datos se encarga de tomar los valores de posición del motor, flujo y presión del aire, estas son las variables fundamentales de la realización del proyecto. Para el control del motor se utilizó un módulo Puente H IBT-2, se decidió utilizar este driver de alta potencia dado 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.



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

Se utilizaron dos sensores ópticos lm393, 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.

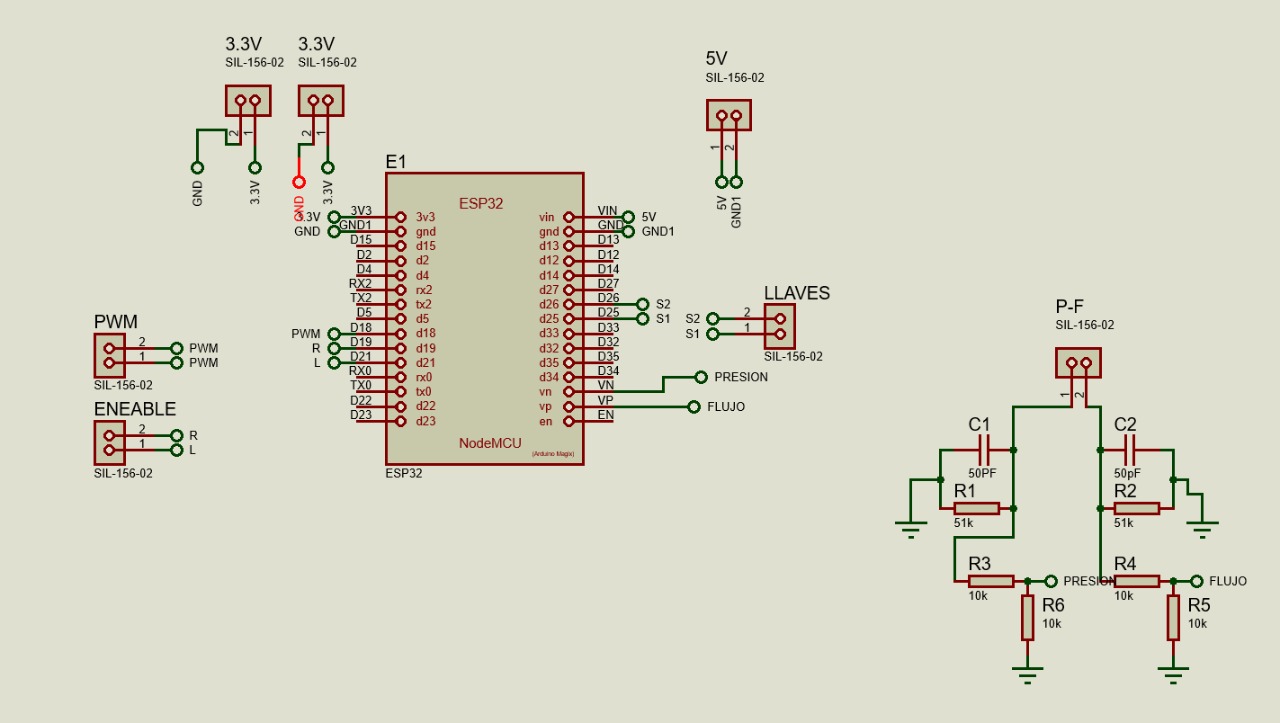
Para la comunicación de los datos con el software de visualización se utilizó una tarjeta de desarrollo ESP32 DEVKIT V1. La cual posee las características que se requieren, Wi-fi de 2.4 MHz a 150 Mbits/s, doble núcleo para mayor capacidad de procesamiento, protocolo de comunicación I2C y versatilidad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



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

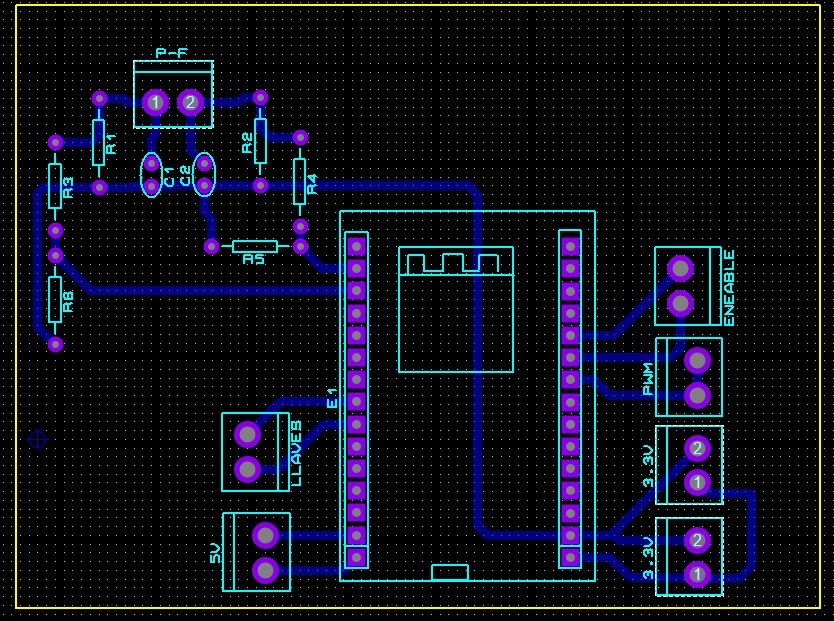
La tarjeta anteriormente mencionada se alimenta con 3.3V por intermedio del puerto microUSB al equipo.

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



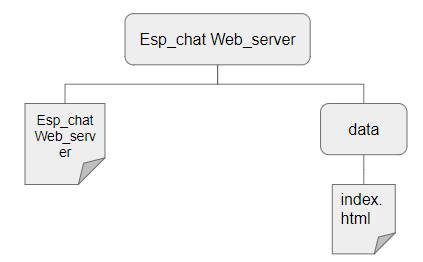
**Figura 16. Circuito de adquisición de datos.**

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



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

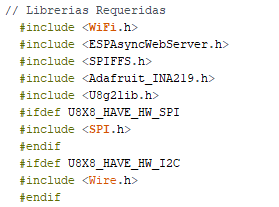
Se diseñó el código de la tarjeta en el IDE de Arduino, con el fin de utilizar la librería ESPAsyncWebServer.h para la creación de un servidor web asíncrono, asimismo el código HTML de la página estará también en la memoria de la ESP32 contando con gráficas para la visualización en tiempo real de las 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.



**Figura 18. Diagrama estructura organizacional ESP32.**

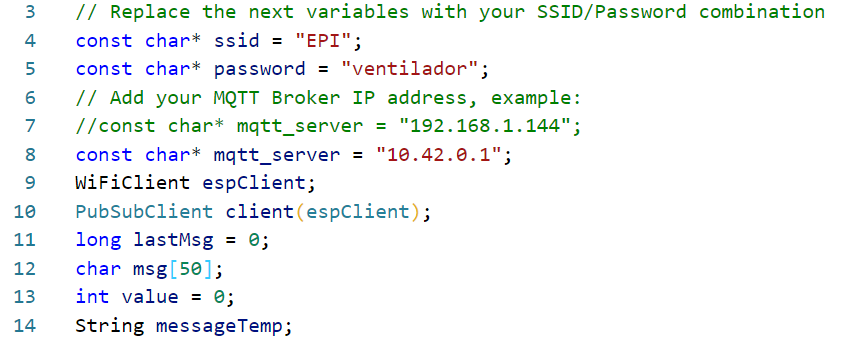
**6.1 Arduino Script:**

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.



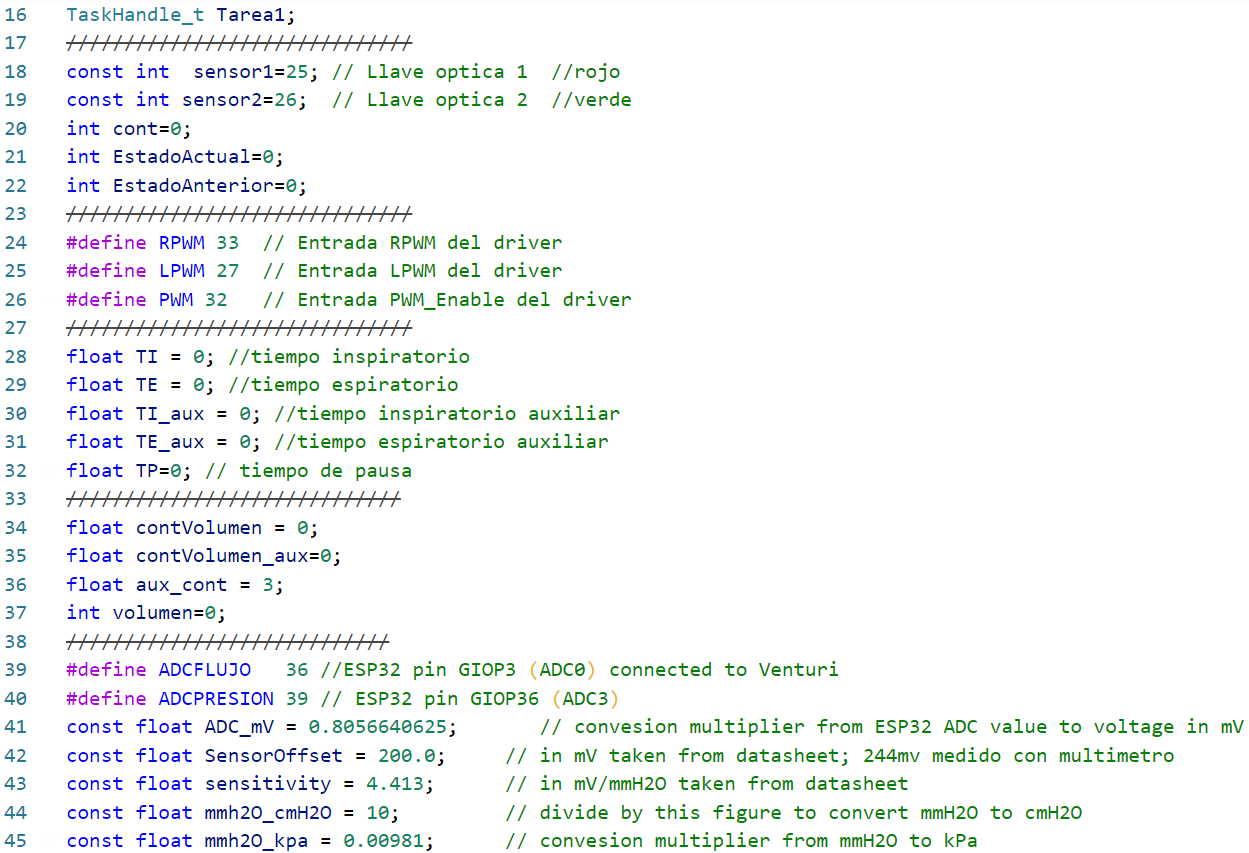
**Figura 19. Código Arduino Sketch Librerías.**

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.



**Figura 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.



**Figura 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.

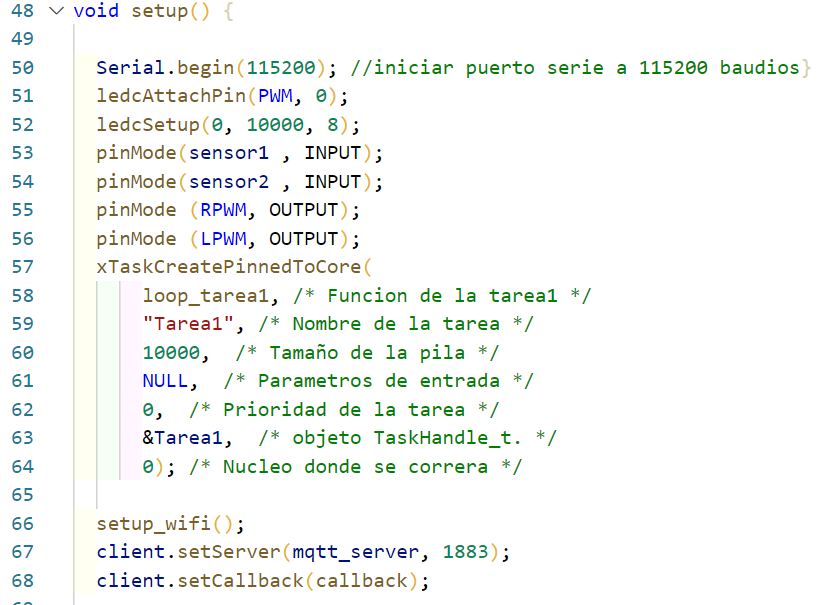
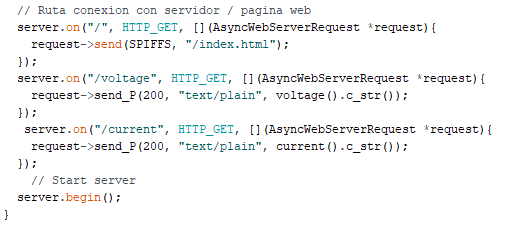


Figura 22. ***Código Arduino Sketch Setup.***

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.



**Figura 23*. Código Arduino Sketch Setup.***

En el Void Loop hacemos la lectura constante de los sensores del enconder, 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.



**Figura 24. Código Arduino Void Loop.**

## **6.2 HTML Script:**

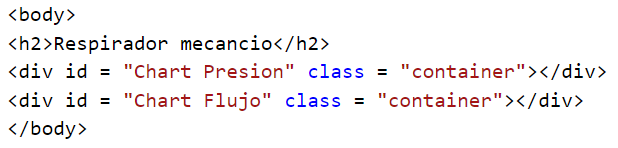
Este Script es utilizado con la finalidad de poder construir la aplicación móvil a través de una dirección web, en la cual visualizamos la información de los datos o valores generados por los dispositivos que conforman el Kit Horizon. Cabe destacar que una de sus ventajas es que se puede obtener esta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

****

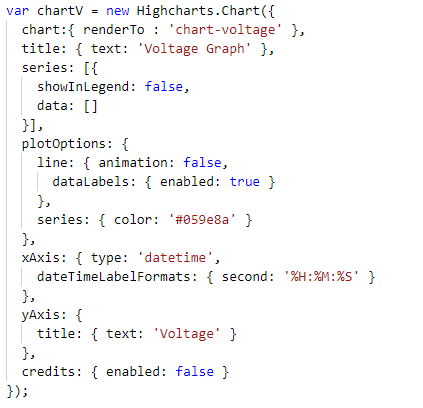
**Figura 25. Código HTML Librería.**

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.



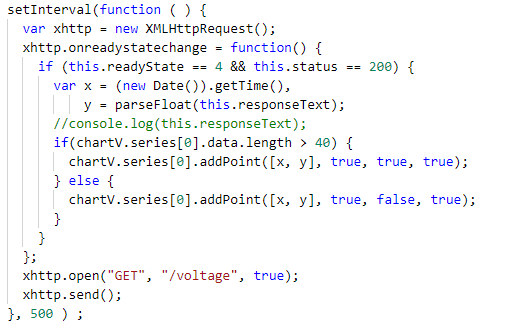
**Figura 26. Código HTML Instancias de cada gráfica.**

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



**Figura 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.

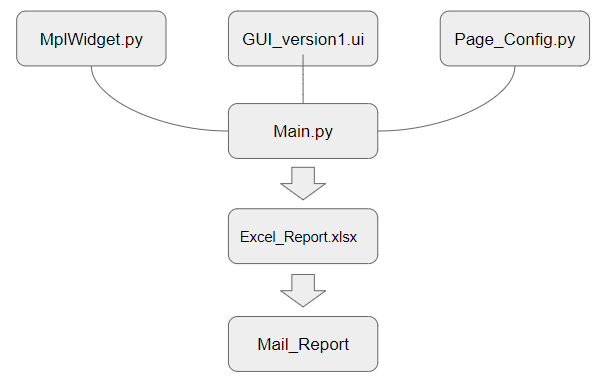


**Figura 28. Código HTML Actualización gráfica.**

## **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 tambienla habilidad de entrar en un modo desarrollador donde puede variar las constantes de control y por ultimo puede ingresar las variables de entrada para el arranque del ventilador mecanico.

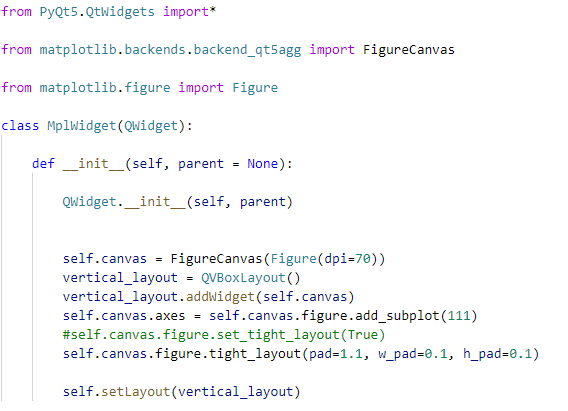
El código se compone de varios scripts llamados desde el código principal para hacer funcionar el programa incluyendo la creación del objeto widget que genera la ventana en windows a modo de programa en MplWidget.py, la configuración del front de la aplicación en el archivo de extensión UI GUI\_version1 y un archivo Page\_Config diseñado para alojar funciones extras que se necesitan fuera del main. Del código main se genera un entregable con el registro de datos.



**Figura 29. Diagrama estructura código en python.**

## **6.4 MplWidget Code**

Como se menciona esta sección contiene lo necesario para la creación de la ventana y las gráficas requeridas dentro del programa desarrollado. Se importan las librerias de Pyqt5.Qtwidgets, Matplotlib.qt5agg y matplotlib.figure, se crea una clase de una instancia tipo widget y se inicializa la configuración del marco de la figura.



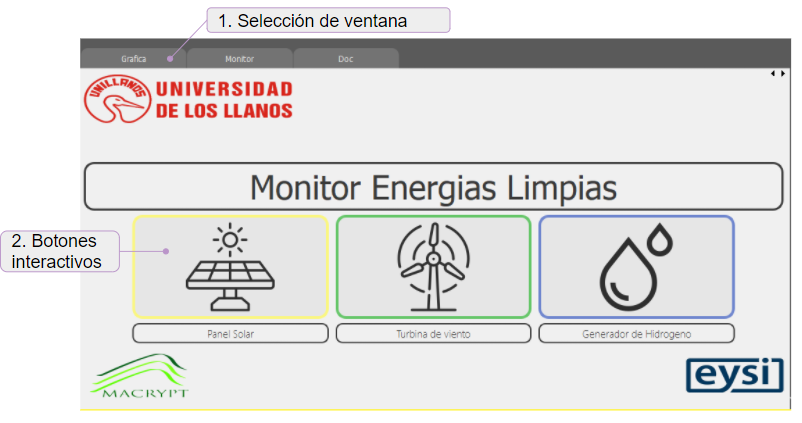
**Figura 30. Código Python MplWidget configuration figura para las gráficas.**

## **6.5 GUI Código**

Este módulo de código en formato UI es un archivo que posee el código del front-end del proyecto, código generado por el editor de interfaz de usuario que se utilizó para la construcción del programa. 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, page home en la cual se ve una presentación del aplicativo y las opciones para ingresar al monitor de cada equipo.

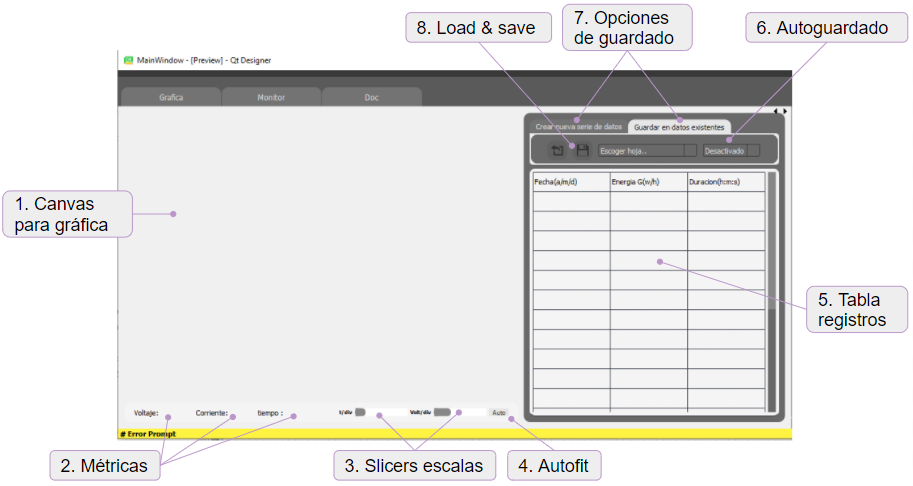
1. Selección de ventana: Al haber ingresado previamente a alguna de las pestañas esta permanecerá activa en la parte superior para volver a abrirla
2. Botones interactivos: las imágenes dinámicas que se encuentran en el centro son los botones para que el usuario acceda a la página de la gráfica correspondiente a cada equipo



**Figura 31. *Gui Front QtDesigner Page Home.***

La siguiente pestaña es page graph donde se encuentra la gráfica las métricas principales, la tabla con los registros de tomas de datos realizados y las opciones de carga y guardado en el archivo excel y vía correo.

1. Canvas para graficar: El espacio amplio es donde se mostrará la gráfica trabajando en tiempo real
2. Métricas: en la sección inferior se muestran los indicadores principales de la gráfica.
3. Slicers escalas: los slicers se utilizan para cambiar la división de las unidades del axis de la gráfica.
4. Autofit: Este botón corrige la posición de la línea de la gráfica en el centro del campo.
5. Tabla registros: en esta se mostrarán los registros creados anteriormente cuando el usuario quiere cargar datos.
6. Autoguardado: Es una lista desplegable con el tiempo que se refiere a la frecuencia con la que se auto guardará la información.
7. Opciones de guardado: En las pestañas superiores el usuario elige si está creando un registro en una hoja nueva o en la misma hoja con otras muestras
8. Load & Save: Con estos botones se carga o guarda manualmente la información.



**Figura 32. *Gui Front QtDesigner Page Graph.***

La pestaña page doc, contiene una guía de texto sobre los elementos que hacen parte del kit Horizon y los experimentos posibles con los mismos, dicha información se encuentra de manera ampliada junto con la teoría detrás del funcionamiento de cada equipo en la sección lateral que se mostrará más adelante.



**Figura 33. *Gui Front QtDesigner Page Doc.***

Esas son las ventanas principales existentes, luego como se menciona existe la seccion lateral derecha con unas funciones extras que incluyen lo siguiente:

1. Texto complementario: Esta sección que posee información dinámica referente al equipo que se está trabajando
2. Botón Play: Este botón pausa o inicia la gráfica en cualquier momento.
3. Desplegable puertos COM: Esta lista contiene los puertos COM disponibles en el equipo, intentará conectar al puerto que se seleccione
4. Botón Home: Este botón devuelve la pantalla a la vista del inicio.
5. Botón extras gráfica: Este botón expande la sección que vimos al lado derecho de la gráfica con las opciones de guardar y cargar.
6. Botón Documentación: este Botón abre la página Doc. mostrada anteriormente
7. Botón Sección complementaria: Cuando se oprime el botón señalado se expande o comprime la sección de texto complementario.

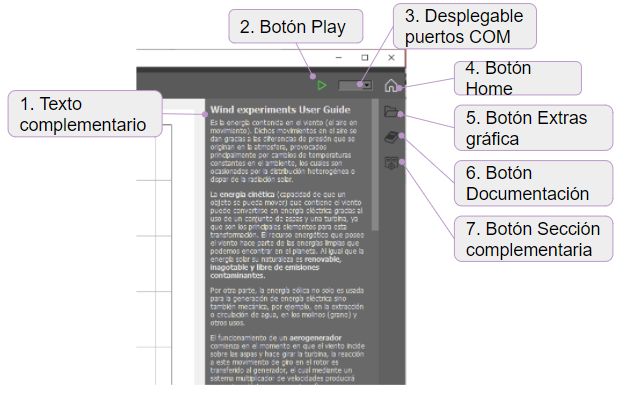


Figura 34. ***Gui Front QtDesigner Sección Lateral.***

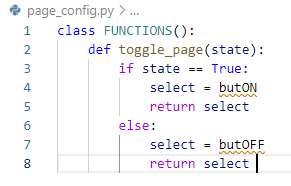
El código que contiene toda la construcción que se mostró se contiene en un archivo compatible con python, por medio de una librería dentro del módulo de Qt se importa el archivo.



Figura 35. ***Código Front archivo en formato .ui***

## **6.6 PAGE\_CONFIG CÓDIGO**

Este módulo de código en formato py contiene una clase definida con la función de alojar todas las funciones de tareas externas que se requieran dentro del script principal, actualmente solo posee una función dedicada al manejo entre páginas de la interfaz.

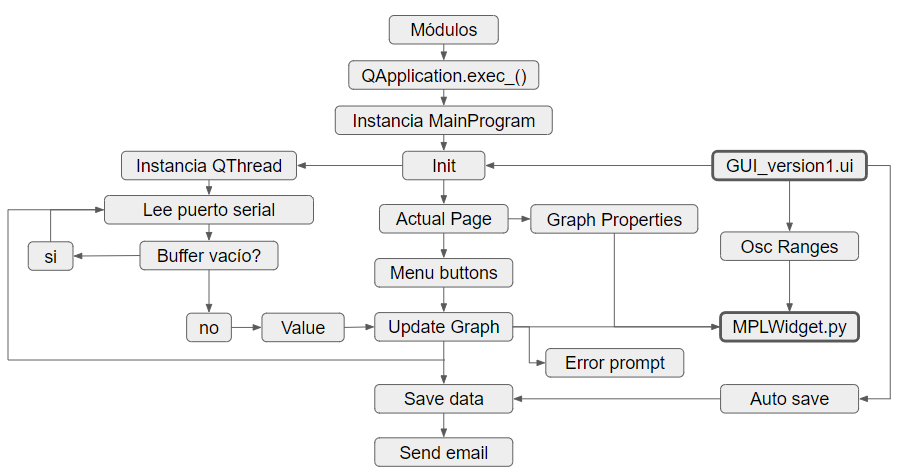


**Figura 36. *Código Python page\_config selector de páginas.***

## **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, debajo se puede ver un diagrama de flujo compactado del código que resume el proceso del programa, 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 MPLWidget que posee la configuración de la aplicación y la gráfica.

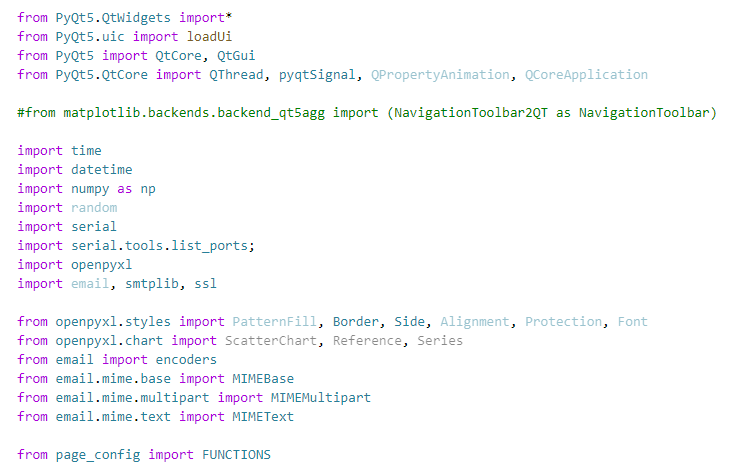
Se crea el hilo de lectura del puerto serial que va estar en loop con la función update graph mientras se esté corriendo la gráfica, y a su vez se fija la pestaña inicial de la GUI y se espera instrucción del usuario. Desde update graph se generan las alertas de error si oportunas y se hace el guardado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



**Figura 37. *Diagrama de flujo Main.py.***

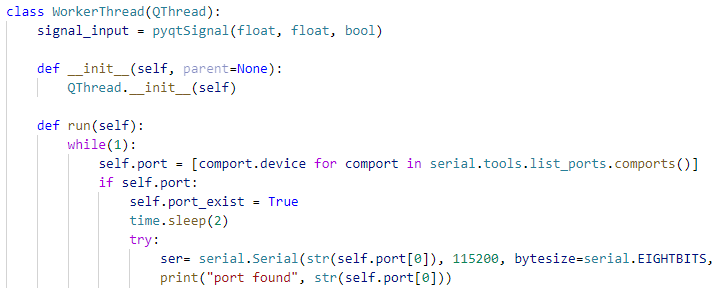
Inicialmente se incluyen los siguientes módulos externos requeridos para el funcionamiento del programa, conformados por: PyQt5 incorporando QWidgets para la generación de las gráficas, QThread y pyqt5signal para la lectura de las variables en paralelo con la actualización del widget que es el programa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

Aparte se incluyen los módulos “time”, “datetime”, “numpy”, “random”, “serial”, “openpyxl” e “email” para cumplir las demás tareas.



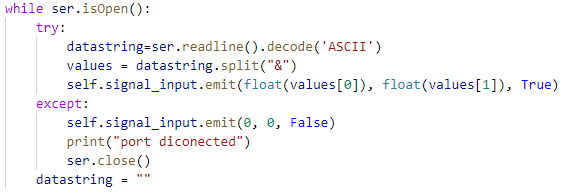
**Figura 38. *Código Python Main code Importación de módulos.***

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 el puerto serial si está disponible y se guarda en una instancia denominada “ser”.



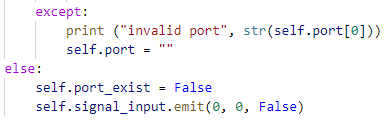
**Figura 39. *Código Python Main código creación de hilos, identificación de puerto serial.***

Si el puerto serial está abierto se lee y almacena en un buffer la información contenida, se decodifica y separa en las variables y luego es transmitida por parámetros con el programa, en caso de no tener lectura o que se desconecte el puerto se transmiten las variables en cero, se cierra la instancia del puerto y salta el prompt de error.



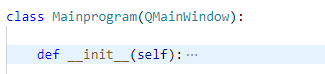
**Figura 40. *Código Python Main código creación de hilos, lectura del puerto serial.***

De igual manera ocurre si no se encuentran puertos o no es posible conectar.



**Figura 41. Código Python Main código creación de hilos, error en puerto.**

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.



**Figura 42. 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.



**Figura 43. 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 y el estado actual del puerto COM, con estos datos se actualiza el siguiente frame de la gráfica en ejecución.



**Figura 44. 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 o error guardando los datos.



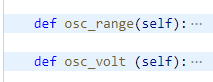
**Figura 45. *Código Python Main code function update\_graph.***

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.



**Figura 46. *Código Python Main code función reboot\_graph.***

Las funciones **osc\_range** y **osc\_volt** permiten manejar la escala de la gráfica en el eje del tiempo y de la variable mediante slicers presentes en la gráfica en tiempo real.



**Figura 47. *Código Python Main code funcion osc\_volt y osc\_range.***

La función **adjust** se utiliza para mantener la gráfica dentro de los ejes del plano al variar su escala, esto mediante un botón presente en tiempo real.



**Figura 48. Código Python Main code función adjust.**

La función **Rightmenu** se utiliza para manejar la sección dinámica a la derecha de la interfaz permitiendo que se oculte o aparezca con un tamaño predeterminado ya parametrizado.



**Figura 49. Código Python Main code función Rightmenu**

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



**Figura 50. Código Python Main code función monitor\_mode.**

La función **actualPage** define las características visuales y de funcionamiento de las 3 gráficas existentes (Panel Solar, Turbina de viento y generador de hidrógeno), hace visibles las características necesarias en cada instancia y selecciona la gráfica escogida.



**Figura 51. 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.



**Figura 52. Código Python Main code función ePrompt.**

La función **customBar** define el estilo visual del frame por medio del código CSS con los elementos de cada gráfica.



**Figura 53. *Código Python Main code función customBar.***

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.



**Figura 54. Código Python Main code función menu\_buttons.**

La función **auto\_save** fija el tiempo de autoguardado de la información dependiendo del valor que se recibe del usuario en el desplegable y genera las interrupciones para hacer efectivo el proceso de guardado.



**Figura 55. Código Python Main código función auto\_save.**

La función **save\_data** ejecuta el guardado de la información en el archivo xlsx mediante el uso del módulo openpyxl, se abre el archivo xlsx en la misma ruta del proyecto, se guardan los datos nuevos junto con unos parámetros del registro incluyendo tiempo transcurrido, la potencia y el tiempo de muestreo en serie con registros anteriores o en una hoja nueva dependiendo de la selección del usuario.



**Figura 56. *Código Python Main code función save\_data.***

La función **load\_combobox** permite cargar las hojas de datos dentro del archivo xlsx en forma de lista desplegable en la interfaz para ser seleccionadas, mediante el uso del módulo openpyxl se explora el archivo, luego se borran los registros que ya hay en el combobox, se recorre la lista de hojas en el archivo y se van agregando en el desplegable.



**Figura 57. *Código Python Main code función load\_combobox.***

La función **table** permite cargar las capturas registradas cuando el usuario selecciona una hoja en el combobox trayendo de la base las hojas existentes en forma de lista extendida con la información de cada registro guardado.



**Figura 58. *Código Python Main code función table.***

La función **load\_data** basado en el registro seleccionado en la tabla de registros mencionada en las funciones anteriores load\_combobox y table carga los datos completos de dicho registro en la gráfica de forma estática para su visualización.

.

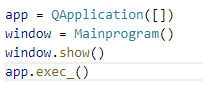
**Figura 59. Código Python Main code función load\_data.**

La función **sendmail** genera el envío de un correo electrónico desde una dirección específica hacia un destinatario programado, añadiendo como adjunto el archivo xlsx que contiene la información completa, dicho envío hace parte del auto\_save que se utiliza implicando que esta permite hacer reportes recurrentes de la información.



**Figura 60. Código Python Main code función sendmail.**

Una vez se corre el programa en el orden que se muestra, se crea la 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.



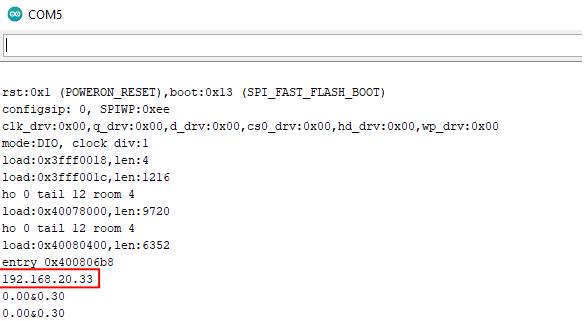
**Figura 61. *Código Python Main code función principal.***

# **7. RESULTADOS Y ANÁLISIS**

## **7.1 RESULTADOS**

### **7.1.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL KIT HORIZON**

Con el sistema de instrumentación construido se realizaron pruebas con los 3 equipos del Kit, al implementar el prototipo del circuito diseñado conectándolo a un computador para hacer la alimentación y teniendo una red Wifi estable presente en el sitio 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.



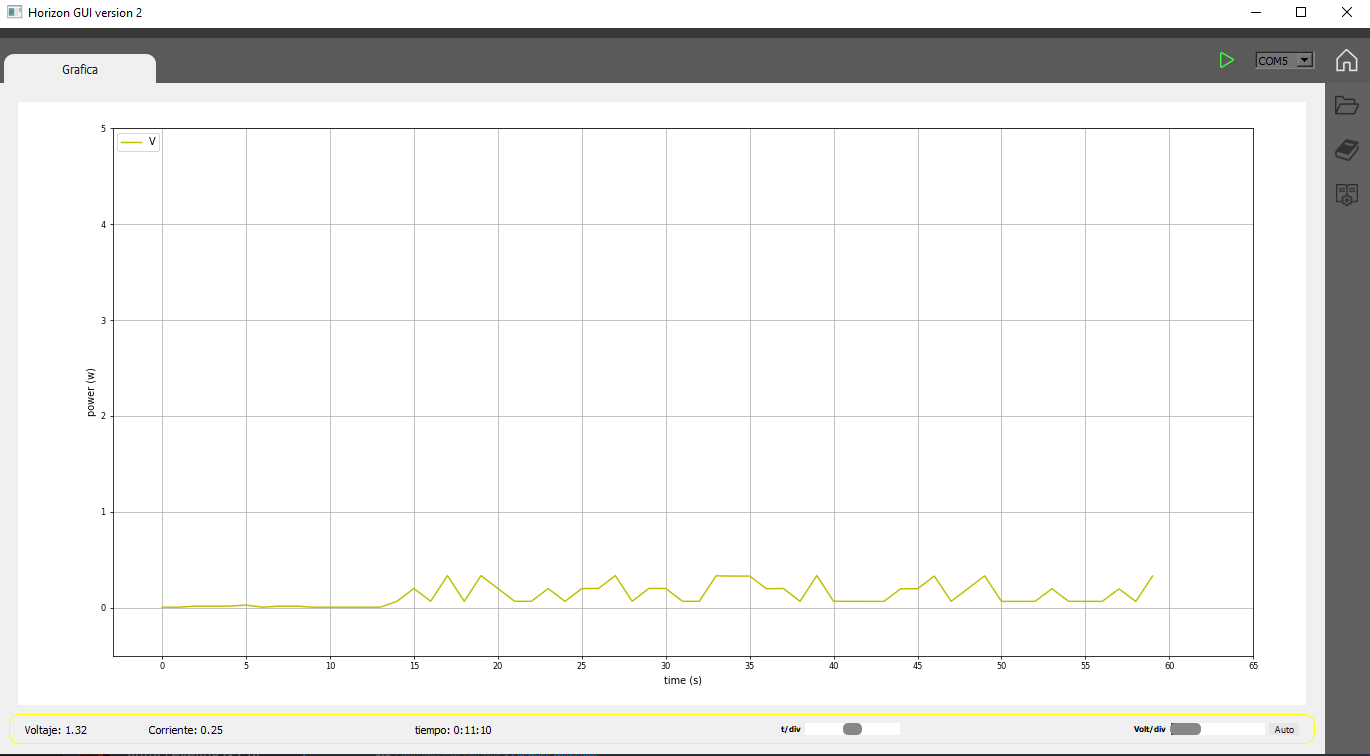
**Figura 62. *Monitor Serial Arduino IDE, IP ESP32.***

El diseño final de la interfaz de usuario que se desarrolló permite escoger el equipo del kit que se quiere utilizar, 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.



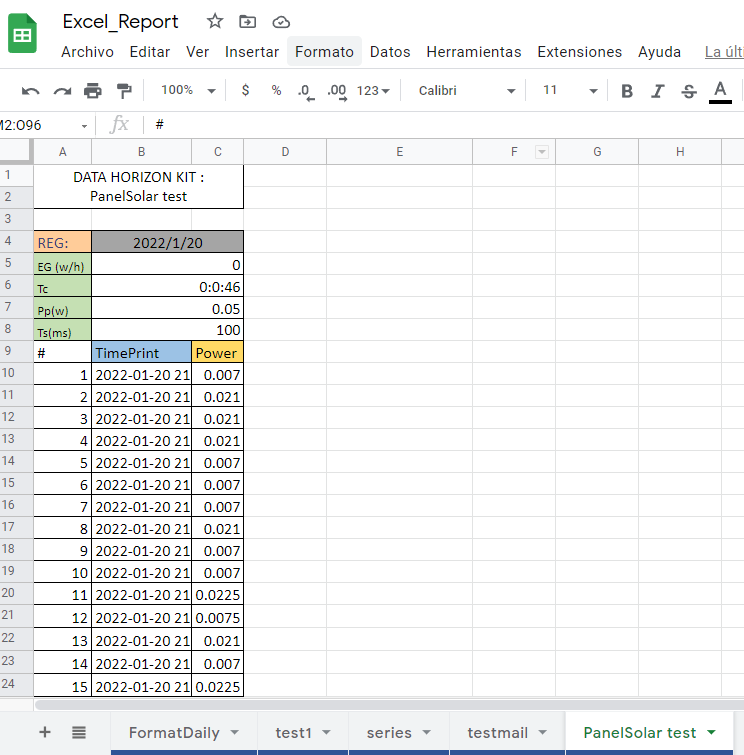
**Figura 63. *Interfaz de usuario diseño final página de inicio.***

Empezando por seleccionar el panel solar se conectó a los pines de entrada al sensor y se capturaron datos en un cuarto sin luz y luego con luz artificial se alimentó el panel, la siguiente figura muestra la lectura recibida en mW, también se pueden apreciar las últimas lecturas de corriente y voltaje a la entrada.



**Figura 64. *Monitor Python muestreo Panel solar.***

De dicha lectura se generó el siguiente apartado en la base de datos (figura 63) donde generando la serie de datos del voltaje en el panel solar basado en la potencia por intervalo y la corriente promedio sin luz y con luz se obtuvo el gráfico de la figura 64 donde se puede observar la generación a partir de la luz recibida y también se ve una variación de aproximadamente 0.1V a 1V en la medición del sensor

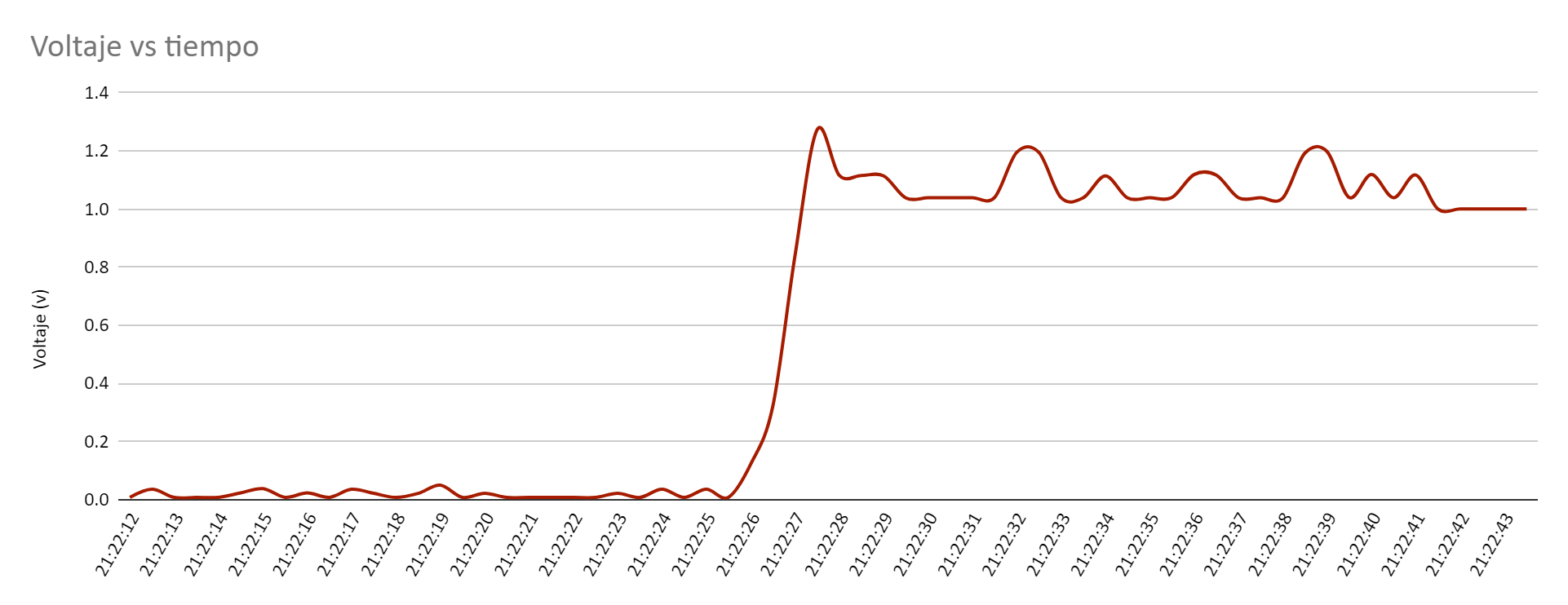


**Figura 65. *Registro de captura de datos en archivo xlsx.***

Como se menciona en el procedimiento el guardado de los datos genera tambien la entrega en correo electrónico la cual vemos en la siguiente figura

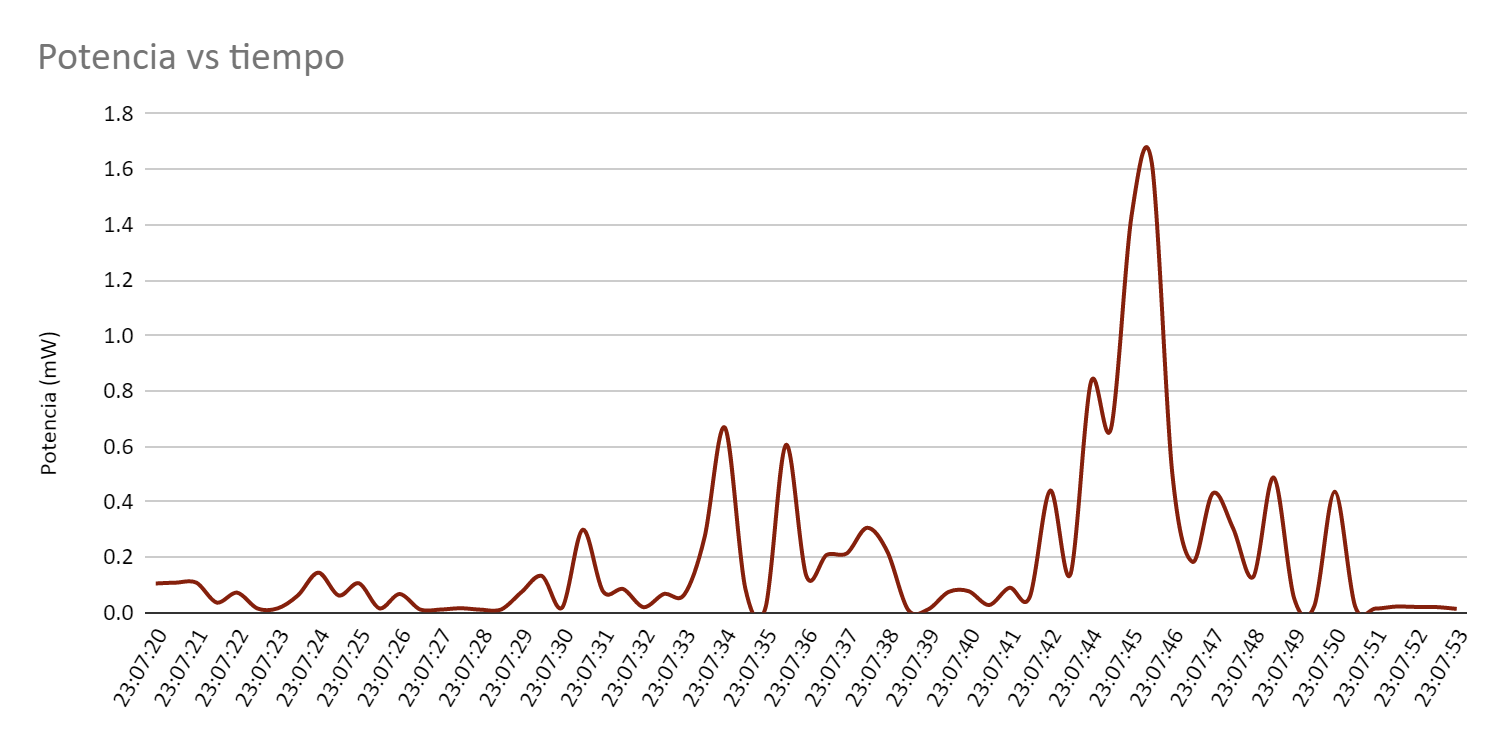


**Figura 66. Reporte vía correo electrónico de una captura de datos.**



**Figura 67. *Gráfica voltaje del panel solar en el tiempo, respuesta escalón.***

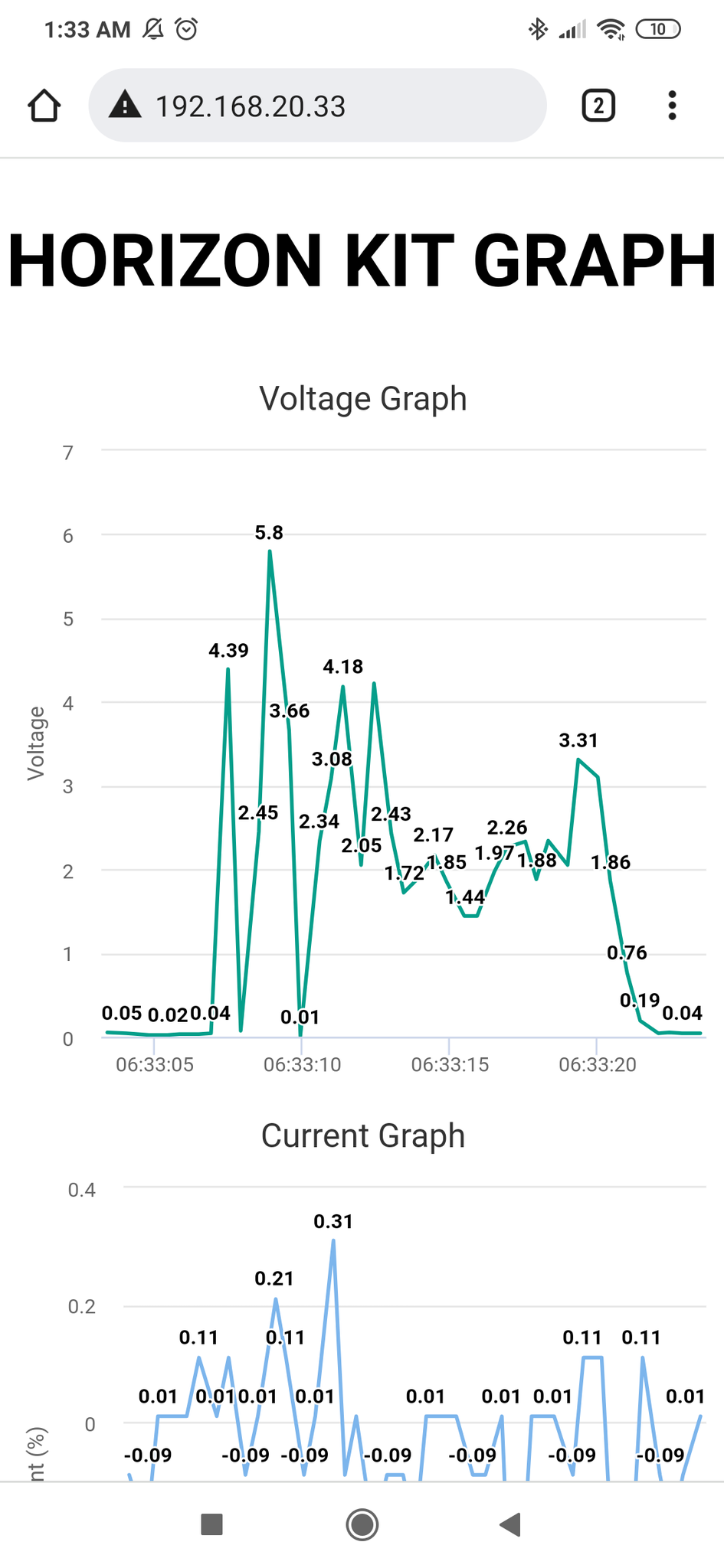
De la misma manera se hizo un muestreo de la turbina de viento, cabe mencionar que no se tenía un generador de viento por lo cual el comportamiento resultante se debe a que se movió la la turbina manualmente para generar la lectura en la aplicación.



**Figura 68. *Gráfica voltaje de la turbina de viento en el tiempo, respuesta a alteraciones manuales.***

El tercer equipo del Kit la celda de hidrógeno del Kit específico en posesión para el proyecto no estaba preparada para ser usada debido a que contiene piezas en estado inadecuado para su correcto funcionamiento.

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 62, 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.



**Figura 69. HTML aplicación en red local*.***

## **7.2 ANÁLISIS**

### **7.2.1 SISTEMA DE INSTRUMENTACIÓN VIRTUAL PARA EL KIT HORIZON**

Para la identificación de los parámetros eléctricos de los equipos se consultó el manual de usuario del Kit y se hicieron las mediciones con 2 de los elementos principales, el panel solar, y la turbina de viento, 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 7V 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 las energías renovables detrás de cada elemento, de lo cual se encontró que la generación a esta escala, se espera ser muy baja 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 en una base y generar reportes de alguna forma, 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 de las energías renovables y experimentos de cada equipo como material educativo, además el amplio catálogo de módulos de fuente abierta encontrados en la red de python permiten añadir funciones al programa, como el guardado que se hizo por medio de archivos excel.

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 bueno 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 grafica 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 Kit Horizon facilita en gran medida la trasmisión de conocimiento por medio de la metodología STEM, puesto que le permite la visualización al usuario de las diferentes variables con las que cuenta un sistema eléctrico, 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 Kit Horizon, y así darle tratamiento a los datos para la toma de decisiones en un posible proyecto macro de energías renovables.
* 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 sensor INA219 maneja una resolución más precisa con lo que respecta a valores eléctricos, algo que nos permite medir los rangos que manejan los equipos del Kit Horizon de manera más eficaz.
* El Kit Horizon Education es una herramienta versátil a la hora de conocer el funcionamiento de las diferentes nuevas tecnologías que existen en la actualidad para la generación de energía eléctrica de una manera limpia y amigable con el medio ambiente, como lo son la energía solar, eólica y la generada desde el agua a través del electrólisis para la generación de hidrógeno.
* 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 energía eléctrica de una manera amigable con el medio ambiente, relacionando todos sus aspectos.
* 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.
* La socialización del proyecto a la institución San Antonio del Ariari nos brindó una amplia visualización del panorama que se vive en esta zona del Ariari, la cual ha sido golpeada por el conflicto armado. Dentro de lo visualizado cabe resaltar el poco conocimiento de estas nuevas tecnologías creadas para la obtención de energía eléctrica.
* El interés logrado por conocer de estas energías renovables en los estudiantes de la institución educativa fue considerable, ya que se generaron preguntas de dónde y cómo podría estudiar este tipo de tecnologías, y a su vez ser aplicadas en sus ambientes de desarrollo.

# **9. REFERENCIAS**

[1] T. C. Kandpal and L. Broman, “Renewable energy education: A global status review,” *Renew. Sustain. Energy Rev.*, vol. 34, pp. 300–324, 2014.

[2] M. YEPES, “STEM Y SUS OPORTUNIDADES EN EL ÁMBITO EDUCATIVO MONOGRAFÍA,” CORDOBA, 2020.

[3] A. Patricia, G. Torres, J. Edgar, C. Montaña, and P. Rocha, “La necesidad de una educación energética desde las ciencias de la sostenibilidad,” pp. 1271–1277, 2016.

[4] C. Quiñones and M. Bernal, “LabVIEW y la instrumentación virtual aplicados a la docencia y la investigación en ciencias básicas,” 2011.

[5] P. Jissette and R. Sanchez, “Apropiación del conocimiento en proyectos energéticos a pequeña escala : factor clave para el desarrollo sostenible Apropiación del conocimiento en proyectos energéticos a pequeña escala : factor clave para el desarrollo sostenible,” no. May, 2019.

[6]https://encolombia.com/educacion-cultura/geografia/colombiana/departamentos/meta/

[7] https://aprende.colombiaaprende.edu.co/ckfinder/userfiles/files/Villavicencio.pdf

[8] http://fcbi.unillanos.edu.co/fcbi/ci/

[9] T. Final and D. E. G. En, “Las energías renovables en educación infantil,” 2017.

[10] V. Alfonso and B. Ballesteros, “La educación en energías renovables como alternativa de promoción del compromiso público ascendente entre los Indígenas Wayuu en la Alta Guajira,” no. 2019, pp. 388–397.

[11] A. Q. Ramírez and J. J. Páez, “Actividades tecnológicas escolares : un recurso didáctico para.”

[12] “Habilidades para la vida: contribución de la vida científica en el marco de la década de la educación para el desarrollo sostenible” 2006.

[13] V. A. Ballesteros-ballesteros and A. P. Gallego-torres, “Modelo de educación en energías renovables desde el compromiso público y la actitud energética,” vol. 28, no. 52, pp. 0–2, 2019.

[14] A. Patricia and G. Torres, “La educación energética , una prioridad para el milenio The energy education a priority for the millennium A energia da educação uma prioridade para o milênio,” no. 21, pp. 111–120, 2015.

[15] N. A. Ávila and V. T. Gómez, “Energías renovables : una propuesta para su enseñanza,” 2014.

[16] V. H. Gómez et al., “Desarrollo y construcción de prototipos experimentales enfocados a la enseñanza de las energías renovables,” pp. 17–23.

[17] C. Guillermo, A. Aldana, O. Leonardo, and F. Ramos, “Diseño , Simulación y Análisis de un Prototipo Generador de Energía Continua Accionado por Energías Renovables ( Energía Eolica y Solar ).,” 2018.

[18] C. Alexander and C. Cerquera, “CONSTRUCCIÓN DE UN MÓDULO AEROGENERADOR EÓLICO (M.A.E), PARA LA ENSEÑANZA Y DIVULGACIÓN DE LAS ENERGÍAS RENOVABLES (EÓLICA” 2019.

[19] S. C. Castillo, J. Sebastián, R. Aguilar, and E. Francisco, “FOTOVOLTAICA.”

[20] L. Cilleruelo, “Una aproximación a la Educación STEAM . Prácticas educativas en la encrucijada arte , ciencia y tecnología An approach to the STEAM . Educational practices at the crossroads art , science and technology,” pp. 1–18, 2014.

[21 ]Á. Alsina, “Conexiones matemáticas a través de actividades STEAM en Educación Infantil,” pp. 168–190, 2020.

[22] Dra Raquel Casado Fernández, Dra Miriam Checa Romero, “Robótica y Proyectos STEAM: Desarrollo de la creatividad en las aulas de Educación Primaria/PIXEL-BIT.”

[23] K. Garc, C. Jos, and B. Escobar, “Producción limpia y sustentable : un análisis a las fuentes de energía renovable,” 2019.

[24] Franco and F. Saúl, “La importancia de los factores productivos y su impacto en las organizaciones agrícolas en León Guanajuato México,” 2016.

[25] S. V. Peña, “PREPARACIÓN,” no. November, 2016.

[26] B. Silva, “Disponible en: http://www.redalyc.org/articulo.oa?id=39113216,” 2004.

[27] C. Paper, C. Milera-rodr, and F. I. Hatuey, “La utilización de fuentes renovables de energía . Importancia de las fincas agroenergéticas,” no. December, 2021.

[28] “La energía renovable : Importancia de su implantación y desarrollo,” pp. 1–47, 2019.

[29] L. Paola, U. Valencia, F. Leonard, A. Moreno, J. César, and R. Rodríguez, “Importancia de las energías renovables en la seguridad energética y su relación con el Importance of renewable energies in the energy security and its relationship with economic growth . Importância das energias renováveis na segurança,” pp. 231–242.

[30] V. Ballesteros. “Panorama mundial de las energías renovables e importancia de la energía fotovoltaica” Revista científica. Universidad Distrital Francisco José de Caldas. 2016.

[31] “Energía solar fotovoltaica para la agricultura y desarrollo rural sostenibles” no. 3, 2000.

[32] C. Espejo and R. García, “La energía eólica en la producción de electricidad en España 1,” vol. 136, pp. 115–136, 2012.

[33] A. Hector, M. Diaz, M. Alonso, and M. Begazo, “Estudio de prefactibilidad para la instalacion de una planta de energia eolica en paracas,” 2017.

[34] F. D. I. Eléctrica, “Universidad Central ‘ Marta Abreu ’ de Las Villas TRABAJO DE DIPLOMA Los PLC ´ s y la generación de energía eléctrica mediante energía eólica y las biomasas Universidad Central ‘ Marta Abreu ’ de Las Villas Facultad de Ingeniería Eléctrica Los PLC ´ s y la generación de energía eléctrica mediante,” 2016.

[35] A. L. Gobierno, D. E. A. Pastrana, P. Arteta, M. Roberto, and G. Arana, “SEGURIDAD , DEFENSA Y EDUCACIÓN BÁSICA EN COLOMBIA : UNA APROXIMACIÓN,” vol. 35, no. 2, pp. 11–23, 2014.

[36] C. Ariari, P. Consuelo, and L. Marín, “Hacia un Modelo de Comunicación Comunitaria para el Pos-acuerdo .,” 2020.

[37] R. Sist and P. Agroecol, “Rev Sist Prod Agroecol. 11: 1: 2020 121,” pp. 121–140, 2020.

[38] H. Ramírez et al., “Disponible en: http://www.redalyc.org/articulo.oa?id=79831197009,” 2014.

[39]https://programarfacil.com/blog/arduino-blog/sensor-ina219-arduino-esp8266-esp32/

[40]https://programarfacil.com/esp8266/esp32/

[41]https://www.becas-santander.com/es/blog/python-que-es.html

[42]https://realpython.com/qt-designer-python/

[43]https://unipython.com/pyqt5-interfaces-graficas-con-python/

[44]https://www.horizoneducational.com/renewable-energy-science-kit/p1218