



UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL
CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

**Diseño de un prototipo de invernadero con control PID
de radiación ultravioleta (UV) utilizando un ESP32**

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Electrónica

AUTORES: AXEL GUSTAVO LEMA PIGUVE
EDDY JOEL HERRERA ORDOÑEZ

TUTOR: Ph.D. Víctor Huilcapi S.

Guayaquil – Ecuador
2025 – 2026

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Axel Gustavo Lema Piguave con documento de identificación N° 0956795306
y Eddy Joel Herrera Ordoñez con documento de identificación N° 0951638006;
manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de
lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de
manera total o parcial el presente trabajo de titulación.

Guayaquil, 26 de enero del 2026.

Atentamente,



Axel Gustavo Lema Piguave

0956795306



Eddy Joel Herrera Ordoñez

0951638006

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Axel Gustavo Lema Piguave con documento de identificación N° 0956795306 y Eddy Joel Herrera Ordoñez con documento de identificación N° 0951638006, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Diseño de un prototipo de invernadero con control PID de radiación ultravioleta (UV) utilizando un ESP32", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 26 de enero del 2026.

Atentamente,

Axel Gustavo Lema Piguave

0956795306

Eddy Joel Herrera Ordoñez

0951638006

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Víctor Huilcapi con documento de identificación N° 0917206294, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: Diseño de un prototipo de invernadero con control PID de radiación ultravioleta (UV) utilizando un ESP32, realizado por Axel Gustavo Lema Piguave con documento de identificación N° 0956795306 y Eddy Joel Herrera Ordoñez con documento de identificación N° 0951638006, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 26 de enero del 2026.

Atentamente,



Ph.D. Víctor Huilcapi

0917206294

DEDICATORIA

Dedico este trabajo a mis padres, quienes han sido el pilar fundamental de mi vida. Su amor incondicional, apoyo constante y sabiduría infinita han sido la brújula que me ha guiado en cada paso de este camino. Gracias por enseñarme el valor del esfuerzo, la importancia de nunca rendirse y la belleza de soñar sin límites. Este logro es tanto mío como suyo, porque sin ustedes, no hubiera llegado hasta aquí. Su sacrificio y dedicación han sido la base de mis sueños y de todo lo que soy.

Con todo mi amor y gratitud, esta titulación es un homenaje a ustedes.

A cada una de las personas que creyeron en mí y me apoyaron en el transcurso de mi formación académica.

Eddy Joel Herrera Ordoñez

DEDICATORIA

Quiero dedicar este trabajo que representa la culminación de una etapa llena de esfuerzo, aprendizaje y crecimiento personal, a mi familia, quienes han sido un pilar fundamental en este largo camino. Gracias por su amor incondicional, por su paciencia infinita y por creer en mi incluso cuando yo dudaba, gracias por hacerme ver que soy capaz de lograr lo que me proponga.

Este logro no es únicamente mío, es el resultado del amor, la dedicación y el acompañamiento que me brindaron durante todo este proceso. Por ello, este trabajo de titulación es un homenaje a ustedes, quienes caminaron a mi lado y celebraron conmigo.

Gracias por ser mi mayor inspiración y por formar parte esencial de este sueño hecho realidad.

Axel Gustavo Lema Piguave

AGRADECIMIENTO

Este trabajo de titulación marca el cierre de una de las etapas más importantes de mi vida: la culminación de mi formación como Ingeniero Electrónico. Llegar hasta aquí no habría sido posible sin el apoyo incondicional de muchas personas, a quienes deseo expresar mi más sincero agradecimiento.

En primer lugar, quiero agradecer a Dios, por darme la fortaleza, la sabiduría y la perseverancia necesarias para superar cada desafío durante este proceso. Su guía ha sido fundamental en cada paso que he dado.

También, y por encima de todo, quiero agradecer profundamente a mis padres, Eddy Herrera y Yessenia Ordoñez, por ser mi mayor fuente de inspiración, fortaleza y amor. Gracias por su esfuerzo incansable, por su ejemplo de perseverancia, y por haber estado siempre a mi lado, creyendo en mí incluso cuando yo mismo lo dudaba.

Este logro es tan mío como suyo, y les dedico con todo mi corazón este trabajo.

Quiero también expresar mi cariño y agradecimiento a mi querida abuelita Petra Morales, cuya sabiduría, amor y apoyo han sido un pilar fundamental en mi vida. Su ejemplo y sus enseñanzas han iluminado mi camino y me han motivado a seguir adelante.

Quiero expresar también mi agradecimiento a Edgar García, mi jefe en el trabajo, por su apoyo, comprensión y la oportunidad de crecer profesionalmente mientras realizaba este proyecto. Su confianza y liderazgo fueron importantes para mantener el equilibrio entre mis responsabilidades laborales y académicas.

A Ingrid Carranza, mi especial reconocimiento por su constante apoyo, confianza y disposición para ayudar. Su acompañamiento fue clave en momentos importantes, y su presencia significó mucho a lo largo de este proceso.

También extendo mi agradecimiento a los docentes y al personal de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana, por compartir sus conocimientos y contribuir a mi formación académica.

A mi novia, por su amor constante, su paciencia y su confianza en mi fueron fundamentales para alcanzar este objetivo. Este trabajo también es reflejo de su compañía, que dio sentido y fortaleza a cada etapa del proceso. Finalmente, a mis amigos, compañeros y todas aquellas personas que de alguna forma estuvieron presentes durante este camino, gracias por cada palabra de aliento, cada conversación, y cada gesto de apoyo.

Con gratitud y cariño,

Eddy Joel Herrera Ordoñez

AGRADECIMIENTO

El presente trabajo es el resultado del esfuerzo, la dedicación y el apoyo de diversas personas, a quienes expreso mi más sincero agradecimiento.

Agradezco profundamente a mis padres por su amor incondicional, por los valores inculcados y por el esfuerzo constante que realizaron para apoyarme en cada paso de mi formación profesional.

Gracias a mi madre Fanny Piguave por su cariño y consejos y por cada mañana despedirme para ir al trabajo y en la noche esperar sentada en el sofá hasta las 11pm que llegaba de la universidad.

Gracias a mi padre Rodrigo Lema el cual me enseñó el valor de la honestidad y el sacrificio, él es el motivo por el cual aprendí a trabajar y le agarré amor al sector industrial.

A mi hermana, por ser un ejemplo de superación, responsabilidad y compromiso. Tu camino ha sido una inspiración en mi formación profesional.

A mi hermano gracias por su compañía y apoyo los cuales me brindaron ánimos para continuar avanzando.

Quiero agradecer infinitamente a mi enamorada Adriana Alcivar que lleva más de 10 años a mi lado gracias por apoyarme en todo este largo camino, gracias por creer en mí y ser mi apoyo constante.

Finalmente quiero agradecer a las personas que pasaron de ser compañeros a amigos, fueron una parte muy importante en toda esta etapa universitaria.

Con mucho cariño,

Axel Gustavo Lema Piguave

RESUMEN

AÑO	ALUMNOS	DIRECTOR DEL PROYECTO	TEMA DE TRABAJO DE TITULACIÓN
2025	EDDY JOEL HERRERA ORDOÑEZ AXEL GUSTAVO LEMA PIGUAVE	Ph.D. Víctor Huilcapi	Diseño de un prototipo de invernadero con control PID de radiación ultravioleta (UV) utilizando un ESP32

En este Trabajo de Titulación se diseña e implementa un prototipo de invernadero automatizado con el objetivo de optimizar el control de la radiación ultravioleta (UV), variable fundamental para el desarrollo de los cultivos. La falta de regulación adecuada en la radiación UV, así como en parámetros como temperatura y humedad, puede afectar negativamente la calidad y productividad agrícola, por lo cual es necesario aplicar estrategias de automatización que garanticen condiciones ambientales estables.

El sistema propuesto utiliza un microcontrolador ESP32 como unidad central de control, integrando sensores de UV, temperatura, humedad y luminosidad, junto con actuadores para la regulación de la luz UV mediante un algoritmo de control PID, que permite mantener la variable dentro de los rangos óptimos. Además, se desarrolla un dashboard web que ofrece la visualización de datos en tiempo real y la interacción remota con el sistema, facilitando la gestión del invernadero a escala.

Palabras clave: Invernadero, ESP32, Control PID, Radiación UV.

ABSTRACT

YEAR	STUDENTS	PRJ. DIRECTOR	SUBJECT
2025	EDDY JOEL HERRERA ORDOÑEZ AXEL GUSTAVO LEMA PIGUAVE	Ph.D. Víctor Huilcapi	“DESIGN OF A GREENHOUSE PROTOTYPE FOR THE PID CONTROL OF THE UV VARIABLE USING AN ESP32.”

In this Degree Project, an automated greenhouse prototype is designed and implemented with the objective of optimizing the control of ultraviolet (UV) radiation, a fundamental variable for crop development. The lack of adequate regulation of UV radiation, as well as parameters such as temperature and humidity, can negatively affect agricultural quality and productivity; therefore, it is necessary to apply automation strategies that ensure stable environmental conditions.

The proposed system employs an ESP32 microcontroller as the central control unit, integrating UV, temperature, humidity, and light sensors, together with actuators for UV light regulation through a PID control algorithm, which enables the variable to be maintained within optimal ranges. Additionally, a web-based dashboard is developed to provide real-time data visualization and remote interaction with the system, facilitating scalable greenhouse management.

Keywords: Greenhouse, ESP32, PID Control, UV Radiation.

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN	1
2. PROBLEMA	2
2.1 IMPORTANCIA DE LA PROBLEMÁTICA	3
3. OBJETIVOS	4
3.1. Objetivo general.	4
3.2. Objetivos específicos.	4
4. FUNDAMENTOS TEÓRICOS	5
4.1. Invernadero	5
4.2. Control PID	5
4.3. Sensores en Invernaderos	6
4.4. Actuadores en sistemas de control de invernaderos	7
4.5. Microcontrolador ESP32	8
4.6. Sensor de humedad y temperatura dht22	8
4.7. Sensor de luz uv VEML6075	9
4.8. Driver de potencia TB6612FNG	10
4.9. Fuente de 5 V DC 2.5 A	10
4.10. Importancia del control PID en invernaderos	12
5. MARCO METODOLÓGICO	12
5.1 Descripción del proyecto	12
5.2 Tipo de metodología	13
5.3 Etapas del prototipo	14
5.3.1 Arquitectura del sistema.	14
5.3.1.1 Bloque de adquisición de datos	15
5.3.1.1.1 Sensor de radiación ultravioleta	15
5.3.1.1.2 Sensor de temperatura y humedad	16
5.3.1.2 Módulos de procesamiento y control	16
5.3.1.3 Bloque de actuación	17
5.3.1.4 Bloque de señalización y supervisión	17
5.3.2 Diseño de Servidor Web	18
5.3.2.1 Pantalla Portada	20
5.3.2.2 Pantalla Panel	22

5.3.2.3 Pantalla Gráficos	26
5.3.3 Controlador PID para luz UV	29
6. RESULTADOS	34
6.1. Implementación del prototipo	34
6.2. Resultados del sistema de adquisición de datos	37
6.3. Resultados del control PID aplicado a la radiación UV	39
6.4. Resultados del bloque de actuación	40
6.5. Resultados del sistema de supervisión y visualización	41
6.6. Análisis global de resultados	41
7. CRONOGRAMA DE ACTIVIDADES	42
8. PRESUPUESTO	43
9. CONCLUSIONES	44
10. RECOMENDACIONES	44
11. REFERENCIAS BIBLIOGRÁFICAS	45
12. ANEXOS	46
12.1 PROGRAMACIÓN ESP32	46
12.2 Evidencia fotográfica del prototipo	63

ÍNDICE DE FIGURAS

Figura 1 .Esquema de un invernadero automatizado	5
Figura 2 .Esquema PID	6
Figura 3 .Sensores integrados en un invernadero	7
Figura 4 .Actuadores controlados por un sistema PID	7
Figura 5 .Microcontrolador ESP32	8
Figura 6 .Sensor de humedad y temperatura DHT22	9
Figura 7 .Sensor de luz UV VEML6075	9
Figura 8 .Driver de potencia TB6612FNG	10
Figura 9 .Fuente de 5VDC 2.5A	11
Figura 10 .Esquema de automatización de invernadero	11
Figura 11 .Control PID manteniendo la temperatura estable	12
Figura 12 .Diagrama de conexiones entre dispositivos.	15
Figura 13 .Diagrama de cableado de elementos instalados en el prototipo.	18
Figura 14 .Servidor Web ESP32 - Pantalla Portada.	21
Figura 15 .Servidor Web ESP32 - Pantalla Panel.	23
Figura 16 .Servidor Web ESP32 - Pantalla Panel – índice de UV.	25
Figura 17 .Servidor Web ESP32 - Pantalla de Gráficos.	27
Figura 18 .Variables para control PID en ESP32.	30
Figura 19 .Algoritmo de control PID en ESP32.	31
Figura 20 .Llamado de algoritmo de control PID desde el lazo principal del ESP32.	32
Figura 21 .Diseño CAD 3D del invernadero.	34
Figura 22 .Cotas del prototipo 3D del invernadero.	35
Figura 23 .Detalle de sensores y actuadores instalados en el prototipo.	36
Figura 24 .Detalle de sensores y actuadores instalados en el prototipo.	37
Figura 25 .Visualización en tiempo real de las variables ambientales medidas.	38
Figura 26 . Respuesta al sistema de control PID	39
Figura 27 .Ajuste de parámetros de controlador PID desde la interfaz web	40
Figura 28 . Estados de funcionamiento del sistema de iluminación UV.	40

Figura 29 .Fabricación de estructura metálica	63
Figura 30 .Diseño de base para estructura	64
Figura 31 .Montaje de paredes de acrílico	64
Figura 32 .Programación del ESP32	65
Figura 33 . Conexión de elementos electrónicos	65
Figura 34 . Pruebas de funcionamiento	66
Figura 35 . Montaje de circuito en maqueta	67
Figura 36 . Pruebas de funcionamiento	67
Figura 37 . Verificación de funcionalidad	68

ÍNDICE DE TABLAS

Tabla 1 . <i>Cronograma de de actividades del proyecto</i>	42
Tabla 2 .Presupuesto general del proyecto	43

1. INTRODUCCIÓN

En la agricultura moderna, los invernaderos representan una solución tecnológica que facilita el control del entorno de producción agrícola, permitiendo regular variables ambientales como la temperatura, la humedad, la luminosidad y la radiación ultravioleta (UV), con el fin de mejorar el rendimiento y la calidad de los cultivos. El manejo inadecuado de estas condiciones puede generar disminución en la productividad, pérdida de calidad y mayor vulnerabilidad a plagas y enfermedades.

La radiación UV, en particular, desempeña un papel importante en los procesos fisiológicos de las plantas, ya que interviene en la fotosíntesis, la síntesis de compuestos bioactivos y en la resistencia a agentes externos. Sin embargo, un exceso o déficit de esta radiación puede ser perjudicial, por lo que su regulación precisa es fundamental dentro de los sistemas de cultivo controlado.

Este Trabajo de Titulación tiene como objetivo principal diseñar e implementar un prototipo de invernadero automatizado utilizando un microcontrolador ESP32 y un algoritmo de control PID, enfocado en la regulación de la variable UV. El sistema se complementa con sensores de temperatura, humedad y luminosidad, además de un dashboard web que permite supervisar y gestionar en tiempo real las condiciones del invernadero.

Con este propósito, se abordan aspectos como la construcción del prototipo a escala, la programación del microcontrolador con el algoritmo PID, el diseño de la interfaz web para la visualización de datos, y la validación experimental del sistema.

La finalidad de este trabajo es demostrar la viabilidad técnica de integrar tecnologías de automatización y control en el ámbito agrícola, mediante el diseño e implementación de un sistema basado en ESP32 y control PID, orientado a regular de manera precisa y eficiente la radiación ultravioleta (UV) en invernaderos, con el propósito de mejorar la productividad de los cultivos y contribuir al desarrollo de la agricultura inteligente.

2. PROBLEMA

En los invernaderos, el crecimiento y desarrollo de los cultivos dependen de múltiples variables ambientales, entre ellas la radiación ultravioleta (UV), la temperatura, la humedad y la luminosidad. El control inadecuado de esta condición provoca una disminución en la productividad, genera cultivos de baja calidad y aumenta la vulnerabilidad frente a plagas y enfermedades (Lycoskoufis, 2022).

La radiación UV desempeña un papel fundamental en los procesos fisiológicos de las plantas, contribuyendo a la fotosíntesis y a la síntesis de compuestos bioactivos que fortalecen la resistencia de los cultivos. Sin embargo, tanto el exceso como la deficiencia de radiación pueden ser perjudiciales, ocasionando daños en el crecimiento y reduciendo la vida útil de las plantas (Martínez, 2024). Por esta razón, la regulación precisa de esta variable es indispensable para garantizar un entorno óptimo de producción.

Estudios recientes indican que la falta de automatización en los sistemas de invernaderos reduce hasta en un 40% la eficiencia productiva, debido a que la regulación manual de la radiación UV y de otros parámetros ambientales no siempre logra mantener los rangos adecuados (FAO, 2023). Esta problemática exige la implementación de tecnologías modernas que permitan un control más preciso, confiable y sostenible de las condiciones ambientales en los cultivos.

En este contexto, la utilización de microcontroladores como el ESP32 y el diseño de algoritmos de control PID permiten mejorar el manejo de la radiación UV, manteniéndola dentro de los valores de referencia establecidos y garantizando así la estabilidad del sistema (Espressif Systems, 2023). Asimismo, el desarrollo de interfaces web para la visualización y gestión en tiempo real brinda al usuario una herramienta eficiente para supervisar y ajustar los parámetros críticos del invernadero.

No obstante, en la carrera de Electrónica y Automatización de la Universidad Politécnica Salesiana, sede Guayaquil, se ha identificado que la formación práctica en el campo de la automatización agrícola aún es limitada, ya que existen pocos prototipos experimentales disponibles en los laboratorios. Esto restringe el aprendizaje de los estudiantes en cuanto a la implementación de sistemas automatizados de control ambiental aplicados a invernaderos, lo que dificulta el desarrollo de competencias

relacionadas con la agricultura inteligente y el uso de microcontroladores para la optimización de procesos agrícolas.

Ante esta problemática, se plantea la necesidad de diseñar e implementar un prototipo de invernadero automatizado con control PID de la radiación UV mediante un ESP32, como una herramienta académica y experimental que permita tanto validar la efectividad de estas tecnologías en la agricultura como fortalecer la formación de los estudiantes en el área de la automatización.

2.1 IMPORTANCIA DE LA PROBLEMÁTICA

La agricultura es uno de los pilares fundamentales de la economía y la seguridad alimentaria, por lo que cualquier falla en el control de las condiciones ambientales dentro de un invernadero puede ocasionar pérdidas significativas en la producción y afectar la calidad de los cultivos (FAO, 2023). Mantener un equilibrio en variables como la radiación UV, la temperatura, la humedad y la luminosidad no solo mejora la productividad, sino que también garantiza cosechas más resistentes y sostenibles en el tiempo.

El control preciso de la radiación ultravioleta resulta esencial, ya que niveles inadecuados pueden ocasionar efectos negativos en el crecimiento y desarrollo de las plantas. Una regulación eficiente mediante sistemas automatizados permite reducir pérdidas energéticas, minimizar riesgos de sobreexposición o deficiencia de radiación, y asegurar la estabilidad del entorno de cultivo (Martínez, 2024).

Dentro del ámbito académico, este proyecto busca aportar significativamente a los estudiantes de la Carrera de Electrónica y Automatización al presentar un caso práctico de automatización agrícola con ESP32 y control PID. De esta manera, los estudiantes adquieren conocimientos aplicados sobre la selección, configuración y programación de microcontroladores, así como sobre el diseño de algoritmos de control para procesos reales.

Además, al enfrentar la limitada disponibilidad de prototipos en los laboratorios de la Universidad Politécnica Salesiana, sede Guayaquil, esta propuesta representa una

oportunidad para fortalecer la formación técnica, proporcionando un módulo didáctico que integra electrónica, programación, control y agricultura inteligente.

Contar con un prototipo de invernadero automatizado permite a los estudiantes desarrollar una visión más integral sobre el uso de tecnologías emergentes en la agricultura, comprendiendo la relevancia de los sistemas de control PID y la comunicación IoT como soluciones reales a los problemas de sostenibilidad y eficiencia productiva.

3. OBJETIVOS

3.1. Objetivo general.

Desarrollar un prototipo de invernadero que incorpore un sistema de control PID para regular la radiación ultravioleta (UV), empleando un microcontrolador ESP32, con el fin de optimizar las condiciones ambientales para el cultivo.

3.2. Objetivos específicos.

- Construir un prototipo de invernadero a escala, usando un microcontrolador ESP32 para el control de la radiación UV, y el monitoreo de los parámetros de temperatura, humedad y luminosidad.
- Diseñar un algoritmo de control PID para regular de manera precisa la radiación ultravioleta (UV) dentro del invernadero
- Desarrollar un dashboard web para la supervisión y operación del sistema en tiempo real.
- Validar el desempeño del prototipo bajo diferentes condiciones ambientales verificando los requerimientos establecidos para el cultivo.

4. FUNDAMENTOS TEÓRICOS

4.1. Invernadero

Un invernadero es una estructura diseñada para proteger y controlar las condiciones ambientales necesarias para el crecimiento óptimo de los cultivos. Estos ambientes permiten regular parámetros como temperatura, humedad, iluminación y ventilación, incrementando la productividad y calidad de los productos (FAO, 2023).

Figura 1

Esquema de un invernadero automatizado



Nota. El invernadero cuenta con sensores de temperatura, humedad, luz y CO₂ integrados con actuadores que permiten la automatización de las condiciones ambientales. Tomado de (FAO, 2023).

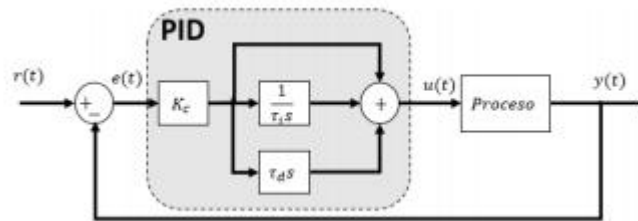
4.2. Control PID

El control Proporcional-Integral-Derivativo (PID) es una de las estrategias más utilizadas en la automatización debido a su simplicidad y eficacia. Combina tres acciones:

- Proporcional (P): responde al error instantáneo.
- Integral (I): elimina errores acumulados en el tiempo.
- Derivativo (D): predice el comportamiento futuro del sistema.

En un invernadero, el control PID permite regular variables críticas como temperatura, humedad o iluminación, reduciendo fluctuaciones y garantizando condiciones óptimas de crecimiento (Ogata, 2021)

Figura 2
Esquema PID



Nota. El controlador PID recibe la señal de error generada por la diferencia entre el valor medido por el sensor y el valor de referencia, y envía señales a los actuadores para corregir la variable controlada. Tomado de (Ogata, 2021)

4.3. Sensores en Invernaderos

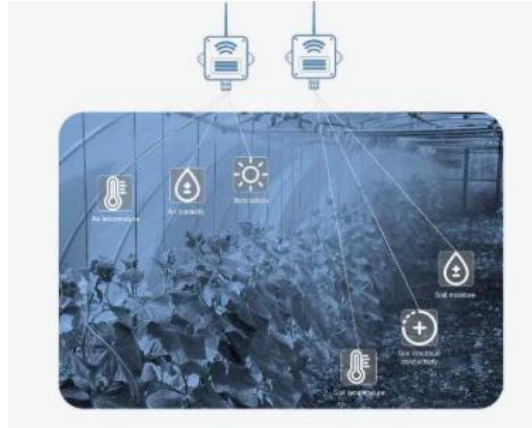
Los sensores son dispositivos electrónicos que miden parámetros ambientales en tiempo real:

- Sensor de temperatura y humedad: mide el estado del aire dentro del invernadero, activando sistemas de ventilación o calefacción según sea necesario.
- Sensor de luz (LDR o fotodiodo): regula la intensidad lumínica para garantizar un fotoperiodo adecuado de las plantas.
- Sensor de CO₂: controla la concentración de gases para mejorar la fotosíntesis.

La integración de estos sensores con un controlador PID permite ajustar automáticamente los actuadores del sistema (Serrano, 2022).

Figura 3

Sensores integrados en un invernadero



Nota. Ejemplo de sensores de temperatura, humedad, luz y CO₂ utilizados para controlar el ambiente dentro de un invernadero automatizado. Tomado de (Serrano, 2022)

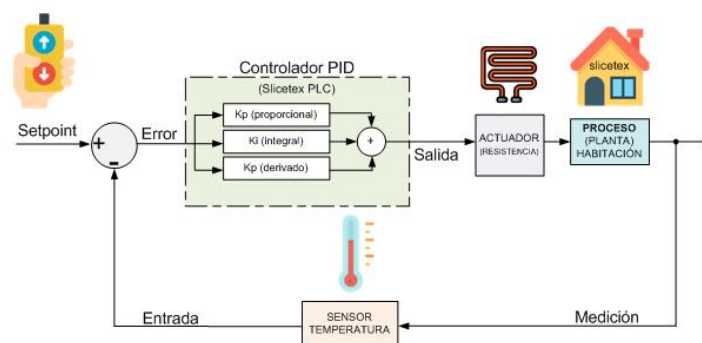
4.4. Actuadores en sistemas de control de invernaderos

Los actuadores ejecutan las órdenes del controlador:

- Ventiladores: regulan el flujo de aire para controlar temperatura y humedad.
- Sistemas de riego: mantienen el nivel de humedad del suelo.
- Lámparas LED de cultivo: proporcionan luz necesaria para la fotosíntesis.
- Calefactores eléctricos: mantienen la temperatura estable en climas fríos.

Figura 4

Actuadores controlados por un sistema PID



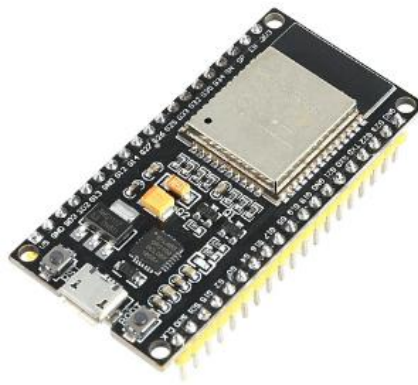
Nota. Ejemplo de ventiladores, calefactores, bombas de riego y lámparas LED controlados mediante señales de un controlador PID basado en ESP32. Tomado de (Promelsa, 2023).

4.5. Microcontrolador ESP32

El ESP32 es un microcontrolador de bajo costo y alto rendimiento que incluye conectividad Wi-Fi y Bluetooth, ideal para proyectos de automatización y control remoto. Su arquitectura permite integrar sensores, procesar datos y controlar actuadores, siendo perfecto para sistemas de invernaderos inteligentes (Espressif Systems, 2023)

Figura 5

Microcontrolador ESP32



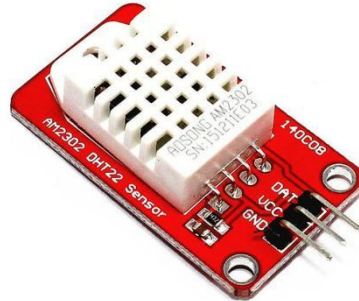
Nota. El ESP32 permite la programación de algoritmos PID y la comunicación con un dashboard web para la supervisión en tiempo real. Tomado de (Espressif Systems, 2023).

4.6. Sensor de humedad y temperatura dht22

El DHT22 (también llamado AM2302) es un sensor económico de temperatura y humedad relativa que entrega datos digitales directamente a los microcontroladores. Cuenta con un sensor capacitivo para humedad y un sensor de temperatura interna, lo que lo hace adecuado para muchas aplicaciones ambientales o de control climático en IoT. Su uso en investigación es común para validar sistemas de monitoreo ambiental con microcontroladores avanzados como el ESP32.(Ansari et al., 2023)

Figura 6

Sensor de humedad y temperatura DHT22



Nota. Permite la integración digital simplifica su lectura por microcontroladores.

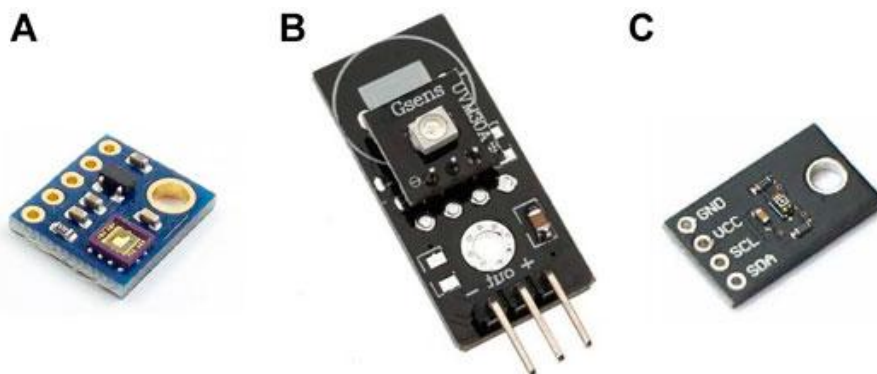
Tomado de (Ansari et al., 2023)

4.7. Sensor de luz uv VEML6075

El VEML6075 es un sensor óptico digital para radiación ultravioleta que mide tanto UVA como UVB y comunica los datos mediante el bus I²C. Este tipo de sensor es útil para aplicaciones ambientales, medición de índice UV, estaciones meteorológicas inteligentes o proyectos que requieran cuantificar radiación solar. Los estudios comparativos señalan que el VEML6075 tiene canales UVA y UVB separados y salida digital integrada, lo que lo hace más preciso y fácil de integrar que sensores UV analógicos convencionales (Serrano et al., 2022)

Figura 7

Sensor de luz UV VEML6075



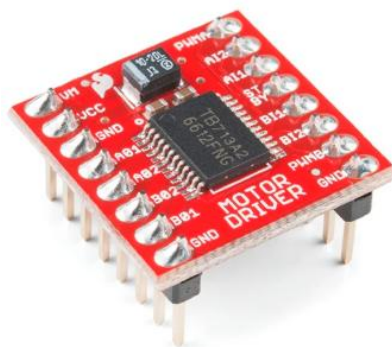
Nota. Tomada de (Serrano et al., 2022)

4.8. Driver de potencia TB6612FNG

El TB6612FNG es un driver de motor tipo puente H MOSFET que permite controlar motores DC bidireccionales desde microcontroladores como el ESP32 usando señales de PWM y dirección. Este driver gestiona hasta 1.2 A por canal (picos de 3.2 A) con modos de control flexibles (CW/CCW/brake/stop), lo que lo hace ideal para robótica, automatización y mecatrónica de pequeña escala. (Toshiba Electronic Devices & Storage Corporation, n.d.)

Figura 8

Driver de potencia TB6612FNG



Nota. Tomado de (Sparkfun Electronics S.A, n.d.)

4.9. Fuente de 5 V DC 2.5 A

Una fuente de 5 V DC estable es vital para alimentar un sistema sin ruidos o fluctuaciones que puedan causar reinicios del ESP32 o lecturas erróneas de los sensores. Aunque no hay artículos científicos específicos con DOI sobre un modelo comercial, existe literatura técnica sobre la fiabilidad de fuentes y sistemas de alimentación regulada en electrónica que discute cómo la estabilidad de la fuente afecta en la confiabilidad de sistemas embebidos. (Łukasiak et al., 2024)

Figura 9.
Fuente de 5VDC 2.5A

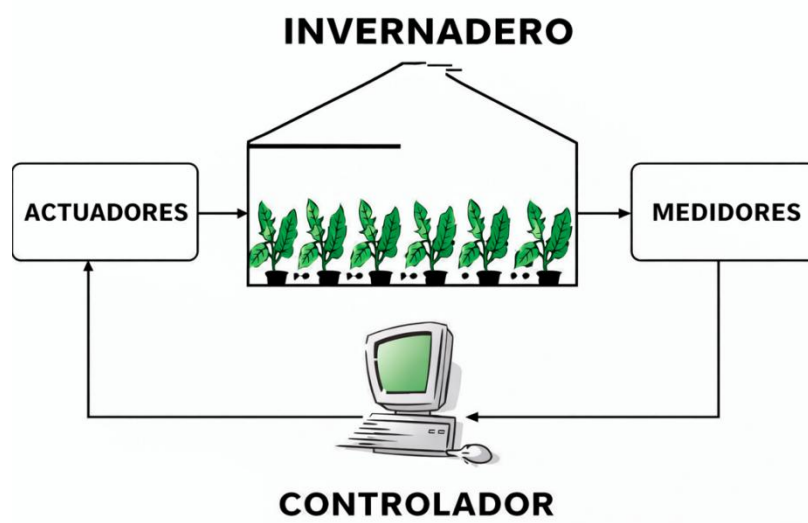


Nota. Tomada de (Shao, 2014)

Automatización de invernaderos

La automatización integra sensores, actuadores y controladores en un sistema central que monitorea y regula variables ambientales. Esto permite un ajuste automático de los actuadores frente a cambios en el entorno, garantizando condiciones óptimas de cultivo (López, 2024)

Figura 10
Esquema de automatización de invernadero



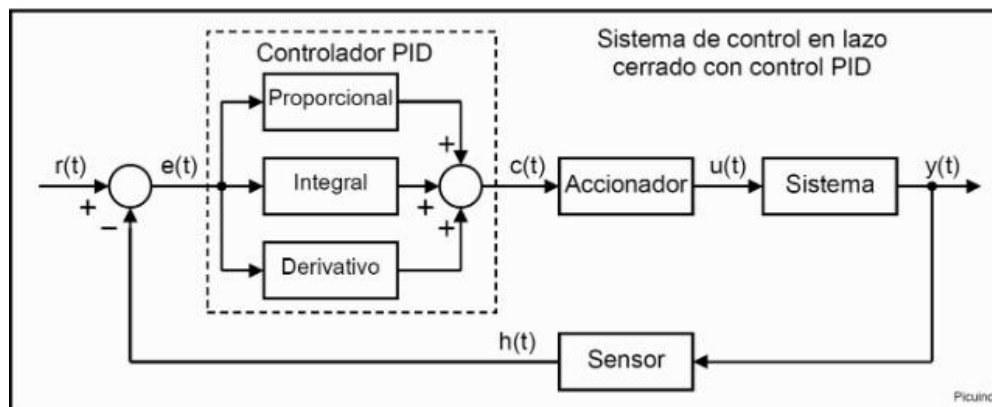
Nota. Integración de sensores, actuadores y microcontrolador para automatización de variables de temperatura, humedad y luz. Tomado de (López, 2024).

4.10. Importancia del control PID en invernaderos

La aplicación del control PID en invernaderos mejora la productividad agrícola al reducir errores humanos en la gestión del clima. Además, optimiza el consumo energético en sistemas de iluminación y ventilación, contribuyendo a un desarrollo sostenible de la agricultura (FAO, 2023).

Figura 11

Control PID manteniendo la temperatura estable



Nota. Ejemplo de la acción de un controlador PID regulando la temperatura dentro del invernadero. Tomado de (Ogata, 2021).

5. MARCO METODOLÓGICO

5.1 Descripción del proyecto

El presente proyecto consiste en el diseño e implementación de un prototipo de invernadero a escala orientado al control automatizado de variables ambientales que influyen directamente en el desarrollo de los cultivos. El sistema tiene como finalidad principal regular la radiación ultravioleta (UV) mediante la aplicación de un controlador PID, garantizando condiciones estables y adecuadas para el crecimiento vegetal.

El prototipo utiliza un microcontrolador ESP32 como unidad central de procesamiento y control, encargado de la adquisición de datos provenientes de sensores de radiación UV, temperatura, humedad y luminosidad. A partir de esta información, el sistema ejecuta un algoritmo de control PID que regula la intensidad de las luminarias UV instaladas en el invernadero.

El desarrollo del proyecto se estructura en tres fases principales. La primera fase corresponde a la construcción del prototipo físico del invernadero a escala, incluyendo la instalación de los sensores y actuadores necesarios para la adquisición de datos y la ejecución de las acciones de control. En esta etapa también se realiza la configuración inicial del microcontrolador ESP32.

La segunda fase comprende el desarrollo e implementación del algoritmo de control PID para la regulación de la radiación UV. En esta etapa se programa la lógica de control automático y se realiza la sintonización de los parámetros proporcional, integral y derivativo, con el fin de obtener una respuesta estable y eficiente del sistema.

La tercera fase se enfoca en el diseño e implementación de un dashboard web, el cual permite la supervisión y operación remota del prototipo. A través de esta interfaz es posible visualizar en tiempo real las variables ambientales medidas y analizar el comportamiento dinámico del controlador PID aplicado a la radiación UV.

Finalmente, el prototipo es validado bajo diferentes condiciones de operación, verificando que el sistema mantenga los parámetros ambientales dentro de los rangos establecidos para el cultivo seleccionado.

5.2 Tipo de metodología

Para el desarrollo del presente trabajo de titulación se aplica una metodología experimental, basada en la construcción, implementación y validación de un prototipo funcional de invernadero automatizado. Esta metodología permite evaluar el desempeño real del sistema mediante pruebas prácticas realizadas en el Laboratorio de Automatización Industrial II.

Durante la fase experimental se llevan a cabo pruebas de funcionamiento de los sensores y actuadores conectados al microcontrolador ESP32, así como la verificación del algoritmo de control PID aplicado a la variable de radiación UV. A partir de los resultados obtenidos, se realizan ajustes en los parámetros de control y calibración con el objetivo de lograr un comportamiento estable y confiable del sistema.

Adicionalmente, se emplea una metodología sistemática, en la cual se definen de manera ordenada las etapas de diseño, construcción, programación, integración de

hardware y software, y pruebas del prototipo. Este enfoque garantiza el cumplimiento de los objetivos planteados y asegura que cada fase del desarrollo contribuya de forma estructurada a la validación final del sistema.

5.3 Etapas del prototipo

El sistema desarrollado corresponde a un prototipo de invernadero a escala completamente automatizado, diseñado para controlar la iluminación ultravioleta artificial mediante un algoritmo de control PID. La regulación de la radiación UV se realiza en función de las condiciones ambientales medidas en tiempo real, permitiendo simular el comportamiento de un invernadero tecnificado.

La radiación ultravioleta, cuando es aplicada de manera controlada, contribuye al fortalecimiento fisiológico de las plantas, incrementa su resistencia a enfermedades y mejora la calidad del cultivo. Por esta razón, el prototipo se implementa con un enfoque didáctico y funcional, integrando sensores ambientales, una unidad central de procesamiento, dispositivos de actuación eléctrica y elementos de señalización visual.

Esta configuración permite analizar, experimentar y validar estrategias de control automático, facilitando la comprensión del funcionamiento de sistemas de automatización aplicados a la agricultura inteligente.

5.3.1 Arquitectura del sistema.

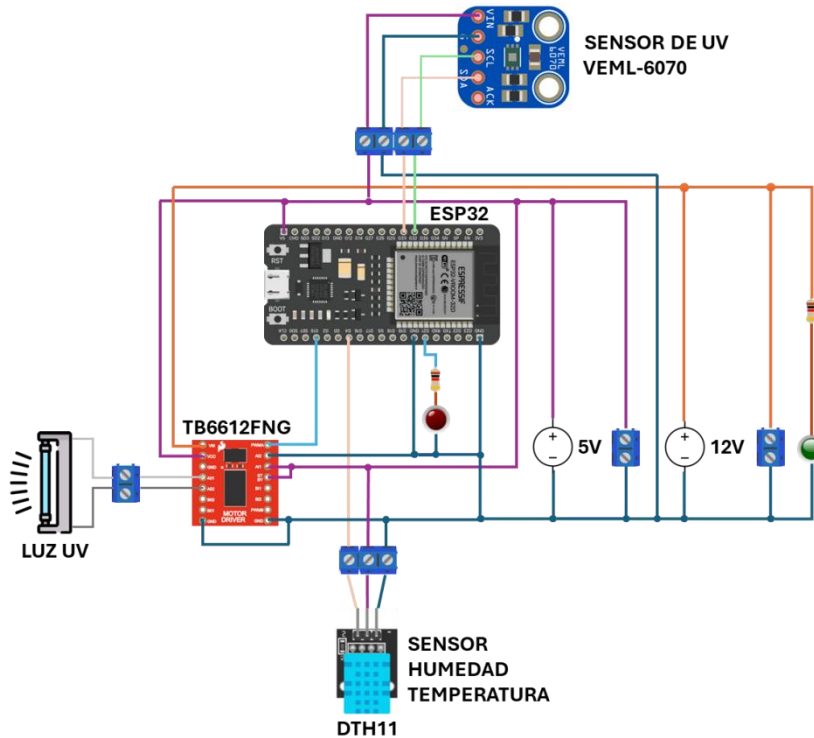
Desde el punto de vista estructural, la arquitectura del sistema se compone de cuatro bloques funcionales principales:

- Bloque de adquisición de datos.
- Bloque de procesamiento y control.
- Bloque de actuación.
- Bloque de señalización y supervisión.

Esta arquitectura modular permite identificar claramente las funciones de cada componente, facilitando el mantenimiento del sistema y la posibilidad de realizar futuras ampliaciones sin necesidad de rediseñar completamente el prototipo.

Figura 12

Diagrama de conexiones entre dispositivos.



Nota. Conexión de componentes electrónicos presentes en arquitectura propuesta.

5.3.1.1 Bloque de adquisición de datos

5.3.1.1.1 Sensor de radiación ultravioleta

El sensor de radiación ultravioleta se encarga de medir la intensidad de radiación UV presente en el interior del invernadero. Este sensor convierte la energía luminosa incidente en una señal eléctrica proporcional, la cual es procesada por el microcontrolador ESP32.

La información proporcionada por este sensor constituye la variable de proceso del sistema de control, permitiendo determinar si el nivel de radiación UV se encuentra por debajo, dentro o por encima del rango óptimo. En función de esta medición, el sistema regula de manera automática la activación de las luminarias UV artificiales.

5.3.1.1.2 Sensor de temperatura y humedad

El sistema incorpora un sensor digital de temperatura y humedad relativa, debido a que estas variables influyen directamente en el desarrollo de las plantas y en la eficiencia del uso de la radiación UV. El empleo de un sensor con comunicación digital reduce la susceptibilidad a ruidos eléctricos y mejora la precisión de las mediciones.

Los datos obtenidos permiten contextualizar las condiciones internas del invernadero y sirven como información complementaria para el análisis del comportamiento global del sistema.

5.3.1.2 Módulos de procesamiento y control

El módulo de procesamiento está dado por un microcontrolador ESP32 que actúa como la unidad central de control del sistema y ha sido elegido por su gran capacidad de procesamiento, bajo consumo y por haber múltiples entradas y salidas analógicas y digitales. Este microcontrolador ejecuta un programa que realiza las principales funciones:

- Inicialización de periféricos y configuración de pines.
- Lectura periódica de los sensores ambientales.
- Conversión y procesamiento de las señales adquiridas.
- Comparación de los valores medidos con umbrales definidos.
- Ejecución de la lógica de control de la iluminación UV.
- Generación de señales de control de los actuadores.
- Gestión de los elementos de señalización visual.

La lógica de control que se ejecuta permite que el sistema funcione de modo autónomo, sin intervención humana, asegurando un funcionamiento continuo y eficiente. Los valores de referencia utilizados por el sistema pueden ser modificados por software, asegurando la adecuación a diferentes escenarios experimentales.

5.3.1.3 Bloque de actuación

El bloque de actuación es responsable de ejecutar físicamente las decisiones generadas por el módulo de procesamiento y control. Para este propósito se emplea un módulo de potencia que actúa como interfaz entre el microcontrolador y las luminarias UV.

Dado que las salidas del microcontrolador operan a bajos niveles de tensión y corriente, se requiere un driver de potencia que permita controlar cargas eléctricas de mayor capacidad. Este módulo protege al microcontrolador frente a sobrecorrientes, cortocircuitos y sobretensiones.

Cuando el sistema detecta que la radiación UV se encuentra por debajo del valor mínimo establecido, el microcontrolador activa las luminarias UV. De manera inversa, cuando el nivel de radiación alcanza o supera el valor de referencia, el sistema reduce o desactiva la iluminación para evitar exposiciones inadecuadas.

5.3.1.4 Bloque de señalización y supervisión

Con la finalidad de facilitar la supervisión del sistema y mejorar el vínculo con el usuario, se ha implementado un bloque de señalización visual basado en LEDs (diodos emisores de luz), los cuales permiten reconocer rápidamente el estado de funcionamiento del sistema.

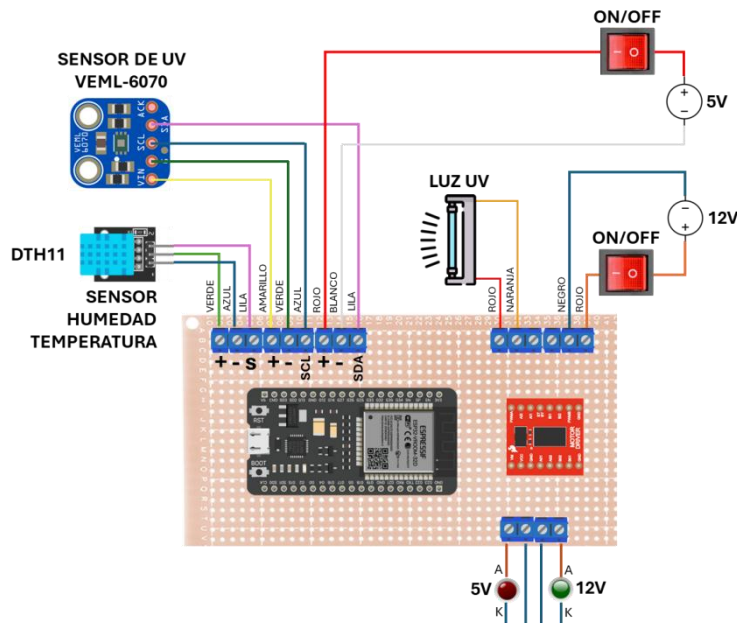
Los estados que pueden ser señalizados mediante los indicadores visuales son:

- Sistema eléctrico y funcionando.
- Activación de la iluminación UV.
- Modo normal de funcionamiento o modos de alerta.

Por su parte, cada LED se conecta con un pin digital de salida del microcontrolador a través de una resistencia limitadora de corriente, de forma tal de garantizar un funcionamiento seguro y duradero del componente. Este tipo de señalización resulta especialmente interesante para aplicación en entornos académicos y de laboratorio en los que resulta necesario validar rápidamente el comportamiento del sistema.

Figura 13

Diagrama de cableado de elementos instalados en el prototipo.



Nota. Baquelita perforada que incluye el controlador lógico y de potencia.

5.3.2 Diseño de Servidor Web

La adopción de un servidor web embebido en el ESP32 obedece a las actuales tendencias recogidas en la Industria 4.0 y la agricultura inteligente, donde las conexiones, el monitoreo en tiempo real y el acceso remoto a los procesos productivos se convierten en elementos necesarios para la optimización de los recursos, la toma de decisiones basadas en datos, etc. El ESP32 funciona como unidad de control y servidor web al mismo tiempo. Para ello, combina las funciones que se indican a continuación:

- Configuración de una red inalámbrica (modo AP o estación Wi-Fi).
- Implementación de un servidor HTTP que escucha las peticiones de las estaciones web clientes.
- Respuesta a las solicitudes que emiten el navegador.
- Envío de datos de sensores en tiempo real.
- Recepción de comandos de control para el sistema de iluminación UV.

Esta arquitectura evita sistemas de servidores externos de tal manera que el sistema es menos complejo y más seguro al hacer que todas las operaciones que tienen que ver con el sistema crítico la ejecute el microcontrolador.

Desde el enfoque de la programación, la primera consideración a tener para implementar el servidor web es la configuración del módulo Wi-Fi del ESP32, para la cual se emplean las librerías del entorno Arduino, las cuales hacen una abstracción del protocolo IEEE 802.11, ocultando detalles de implementación de nivel bajo. El microcontrolador lo puede hacer en dos modos de funcionamiento:

- Modo estación (STA): el ESP32 se conecta a una red Wi-Fi ya existente.
- Modo punto de acceso (AP): el ESP32 genera su propia red Wi-Fi.

El modo punto de acceso es de especial interés en el contexto del invernadero a escala, ya que permite el acceso al sistema desde un dispositivo móvil o computador, desconectado de la infraestructura de red subyacente, dado que la propia red será generada por el ESP32. Después de definir la red, el ESP32 genera direcciones IP a los clientes utilizando DHCP y puede atender requerimientos HTTP.

El servidor web embebido en el ESP32 genera tres pantallas principales que estructuran la interacción entre el usuario y el sistema de control de iluminación UV del invernadero a escala. Cada pantalla cumple una función específica dentro del proceso de supervisión y control, y ha sido diseñada considerando criterios de usabilidad, claridad visual y coherencia con los objetivos académicos del proyecto. A continuación, se procede a listar y describir las pantallas realizadas en el proyecto:

- Pantalla Portada
- Pantalla Panel
- Pantalla Gráficos

5.3.2.1 Pantalla Portada

La pantalla inicial es la primera interfaz del sistema de sistema de monitorización y control del prototipo de invernadero, desarrollada e integrada en el microcontrolador ESP32, que actúa como el punto de acceso WiFi y el servidor Web asíncrono. Esta interfaz se visualiza cuando el usuario se conecta con el dispositivo a la red WiFi que produce el mismo, identificada como ESP32_TESIS, y funciona sobre la dirección IP automática que comporta el acceso a la interfaz.

En el aspecto funcional y como se puede observar en la figura 4, esta interfaz es tanto introductoria como informativa, y sirve para que el usuario tenga un conocimiento general del proyecto y un acceso claro y estructurado a las distintas interfaces que forman parte del sistema.

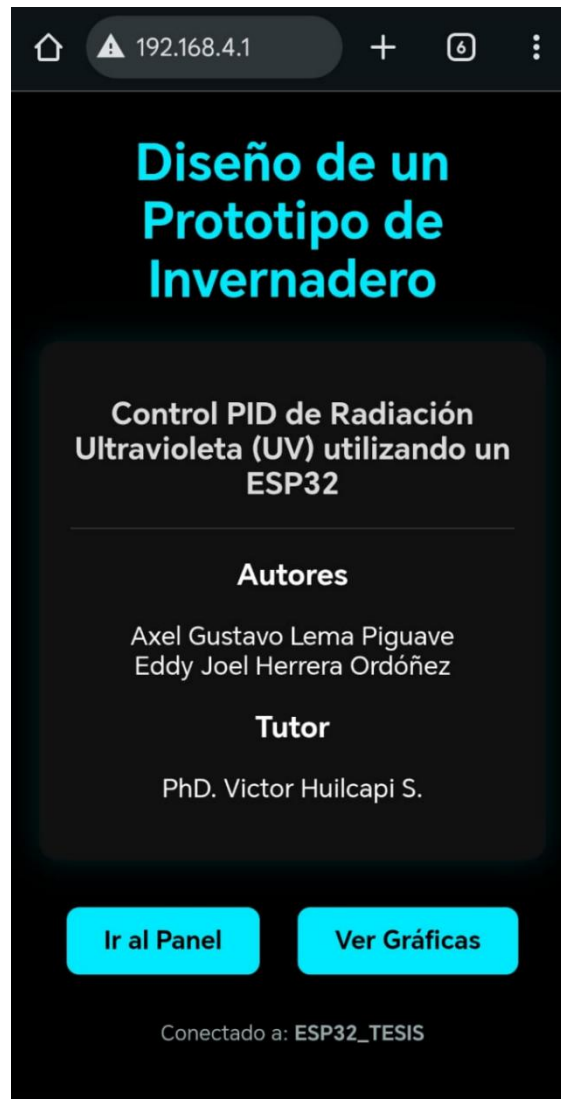
La portada incluye el título de la propuesta de trabajo, que responde al nombre de "Diseño de un Prototipo de Invernadero" y el subtítulo "Control PID de Radiación Ultravioleta (UV) a partir de un ESP32" dan rápidamente un contexto claro a quienes entran en la propuesta de trabajo en líneas generales, ya que detectan desde el inicio el objetivo primigenio del sistema, que no es otro que el del control automático del nivel radiación ultravioleta a partir de la implementación de un controlador PID en hardware embebido.

Además, la propia propuesta de trabajo sirve para identificar los autores del trabajo y sus compañeros o tutores académicos, dando un carácter académico y formal al desarrollo de la propuesta de trabajo que queda expuesta de la siguiente manera, donde puede leerse dentro de una tarjeta que se sitúa en la parte del medio, y cuyo diseño visual es relativamente sobrio, propiciando la lectura y la exposición correcta.

En la parte de la parte inferior queda especificado el estado de conexión de la propuesta de trabajo, haciendo mención del nombre de la red WiFi a la que el usuario se encuentra conectado, de forma que pueda llegar a confirmarse que la conexión con el ESP32 ha sido llevada a cabo con éxito.

Figura 14

Servidor Web ESP32 - Pantalla Portada.



Nota. Diseño de pantalla inicial para aplicación móvil.

Como se observa en la figura 4, la pantalla inicial incluye dos botones útiles que permiten el acceso a las funcionalidades del sistema, en particular:

Botón "Ir al Panel": Este botón dirige a la dirección /panel, donde se encuentra la pantalla de monitorización y control en tiempo real. En esta sección se presentan las variables ambientales que están siendo medidas por los sensores (temperatura, humedad e índice UV) y se incluyen también los controles para modificar el setpoint de radiación UV, así como también para cambiar los parámetros del PID (K_p , K_i y K_d). Esta vista está orientada para la monitorización activa y ajuste del sistema.

Botón "Ver Gráficas": Este botón lleva a la dirección /graficas, donde se presenta la visualización histórica de las variables medidas. En esta pantalla se ofrecen gráficas de la temperatura, de la humedad y del índice UV, en la que se presenta también la comparación de los valores reales (PV) con los valores de referencia (SP) para la radiación UV. Esta funcionalidad permite analizar el comportamiento dinámico del sistema, así como también apreciar la capacidad del PID para controlar el sistema a través del tiempo.

5.3.2.2 Pantalla Panel

El servidor web no solo cumple una función de supervisión, sino que también se integra directamente con el algoritmo de control PID implementado en el ESP32. A través de la interfaz web, el usuario puede modificar parámetros clave del controlador, tales como:

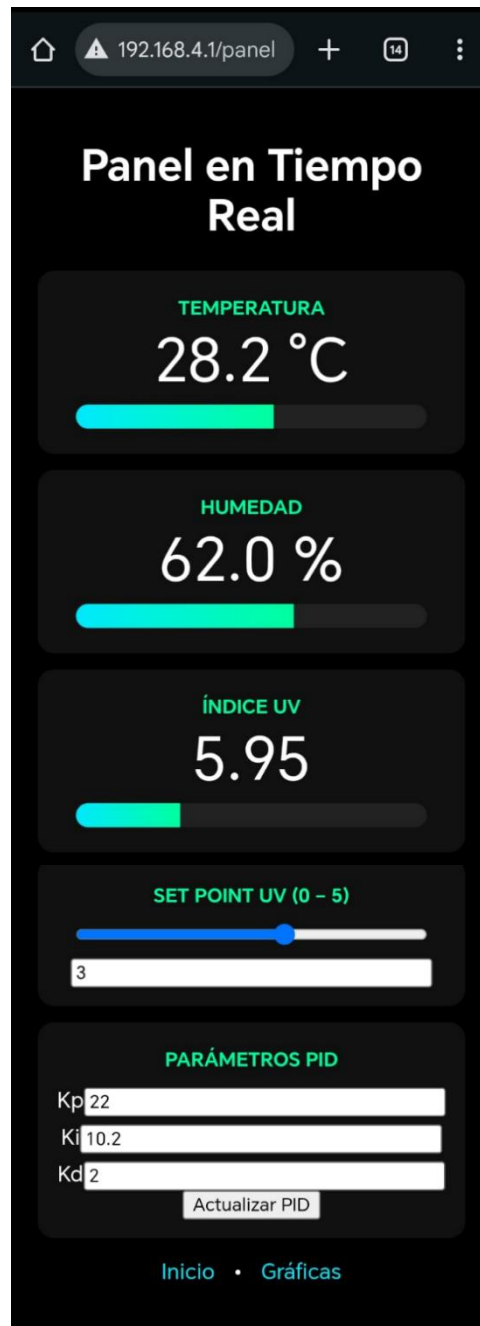
- Valor de referencia (setpoint) de radiación UV.
- Ganancia proporcional (K_p), integral (K_i) y derivativa (K_d).

Estos valores son enviados al microcontrolador mediante solicitudes HTTP y actualizados en tiempo real dentro del programa. De esta manera, el sistema permite una sintonización remota del controlador PID, lo cual resulta especialmente valioso en entornos experimentales y académicos.

En la figura 5 se puede notar el conjunto de funciones que integra la interfaz se diseñan tarjetas (*cards*) que remiten a cada variable o función. Gracias a la variante de la interfaz que propone esta estructura modular, se facilita la lectura de la información, se mejora la comprensión de dicha información y se permite una interacción intuitiva.

Figura 15

Servidor Web ESP32 - Pantalla Panel.



Nota. Diseño de pantalla de medición y parametrización para aplicación móvil.

El sistema modifica los valores mostrados de una forma cíclica por medio de llamadas asíncronas al *endpoint* /datos con un intervalo aproximado de 2 segundos para garantizar y dar un servicio de supervisión "real-time" del estado del invernadero. Las variables ambientales monitoreadas en esta pantalla son las siguientes:

- **Temperatura**

La primera tarjeta corresponde a la temperatura ambiente, medida mediante el sensor DHT11. El valor se presenta en grados Celsius ($^{\circ}\text{C}$) con un decimal de precisión. Adicionalmente, se incluye una barra de progreso animada, cuya longitud es proporcional al valor medido con respecto a un rango máximo predefinido (50°C). La animación de esta barra se actualiza suavemente gracias a transiciones CSS, lo que permite al usuario percibir de forma visual los cambios graduales de la variable.

- **Humedad Relativa**

La segunda tarjeta muestra la humedad relativa del aire, expresada en porcentaje (%). Al igual que la temperatura, se representa mediante un valor numérico y una barra de progreso dinámica que refleja el nivel de humedad en un rango de 0 a 100 %. Ambas variables (temperatura y humedad) cumplen una función informativa y permiten contextualizar el entorno climático del invernadero, aunque no intervienen directamente en el lazo de control PID del sistema UV.

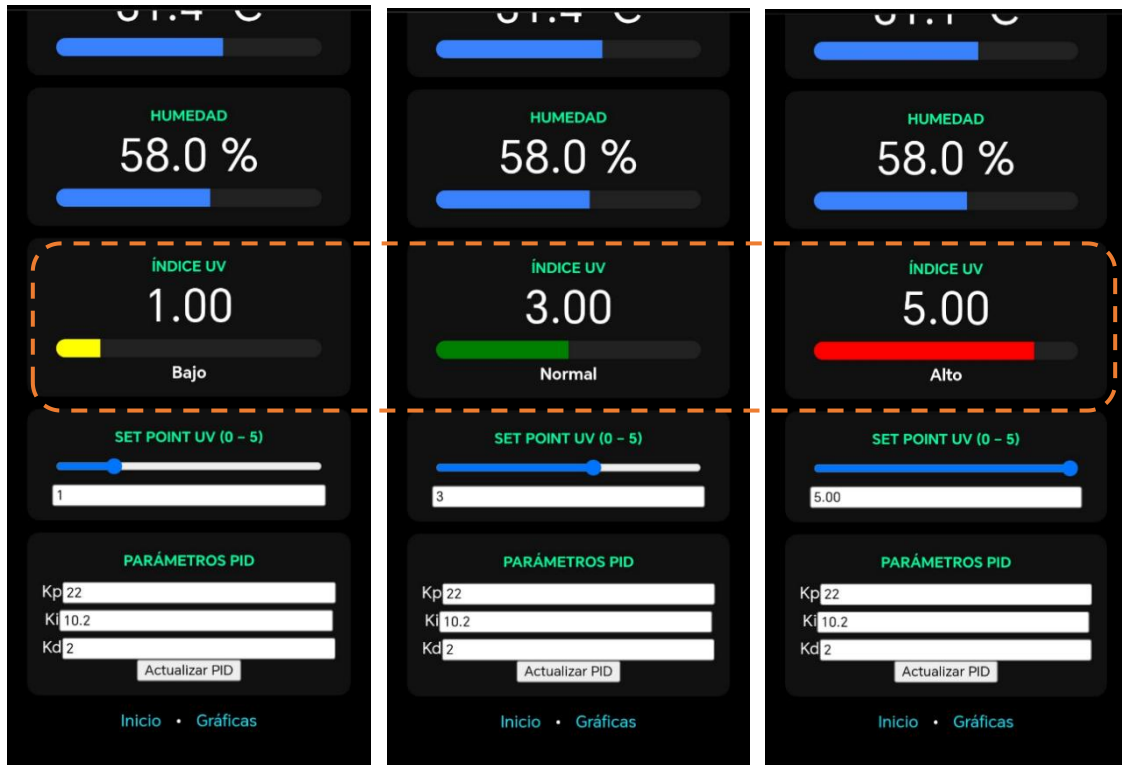
La tercera tarjeta muestra el índice de radiación ultravioleta (UV) que es calculado a partir de las lecturas suministradas por el sensor VEML6070; dicha variable conforma la variable de proceso (PV) del sistema de control.

Como se observa en la figura 16, la interfaz muestra, junto al valor numérico (con una precisión de dos decimales), también una barra de progreso en donde el índice se encuentra asociado a un esquema de codificación en colores; este elemento tiene como objetivo proporcionar un soporte visual inmediato a la interpretación del nivel de radiación UV:

- **Color amarillo:** índice UV bajo ($\text{UV} < 2$), significa que es seguro.
- **Color verde:** índice UV normal ($2 \leq \text{UV} < 4$), significa el rango óptimo de operación.
- **Color rojo:** índice UV alto ($\text{UV} \geq 4$), significa peligro potencial.

Figura 16

Servidor Web ESP32 - Pantalla Panel – índice de UV.



Nota. Ejemplo de varias mediciones tomadas para diversos rangos de UV.

De manera complementaria a la barra de progreso, tal como se muestra en la figura 6, se ha colocado una etiqueta de texto (“Bajo”, “Normal” o “Alto”) que refuerza la interpretación del estado del sistema. Estos cambios de color y de estado se encuentran en perfecta sincronización con la medición recibida, representando una forma de retorno visual muy efectiva.

La cuarta tarjeta proporciona la oportunidad de ajustar el punto de referencia (*setpoint*) del índice UV, que es la referencia (SP) del controlador PID, pudiendo variar este parámetro utilizando dos elementos sincronizados; un control deslizante o slider que acepta valores desde 0 hasta 5 y un campo numérico que permite insertar directamente el valor deseado de forma numérica. Ambos elementos se encuentran interrelacionados, por lo que cualquier cambio en uno de ellos automáticamente se propaga al otro.

Cuando se modifica el valor deseado y se hace clic en el botón “Aplicar Setpoint UV”, el nuevo valor se envía al ESP32 a través de una llamada HTTP al endpoint /setUV y se actualiza la referencia a emplear por parte del algoritmo PID. De este modo,

se hace posible una interacción controlada en este proceso entre el navegador web y el ESP32 (siendo permitido un rango de valores que para que sea válida la acción de descansar un impacto más alto), modificando así en tiempo real el comportamiento del sistema.

Al igual que en la pantalla portada, para la navegación entre pantallas se encuentran disponibles textos con enlaces preconfigurados ubicados en la parte inferior del panel que permiten al usuario desplazarse a otras secciones del sistema:

- **Inicio:** dirigido a la pantalla de la portada del sistema, la que presenta la información general del proyecto.
- **Gráficas:** dirigida a la pantalla de visualización histórica, donde aparecen las curvas con los datos de la temperatura, la humedad, el índice UV, comparando la variable de proceso y el setpoint.

5.3.2.3 Pantalla Gráficos

La pantalla está dividida en secciones independientes, cada una de las cuales representa uno de los elementos a estudiar. Como se aprecia en la figura 17, cada sección incluye un gráfico cartesiano bidimensional, que va acompañado de su correspondiente etiqueta y el valor actual de la variable de proceso (PV).

Los gráficos son generados dinámicamente a través del elemento HTML para luego ser actualizados de forma asíncrona a partir de los datos devueltos por los *endpoints* /datos y /hist, lo que permite, por un lado, conjugar el histórico almacenado como el de tiempo real en un único gráfico.

El eje horizontal de los gráficos representa el tiempo, expuesto en formato de hora real, es decir, en HH:MM:SS y el eje vertical representará la magnitud de la variable correspondiente, con escalas acordes para su rango.

Figura 17

Servidor Web ESP32 - Pantalla de Gráficos.



Nota. Ejemplo de medición de las tres variables monitoreadas.

- **Gráfica de Temperatura (°C)**

La primera gráfica representa la evolución temporal de la temperatura del aire medida con el sensor DHT11, donde resulta conveniente asignar a la variable temperatura un gráfico de línea de color rojo que represente el valor de proceso (PV).

El eje vertical ha sido escalado entre 0 y 60 °C con escalas iguales, lo que permite realizar una fácil interpretación visual del gráfico en relación a las variaciones de temperatura. En la parte superior de la curva se sitúa el valor

instantáneo de temperatura, lo que permite realizar una fácil comparación del estado del proceso con la historia del mismo. Esta gráfica puede ser especialmente útil para la identificación de cambios de temperatura, de estabilidad del ambiente y perturbaciones externas que pudieran tener una influencia indirecta en la radiación de UV.

- **Gráfica de Humedad Relativa (%)**

La segunda gráfica corresponde a la humedad relativa del aire, que, al igual que la temperatura, se mide con el sensor DHT11. La variable en este caso se representa con una curva continua de color azul y se encuentra asociada al valor de proceso (PV).

El eje vertical presenta un intervalo de valores que va de 0 a 100 %, de manera que se pueden apreciar con claridad las oscilaciones que tiene la humedad del aire del invernadero. El valor de la humedad actual, por cierto, queda reflejado de forma numérica sobre el gráfico para poder tener unas lecturas más exactas del estado del sistema en la actualidad.

No obstante que esta variable no interviene de forma directa en el lazo de control PID del sistema del UV, el seguimiento de la humedad relativa resulta importante para llevar a cabo un análisis más global del microclima del invernadero.

- **Gráfica del Índice de Radiación Ultravioleta (UV)**

La figura que aparece en tercer lugar describe el índice de radiación ultravioleta, siendo el elemento de control más importante desde la perspectiva del control automático. En dicha figura se describen dos curvas a la vez:

- **PV (Variable de Proceso):** que aparece como una curva de color amarillento y es el índice UV medido por el sensor VEML6070.
- **SP (Setpoint):** que aparece como una curva de color celeste y señala el valor de la referencia tal y como ha sido definido por el usuario desde la pantalla del panel.

Esta representación que afianza a la vez las dos curvas permite poder medir de una manera clara la forma de comportamiento dinámica del controlador PID, observando la respuesta que da el sistema ante los cambios en el setpoint y fenómenos de sobreimpulso, el tiempo de establecimiento o el error en estado estacionario.

El eje vertical de la figura UV aparece escalado de una manera típica en el rango de 0 a 10 unidades, con unas divisiones cada 2 unidades que permiten perseguir claramente el nivel de radiación. En la parte superior aparecen escritos los valores actuales de PV y de SP, lo cual facilita hacer una comparación inmediata entre la variable controlada y la referencia.

5.3.3 Controlador PID para luz UV

El control de la radiación UV en el invernadero a escala se hace mediante un controlador PID (Proporcional–Integral–Derivativo), que es útil para mantener el índice UV en el valor de referencia deseado, es decir, el setpoint, a pesar de las perturbaciones y de la variación de las condiciones ambientales.

Cabe recordar que en la pantalla Panel, se incluye una sección dedicada a la configuración de los parámetros del controlador PID, específicamente las constantes:

- **Kp:** ganancia proporcional
- **Ki:** ganancia integral
- **Kd:** ganancia derivativa

Cada parámetro se ajusta mediante un campo numérico, permitiendo la sintonización fina del controlador directamente desde la interfaz web. Al presionar el botón “Actualizar PID”, los nuevos valores son enviados al ESP32 a través del *endpoint* */setPID*, donde se aplican inmediatamente al algoritmo de control. Para realizar una programación ordenada y con posibilidades a cambios, se hace el uso de variables globales al inicio de la codificación en el controlador ESP32, esto se puede evidenciar con facilidad en la figura 8.

Figura 18

Variables para control PID en ESP32.

```
// ----- PID UV -----  
float Kp = 22.0;          // ganancia proporcional inicial sugerida  
float Ki = 10.20;         // ganancia integral (por segundo)  
float Kd = 2.0;           // ganancia derivativa  
  
float pidIntegral = 0.0;  
float lastError = 0.0;  
float lastMeasurement = 0.0;  
unsigned long lastPIDMillis = 0;  
  
// límites  
const int PWM_MIN = 0;  
const int PWM_MAX = 255;  
const float INTEGRAL_MIN = -300.0;  
const float INTEGRAL_MAX = 300.0;
```

Nota. Declaración de variables globales disponibles para todas las funciones.

El controlador PID se emplea mucho en las plantas industriales ya que es simple, robusto y eficaz para regular variables continuas. En el sistema desarrollado, el controlador PID actúa sobre la intensidad de las luminarias UV, modulando la potencia mediante una señal PWM generada por la ESP32.

En el contexto del sistema, se definen las siguientes variables fundamentales:

- **Variable de proceso (PV):** Índice de radiación ultravioleta medido por el sensor VEML6070.
- **Setpoint (SP):** Valor de radiación UV deseado, definido por el usuario mediante la interfaz web.
- **Variable manipulada (MV):** Ciclo de trabajo de la señal PWM aplicada a las luces UV.

En la programación, el setpoint se almacena en una variable de tipo flotante y este valor puede ser modificado dinámicamente desde la interfaz web, lo que permite evaluar el comportamiento del controlador ante cambios en la referencia.

Como se puede apreciar en la figura 9, la lógica de control PID ha sido implementada como una función específica **computePID ()** que encierra toda la lógica del control. Recibe como argumentos el setpoint y la medición actual del índice de UV y devuelve un valor de salida correspondiente al nivel de potencia de las luminarias requerido.

Figura 19

Algoritmo de control PID en ESP32.

```
//----- PID -----  
float computePID(float setpoint, float measurement)  
{  
    unsigned long now = millis();  
    float dt = (now - lastPIDMillis) / 1000.0;  
  
    if (dt <= 0.0) return duty;  
    lastPIDMillis = now;  
  
    float error = setpoint - measurement;  
  
    // --- INTEGRAL ---  
    pidIntegral += error * Ki * dt;  
    pidIntegral = constrain(pidIntegral, INTEGRAL_MIN, INTEGRAL_MAX);  
  
    // --- DERIVADA FILTRADA ---  
    float deriv = (measurement - lastMeasurement) / dt;  
    float derivFiltered = derivFilterAlpha * lastDerivFiltered +  
    | | | | | | | | | | (1.0 - derivFilterAlpha) * deriv;  
  
    lastDerivFiltered = derivFiltered;  
    lastMeasurement = measurement;  
  
    // --- PID ---  
    float output = (Kp * error) + pidIntegral - (Kd * derivFiltered);  
  
    // Limitar a PWM  
    output = constrain(output, PWM_MIN, PWM_MAX);  
  
    return output;  
}
```

Nota. Estructura de control PID utilizada en el controlador ESP32.

La acción proporcional genera una salida directamente proporcional al error instantáneo del sistema. En el controlador implementado, esta acción se define mediante la constante K_p . La acción proporcional permite una respuesta rápida del sistema ante

cambios en el error. Sin embargo, un valor excesivamente alto de K_p puede provocar oscilaciones, mientras que un valor bajo puede generar una respuesta lenta.

La acción integral tiene como objetivo eliminar el error en estado estacionario, acumulando el error a lo largo del tiempo. La constante K_i determina la velocidad con la que el controlador corrige errores persistentes.

La acción derivativa anticipa el comportamiento futuro del sistema, evaluando la velocidad de cambio de la variable de proceso. En el controlador implementado, la derivada se calcula con base en la medición, reduciendo la sensibilidad al ruido. Adicionalmente, se aplica un **filtro de primer orden** para suavizar la señal derivativa. Este filtrado resulta especialmente relevante debido a la naturaleza ruidosa de las mediciones de radiación UV. La constante *derivFilterAlpha* permite ajustar el nivel de suavizado, mejorando la estabilidad del controlador.

Figura 20

Llamado de algoritmo de control PID desde el lazo principal del ESP32.

```
// ----- LOOP -----  
void loop() {  
  
    unsigned long now = millis();  
  
    // --- Lectura del sensor VEML6070 ---  
    uint16_t uvRaw = uv.readUV();    // lectura del sensor  
    cachedUV = uvRaw * 0.05;        // escala típica → índice UV aproximado, 0.005  
  
    if (now - lastSamplePID >= SAMPLE_INTERVAL_PID) {  
        lastSamplePID = now;  
  
        // ---- PID ----  
        duty = computePID(uvSetpoint, cachedUV);  
        analogWrite(pwmPin, duty);  
  
        Serial.print("SP: "); Serial.print(uvSetpoint, 2);  
        Serial.print(" | PV: "); Serial.print(cachedUV, 2);  
        Serial.print(" | PWM: "); Serial.println(duty);  
    }  
}
```

Nota. Codificación de función principal para control de UV, lectura de sensores y visualización a través de puerto serial.

La figura 20 ilustra llamado del controlador PID desde el *void loop* y para que este funcione de la manera adecuada, es necesario que se considere el tiempo transcurrido entre dos ejecuciones continuas del propio algoritmo. En el sistema que se ha desarrollado, el tiempo transcurrido entre dos ejecuciones se genera mediante la función *millis()* del microcontrolador. La variable *dt* representa el tiempo de muestreo en segundos y permite que los términos integral y derivativo sean calculados de forma consistente, independientemente de pequeñas variaciones en el tiempo de ejecución del programa.

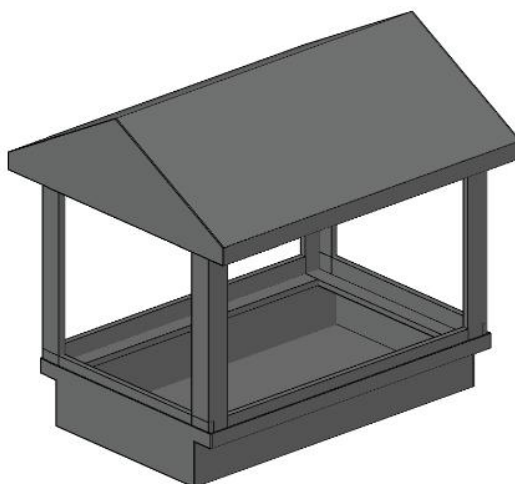
6. RESULTADOS

6.1. Implementación del prototipo

En esta sección se presenta el resultado del trabajo realizado. De forma más concreta, se va a establecer el prototipo de invernadero desarrollado en el presente proyecto de trabajo con una función primordial: la de ser una plataforma experimental mediante la medición, control y regulación de la radiación ultravioleta (UV) del invernadero, a partir de un controlador PID desarrollado mediante un microcontrolador ESP32, ya que el invernadero se ha diseñado a escala reducida y permite verificar el comportamiento del sistema de control, los sensores y los elementos de control implicados. Con ello se da cumplimiento a una funcionalidad que hace del prototipo un modelo representativo de un invernadero real.

Figura 21

Diseño CAD 3D del invernadero.



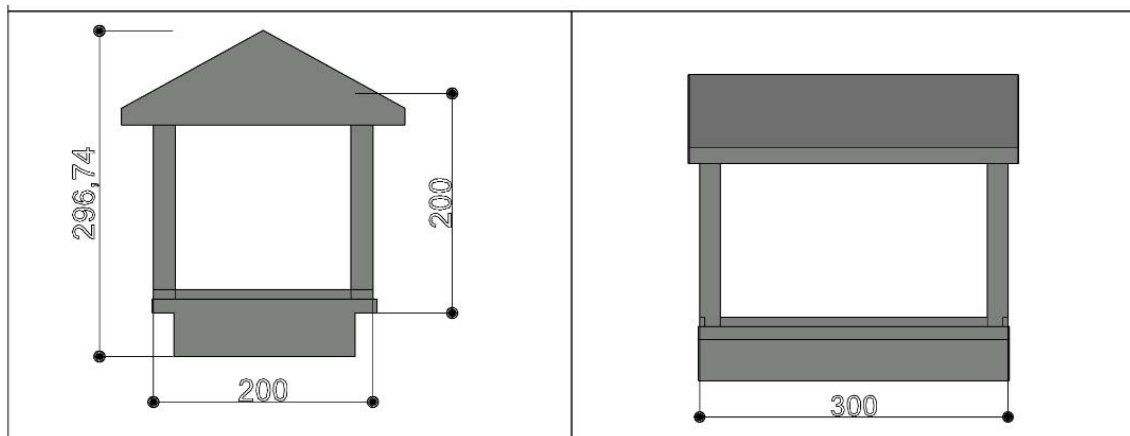
Nota. Prototipo a escala del invernadero

Desde el punto de vista estructural y como se puede apreciar en la figura 21, el invernadero tiene una morfología prismática con cubierta a dos aguas, que es la forma de invernadero más utilizada para los sistemas agrícolas protegidos, dado que es la que permite una mejor distribución de la radiación y la evacuación de la condensación. En primer lugar, el diseño fue modelado a través de un programa tipo CAD, lo que permitió definir las dimensiones, la geometría de los elementos y la disposición interna de los componentes electrónicos y mecánicos. En la figura 22 se puede apreciar las dimensiones aproximadas en milímetros lo cual se realizó previa a la construcción de la misma. El material que se utilizó en la estructura es acero inoxidable AISI 304 de

1.5mm de espesor, lo que permite tener una mejor estética y a la vez la robustez necesaria para pruebas con diferente humedad. Este material asegura la estabilidad y resistencia estructural y ante la oxidación incluso en condiciones de operación prolongadas.

Figura 22

Cotas del prototipo 3D del invernadero.



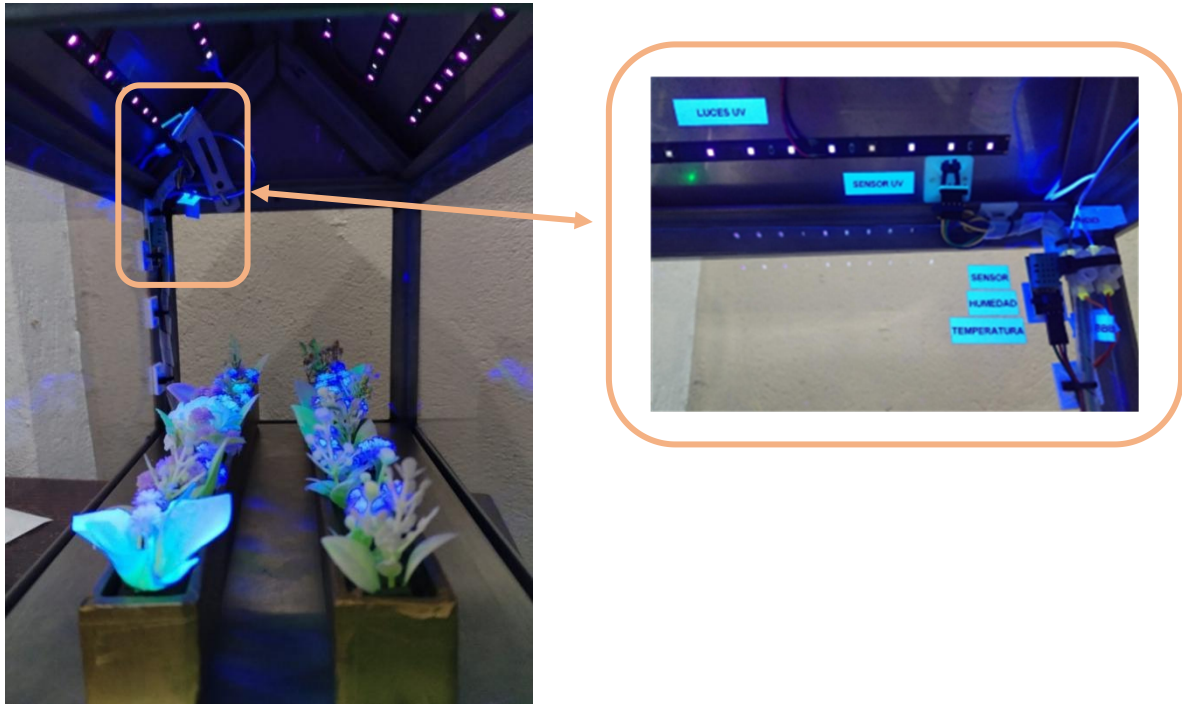
Nota. Dimensiones del prototipo

La cubierta inclinada tiene una doble función: proteger los componentes internos y facilitar la adecuada dirección y distribución de la irradiancia UV emitida por las luminarias LED en el interior del invernadero.

Como se observa en la figura 23, se ha incorporado un sensor de radiación ultravioleta (sensor UV), que se ha colocado dentro del volumen del invernadero, y que permite medir en tiempo real el nivel de radiación que incide sobre las plantas mediante el uso de las tiras de LEDs UV, datos que se utilizarán como variable de proceso para el lazo de control con la que se comparará, a su vez, con un valor de la variable de proceso previamente fijado.

Figura 23

Detalle de sensores y actuadores instalados en el prototipo.



Nota. Elementos de entrada y salida montados en el invernadero a escala.

Además, el prototipo incluye sensores ambientales adicionales (sensor de temperatura, sensor de humedad relativa, etc.) para medir las condiciones internas del invernadero, si bien no forman parte del lazo de control PID de radiación UV; estos sensores también recogen información del microclima y pueden servir, en un futuro, como medios para llevar a cabo ampliaciones del sistema de control.

En la figura 24, específicamente en la parte inferior frontal del prototipo se encuentra un panel de control físico donde aparecen interruptores para la activación de las diferentes tensiones de alimentación (12 VDC y 5 VDC) así como unas luminiscencias tipo LED donde el usuario puede observar el estado del sistema, la presencia de alimentación y la activación del control por pulsos. Esta organización adecúa la operación, supervisión y mantenimiento del prototipo en el transcurso de las pruebas experimentales.

Figura 24

Detalle de sensores y actuadores instalados en el prototipo.



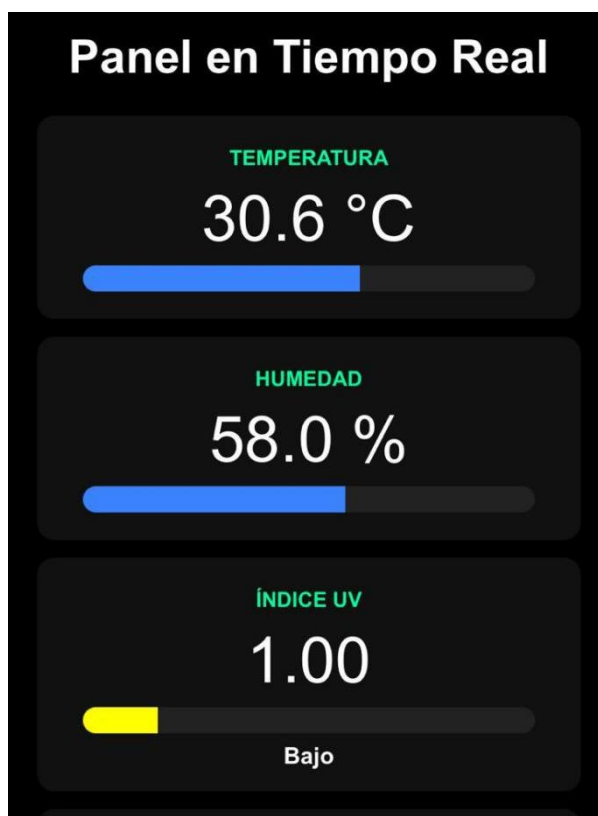
Nota. Panel frontal para control de encendido y apagado.

6.2. Resultados del sistema de adquisición de datos

El sistema de adquisición de datos presentó un desempeño confiable durante todas las pruebas experimentales. Las variables ambientales monitoreadas, correspondientes al índice de radiación ultravioleta, temperatura y humedad relativa, fueron registradas de manera continua y consistente, permitiendo una adecuada supervisión del microclima dentro del invernadero a escala.

Figura 25.

Visualización en tiempo real de las variables ambientales medidas.



Nota. En esta figura se observa datos reales, comunicación estable, funcionamiento del sensor

En particular, la medición del índice de radiación UV demostró ser estable, sin presencia de ruido significativo ni pérdidas de señal. Este comportamiento permitió disponer de una señal de retroalimentación adecuada para el controlador PID, garantizando un control eficiente de la variable de proceso. Las lecturas obtenidas se visualizaron correctamente en tiempo real a través de la interfaz web, como se observa en la Figura 15, lo que facilitó la validación experimental del sistema.

Por otro lado, las variables de temperatura y humedad relativa permitieron contextualizar el comportamiento de la radiación UV dentro del invernadero. Aunque estas variables no fueron controladas de manera automática, su monitoreo resultó útil para analizar el entorno interno del prototipo durante las pruebas. En general, el sistema de adquisición de datos demostró ser adecuado para aplicaciones de monitoreo ambiental en sistemas de agricultura protegida.

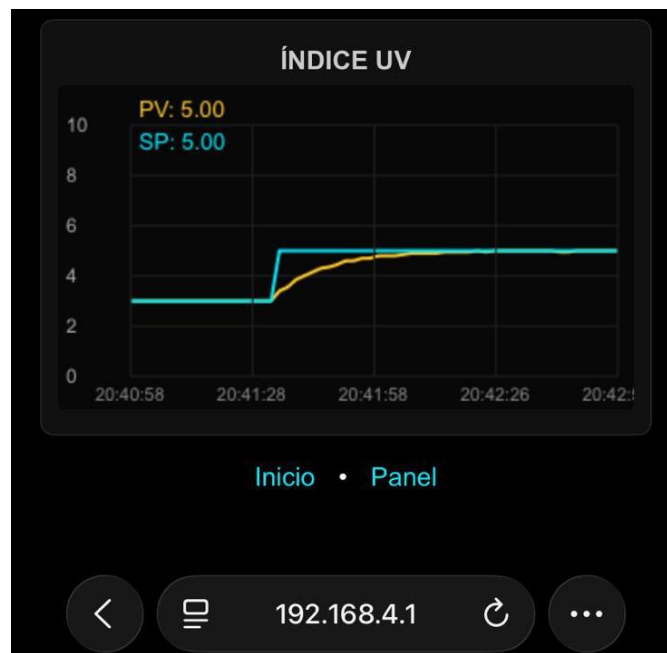
6.3. Resultados del control PID aplicado a la radiación UV

El resultado más relevante del proyecto corresponde al desempeño del controlador PID implementado para la regulación automática de la radiación ultravioleta. Durante las pruebas experimentales, el sistema fue capaz de mantener el índice UV cercano al valor de referencia definido por el usuario, incluso ante variaciones intencionales en las condiciones de iluminación.

El comportamiento dinámico del sistema evidencia una reducción progresiva del error entre la variable de proceso (PV) y el valor de referencia (SP). En la Figura 26 se muestra el comportamiento temporal del índice UV, donde se observa un seguimiento adecuado del setpoint, con oscilaciones controladas y sin sobre impulsos significativos.

Figura 26.

Respuesta al sistema de control PID



Nota. Seguimiento del índice UV respecto al setpoint

La respuesta del sistema permitió evidenciar que el controlador PID contribuyó a mejorar la estabilidad del proceso, reduciendo el error en estado estacionario y manteniendo condiciones de operación dentro de rangos aceptables. Estos resultados confirman que el uso de un controlador PID es una alternativa viable para la regulación de la radiación UV en invernaderos, incluso cuando se implementa sobre plataformas de bajo costo como el ESP32.

Figura 27.

Ajuste de parámetros de controlador PID desde la interfaz web

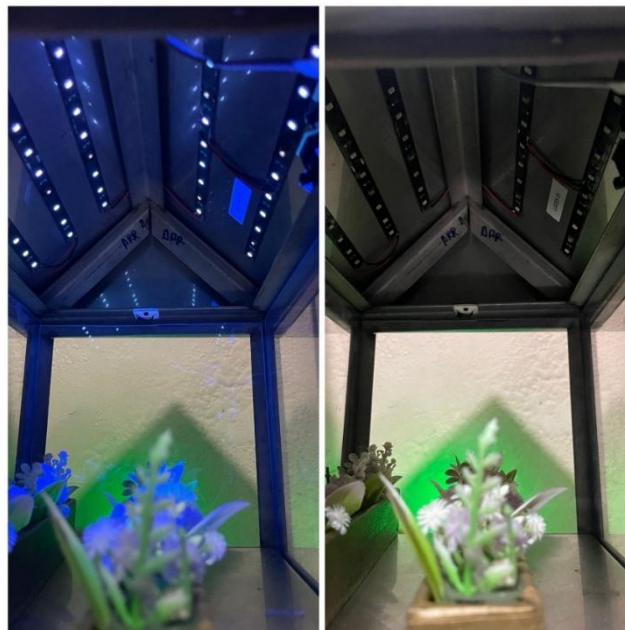


6.4. Resultados del bloque de actuación

El bloque de actuación respondió de manera adecuada a las señales generadas por el controlador PID. Durante las pruebas, se observó que las luminarias UV se activaron o desactivaron en función del valor de la radiación ultravioleta medida, permitiendo una regulación automática de esta variable ambiental.

Figura 28.

Estados de funcionamiento del sistema de iluminación UV.



Nota. Secuencia de LEDs UV apagados vs encendidos

Cuando el índice UV se encontraba por debajo del valor de referencia, el sistema incrementó la activación de las luminarias UV; por el contrario, cuando se alcanzaron valores cercanos o superiores al setpoint, la actuación se redujo progresivamente. Este comportamiento se evidencia en la Figura 24, donde se observa el estado de los actuadores durante el funcionamiento del sistema.

El correcto desempeño del bloque de actuación demuestra que la interacción entre el sistema de control y los elementos de potencia fue adecuada, garantizando una respuesta eficiente y segura. Este resultado es fundamental para validar la aplicación del sistema en escenarios reales donde se requiere un control preciso de la radiación UV.

6.5. Resultados del sistema de supervisión y visualización

El sistema de supervisión basado en una interfaz web permitió un monitoreo eficiente del prototipo durante todas las pruebas experimentales. La interfaz facilitó la visualización en tiempo real de las variables ambientales y del estado del controlador PID, lo que permitió evaluar el comportamiento del sistema de manera continua.

La pantalla principal del sistema, mostrada en la Figura 14, permitió un acceso rápido a la información general, mientras que el panel de control, presentado en la Figura 15, facilitó la supervisión detallada de las variables medidas. Adicionalmente, la pantalla de gráficos, mostrada en la Figura 17, permitió analizar la evolución temporal del índice UV y su relación con el valor de referencia.

Estas funcionalidades resultaron fundamentales para validar experimentalmente el desempeño del sistema de control, ya que permitieron identificar de forma visual la estabilidad y efectividad del controlador PID implementado.

6.6. Análisis global de resultados

A partir de los resultados obtenidos, se evidencia que el sistema desarrollado cumple con los objetivos planteados en el proyecto. El prototipo fue capaz de regular automáticamente la radiación ultravioleta, manteniendo condiciones controladas dentro del invernadero a escala durante las pruebas realizadas.

La estabilidad observada en el sistema, junto con la correcta integración de los distintos subsistemas, demuestra la viabilidad de aplicar técnicas de control automático en entornos de agricultura protegida. Asimismo, el uso de plataformas de bajo costo

representa una alternativa accesible para el desarrollo de soluciones tecnológicas en este ámbito.

7. CRONOGRAMA DE ACTIVIDADES

A continuación, en la Tabla 1 se muestra un cronograma detallado de las actividades a realizar en este proyecto de titulación.

Tabla 1

Cronograma de de actividades del proyecto

ACTIVIDADES	OCTUBRE				OCTUBRE				NOVIEMBRE				DICIEMBRE			
FASE.1: Fase de Investigación y Revisión Bibliográfica																
FASE.2: Selección de Componentes																
FASE.3: Diseño del Prototipo																
FASE.4: Desarrollo del Software																
FASE.5: Pruebas y Validación del Prototipo																
FASE.6: Análisis de Resultados																
FASE7: Documentación y Presentación de Resultados																
FASE8: Recomendaciones y Futuras Investigaciones																

Nota. En esta tabla se desglosa de manera semanal cada actividad que se llevó a cabo durante los tres meses mencionados.

8. PRESUPUESTO

A continuación, en la Tabla 2 se muestra un presupuesto desglosado para el proyecto de creación de un prototipo de control de luz UV para invernaderos. Este presupuesto abarca los costos de materiales, equipos, horas de ingeniería y otros gastos relacionados.

Tabla 2
Presupuesto general del proyecto

DESCRIPCION	CANTIDAD	COSTO UNITARIO(USD)	COSTO TOTAL(USD)
Materiales y Equipos			
Sensor de Radiación UV (VEML6075)	1	10.00	10.00
Sensor de Temperatura y Humedad (DHT22)	1	15.00	15.00
Microcontrolador ESP32	1	8.00	8.00
Lámpara UV LED regulable	2	20.00	40.00
Placa de prototipado (Breadboard)	1	5.00	5.00
Cables y conectores		10.00	10.00
Fuente de alimentación	1	10.00	10.00
Horas de Ingeniería	80	2.94	235.20
Total del Proyecto			333.20

Nota. En esta tabla se desglosa el presupuesto general del proyecto

9. CONCLUSIONES

- El desarrollo e implementación del prototipo de invernadero automatizado demostró que el uso de un microcontrolador ESP32 junto con un algoritmo de control PID permite regular de manera eficiente la radiación ultravioleta (UV), manteniéndola dentro de rangos óptimos para el desarrollo de los cultivos. Esto valida la viabilidad técnica del sistema propuesto y su aplicabilidad en entornos agrícolas controlados.
- La integración de sensores ambientales y un dashboard web para la supervisión en tiempo real facilitó el monitoreo y ajuste de las variables del invernadero, evidenciando que la automatización y el acceso remoto mejoran la gestión del cultivo y reducen la dependencia de la intervención manual, aportando significativamente al concepto de agricultura inteligente.

10. RECOMENDACIONES

- Para trabajos futuros, se recomienda ampliar el sistema de control incorporando otras variables ambientales, como la concentración de CO₂ y el control automático de riego, con el fin de lograr una gestión integral del microclima del invernadero y mejorar aún más la productividad agrícola.
- Se sugiere realizar pruebas del prototipo a mayor escala y durante periodos prolongados, utilizando diferentes tipos de cultivos, con el objetivo de evaluar el desempeño del control PID en condiciones reales y optimizar los parámetros del sistema para aplicaciones comerciales o industriales.

11. REFERENCIAS BIBLIOGRÁFICAS

- Espressif Systems. (2023). *ESP32 Series Datasheet*. Obtenido de Espressif Systems: <https://www.espressif.com/en/products/socs/esp32>
- FAO. (2023). *Protected cultivation: Greenhouses and tunnels*. Obtenido de Food and Agriculture Organization of the United Nations: <https://www.fao.org/>
- INVT. (10 de Abril de 2020). *INVT*. Obtenido de INVT: https://www.invt.com/uploads/file1/20200410/SV-DA200%20Series%20AC%20servo%20drive_V2.2.pdf
- López, M. (29 de Agosto de 2024). *Automatización agrícola con sistemas inteligentes*. Alfaomega. Obtenido de Inspenet: <https://inspenet.com/articulo/aplicaciones-y-ventajas-de-un-servomotor/>
- Lycoskoufis, I. K. (2022). *Ultraviolet Radiation Management in Greenhouse to Improve Red Lettuce Quality and Yield*. Obtenido de <https://doi.org/10.3390/agriculture12101620>
- Ogata, K. (2021). *Ingeniería de control moderno (6.ª ed.)*. Pearson Educación.
- Promelsa. (22 de Noviembre de 2023). *Promelsa*. Obtenido de Promelsa: <https://www.promelsa.com.pe/blog/post/que-es-motor-electrico-tipos.html?srltid=AfmBOoqJkm-Rt5BYC2Wx1X1OoGHnVwrOZ2jhM0ke6PZI-N2vLi8zIYkG>
- Serrano, J. (2022). *Sensores y control de clima en invernaderos inteligentes*. Marcombo.
- Ansari, S., Ansari, A., Kumar, A., Kumar, R., & Nyamasvisva, E. T. (2023). Environmental Temperature and Humidity Monitoring at Agricultural Farms using Internet of Things & DHT22-Sensor. *Journal of Independent Studies and Research Computing*, 21(2). <https://doi.org/10.31645/JISRC.23.21.2.5>
- Łukasiak, J. M., Paś, J., & Rosiński, A. (2024). Selected Reliability Aspects Related to the Power Supply of Security Systems. *Energies*, 17(15), 3665. <https://doi.org/10.3390/en17153665>
- Serrano, A., Abril-Gago, J., & García-Orellana, C. J. (2022). Development of a Low-Cost Device for Measuring Ultraviolet Solar Radiation. *Frontiers in Environmental Science*, 9. <https://doi.org/10.3389/fenvs.2021.737875>
- Shao, Z. L. (2014). Research on 5V Internal Power Supply Circuit of Switching Power Supply. *Applied Mechanics and Materials*, 571–572, 950–954. <https://doi.org/10.4028/www.scientific.net/AMM.571-572.950>
- Toshiba Electronic Devices & Storage Corporation. (n.d.). *Toshiba Electronic Devices & Storage Corporation – TB6612FNG (Brushed DC Motor Driver IC)*.

12. ANEXOS

12.1 PROGRAMACIÓN ESP32

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "DHT.h"
#include <Wire.h>
#include <Adafruit_VEML6070.h>

// ----- CONFIG -----
const char* AP_SSID = "ESP32_TESIS";
const char* AP_PASS = "12345678";
const int pwmPin = 15;
short blinkPin = 21;
float uvSetpoint = 3;      // valor del slider
long cont_blink = 0;
int duty = 0;

// ----- PID UV -----
float Kp = 5.0;           // ganancia proporcional inicial sugerida
float Ki = 10.0;          // ganancia integral (por segundo)
float Kd = 2.0;           // ganancia derivativa

float pidIntegral = 0.0;
float lastError = 0.0;
float lastMeasurement = 0.0;
unsigned long lastPIDMillis = 0;

// límites
const int PWM_MIN = 0;
const int PWM_MAX = 255;
const float INTEGRAL_MIN = -300.0;
const float INTEGRAL_MAX = 300.0;

// filtro derivada (alpha 0..1, 0=noisy, 1=no deriv)
const float derivFilterAlpha = 0.6;
float lastDerivFiltered = 0.0;

#define DHTPIN 4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
```

```

AsyncWebServer server(80);
Adafruit_VEML6070 uv = Adafruit_VEML6070();

// ----- CACHE DE SENSORES -----
volatile float cachedTemp = 0.0;
volatile float cachedHum = 0.0;
volatile float cachedUV = 0.0; // cambiado a float para almacenar índice UV con
decimales

unsigned long lastSample = 0;
unsigned long lastSamplePID = 0;
const unsigned long SAMPLE_INTERVAL = 500UL;
const unsigned long SAMPLE_INTERVAL_PID = 10UL;

const String logoBase64 = "";

// ===== HISTORICOS PARA LAS GRAFICAS =====
#define HIST 60
float histTemp[HIST];
float histHum[HIST];
float histUV[HIST];
float histSP[HIST];
int histIndex = 0;

// ----- HTML: PAGINA INICIO -----
const char index_html[] PROGMEM = R"rawliteral(
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <style>
    body{background:#000;color:#fff;font-family:Arial,sans-
serif;margin:0;padding:18px;text-align:center;}
    .card{background:#111;padding:20px;border-radius:12px;margin:20px auto;max-
width:600px;box-shadow:0 0 20px rgba(0,234,255,0.15);}
    h1 {color:#00eaff;margin:10px 0;}
    h2 {font-size:1.2rem;color:#ddd;}
    a.btn{display:inline-block;margin:10px;padding:12px
24px;background:#00eaff;color:#000;border-radius:8px;text-decoration:none;font-
weight:bold;transition:0.3s;}
    a.btn:hover{background:#fff;box-shadow:0 0 10px #00eaff;}
    p.small{color:#9aa;font-size:13px;margin-top:20px;}
  </style>
</head>
<body>
  <h1>Diseño de un Prototipo de Invernadero</h1>
  <div class="card">
    <h2>Control PID de Radiación Ultravioleta (UV) utilizando un ESP32</h2>
    <hr style="border:0;border-top:1px solid #333;margin:15px 0;">

```

```

        <h3>Autores</h3>
        <p>Axel Gustavo Lema Piguave<br>Eddy Joel Herrera Ordóñez</p>
        <h3>Tutor</h3>
        <p>PhD. Victor Huilcapi S.</p>
    </div>
    <div>
        <a class="btn" href="/panel">Ir al Panel</a>
        <a class="btn" href="/graficas">Ver Gráficas</a>
    </div>
    <p class="small">Conectado a: <strong>ESP32_TESIS</strong></p>
</body>
</html>
)rawliteral";

// ----- HTML: PAGINA PANEL -----
const char panel_html[] PROGMEM = R"rawliteral(
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Panel - Invernadero UV</title>

<style>
body{background:#000;color:#fff;font-family:Arial;text-align:center;padding:12px;}
.card{background:#111;padding:14px;border-radius:12px;margin:12px auto;max-
width:600px;}
.val{font-size:44px;color:#fff;margin:6px 0;}
.label{color:#00ff9f;margin:6px 0;font-weight:700;text-transform:uppercase;font-
size:0.9rem;}

.bar-cont{background:#222;border-radius:10px;height:18px;width:90%;margin:5px
auto;overflow:hidden;}
.bar{height:100%;width:0%;transition:width 0.5s, background-color 0.5s;}

#bt{background:#3b82ff;}
#bh{background:#3b82ff;}
#bu{background:#ffd000;}

#uvStatus{margin-top:6px;font-weight:bold;}

input[type=range], input[type=number]{width:90%;margin-top:6px;}
a{color:#00eaff;text-decoration:none;margin:0 10px;}
</style>
</head>

<body>
<h1>Panel en Tiempo Real</h1>

<div class="card">

```

```

<div class="label">Temperatura</div>
<div class="val" id="temp">-- °C</div>
<div class="bar-cont"><div class="bar" id="bt"></div></div>
</div>

<div class="card">
  <div class="label">Humedad</div>
  <div class="val" id="hum">-- %</div>
  <div class="bar-cont"><div class="bar" id="bh"></div></div>
</div>

<div class="card">
  <div class="label">ÍNDICE UV</div>
  <div class="val" id="uv">--</div>

  <div class="bar-cont">
    <div class="bar" id="bu"></div>
  </div>

  <div id="uvStatus">--</div>
</div>

<!-- ===== SET POINT PARA CONTROL UV ENTERO ===== -->
<div class="card">
  <div class="label">Set Point UV (0 – 5)</div>

  <input type="range" id="uvSlider" min="0" max="5" step="1" value="0">
  <input type="number" id="uvNumber" min="0" max="5" step="1" value="0">
  <button id="btnSP" onclick="aplicarSetpoint()" disabled>
    Aplicar Setpoint UV
  </button>
</div>

<script>
const slider = document.getElementById("uvSlider");
const number = document.getElementById("uvNumber");

slider.oninput = () => {
  number.value = slider.value;
  document.getElementById("btnSP").disabled = false;
};

number.oninput = () => {
  let v = parseInt(number.value);
  if (v < 0) v = 0;
  if (v > 6) v = 6;
  number.value = v;
  slider.value = v;
};

```

```

number.addEventListener('input', () => {
  document.getElementById('btnSP').disabled =
    number.value.trim() === '';
});

function aplicarSetpoint(){
  let sp = parseFloat(number.value);

  // Validaciones de seguridad
  if(isNaN(sp)){
    alert("Ingrese un valor válido de Setpoint UV");
    return;
  }

  if(sp < 0) sp = 0;
  if(sp > 15) sp = 15;    // ajusta a tu rango real

  // sincronizar controles
  slider.value = sp;
  number.value = sp;

  // enviar al ESP32
  fetch(`/setUV?valor=${sp}`);
}

</script>

<script>
async function cargarSetpoint(){
  try{
    const r = await fetch('/getSetpoint');
    const sp = await r.text();

    slider.value = sp;
    number.value = sp;
  }catch(e){}
}

window.addEventListener('load', cargarSetpoint);
</script>

<!-- ===== PID ===== -->
<div class="card">
  <div class="label">Parámetros PID</div>

  <label>Kp</label><input type="number" id="kp" step="0.1"><br>
  <label>Ki</label><input type="number" id="ki" step="0.1"><br>
  <label>Kd</label><input type="number" id="kd" step="0.1"><br>

```

```

    <button onclick="updatePID()">Actualizar PID</button>
</div>

<script>
function updatePID(){
    fetch(`/setPID?kp=${kp.value}&ki=${ki.value}&kd=${kd.value}`);
}

async function up(){
    try{
        const r = await fetch('/datos');
        const d = await r.json();

        temp.innerText = d.t.toFixed(1) + ' °C';
        hum.innerText = d.h.toFixed(1) + ' %';
        uv.innerText = d.u.toFixed(2);

        bt.style.width = Math.min(100,(d.t/50)*100)+'%';
        bh.style.width = Math.min(100,(d.h/100)*100)+'%';

        // --- UV BAR + COLOR ---
        let pct = Math.min(100,(d.u/6)*100);
        bu.style.width = pct + "%";

        if(d.u < 2){
            bu.style.backgroundColor = "yellow";
            uvStatus.innerText = "Bajo";
        } else if(d.u < 4){
            bu.style.backgroundColor = "green";
            uvStatus.innerText = "Normal";
        } else{
            bu.style.backgroundColor = "red";
            uvStatus.innerText = "Alto";
        }
    }

    } catch(e){}
}

setInterval(up,2000);
window.onload = up;
</script>

<script>
async function cargarPID(){
    try{
        const r = await fetch('/getPID');
        const d = await r.json();

        document.getElementById("kp").value = d.kp;
        document.getElementById("ki").value = d.ki;
    }
}

```

```

        document.getElementById("kd").value = d.kd;
    } catch(e) {}
}

window.addEventListener('load', cargarPID);
</script>

<p><a href="/">Inicio</a> • <a href="/graficas">Gráficas</a></p>

</body>
</html>
)rawliteral";

// ----- HTML: PAGINA GRAFICAS -----
const char graphs_html[] PROGMEM = R"rawliteral(
<!doctype html>
<html>
<head><meta charset="utf-8"><meta name="viewport" content="width=device-
width,initial-scale=1">
<title>Gráficas</title>
<style>
    body {background:#000;color:#fff;font-family:Arial;text-align:center;padding:10px;}
    .chart-box {background:#111;padding:10px;border-radius:8px;margin:10px auto;max-
width:800px;border:1px solid #222;}
    canvas {width:100%;height:200px;background:#080808;border-radius:4px;}
    h3 {margin:5px 0;color:#ccc;font-size:1rem;}
    a {color:#00eaff;text-decoration:none;margin:0 10px;}
</style>
</head>
<body>
    <h1>Gráficas Históricas</h1>

    <div class="chart-box"><h3>Temperatura (°C)</h3><canvas
id="cT"></canvas></div>
    <div class="chart-box"><h3>Humedad (%)</h3><canvas id="cH"></canvas></div>
    <div class="chart-box"><h3>ÍNDICE UV</h3><canvas id="cU"></canvas></div>

<p><a href="/">Inicio</a> • <a href="/panel">Panel</a></p>

<script>
    let histLoaded = false;
    const time = []; // ← BUFFER DE TIEMPO (HH:MM:SS)

    function fmtTime(d){
        return d.toLocaleTimeString('es-EC', {
            hour12: false,
            hour: '2-digit',
            minute: '2-digit',
            second: '2-digit'
        });
    };

```

```

}

/* === GRAFICA DE UNA SOLA LÍNEA (Temperatura / Humedad) === */
function Plot(id, maxVal){
  this.cvs = document.getElementById(id);
  this.ctx = this.cvs.getContext('2d');
  this.max = maxVal;
  this.data = new Array(60).fill(0);
  this.colorPV = "#ffffff";

  this.add = function(v){
    this.data.push(v);
    this.data.shift();
    this.draw();
  };

  this.draw = function() {
    const w = this.cvs.width = this.cvs.clientWidth;
    const h = this.cvs.height = this.cvs.clientHeight;
    const ctx = this.ctx;

    const marginLeft = 45;
    const marginRight = 10;
    const marginTop = 30;
    const marginBottom = 20;

    ctx.clearRect(0,0,w,h);

    // ===== EJE Y + LINEAS HORIZONTALES =====
    ctx.fillStyle = "#aaa";
    ctx.font = "12px Arial";
    ctx.strokeStyle = "#222";
    ctx.lineWidth = 1;

    // definir paso según variable
    const stepY = (this.max === 60) ? 10 : 20;    // Temp=10°C, Hum=20%

    for(let v = 0; v <= this.max; v += stepY){

      let y = marginTop +
        (1 - v / this.max) *
        (h - marginTop - marginBottom);

      // texto eje Y
      ctx.fillText(v.toFixed(0), 5, y + 4);

      // línea horizontal
      ctx.beginPath();
      ctx.moveTo(marginLeft, y);
      ctx.lineTo(w - marginRight, y);
    }
  }
}

```

```

    ctx.stroke();
}

// ===== CURVA =====
const step = (w - marginLeft - marginRight) / 59;

ctx.beginPath();
ctx.strokeStyle = this.colorPV;
ctx.lineWidth = 2;

for(let i=0;i<this.data.length;i++){
    let y = marginTop +
        (1 - this.data[i]/this.max) *
        (h - marginTop - marginBottom);

    if(i===0) ctx.moveTo(marginLeft,y);
    else ctx.lineTo(marginLeft + i*step, y);
}
ctx.stroke();

// ===== LEYENDA (DESPLAZADA A LA DERECHA) =====
ctx.fillStyle = "#fff";
ctx.font = "14px Arial";
ctx.fillText(
    "PV: " + this.data[59].toFixed(2),
    marginLeft + 10,    // ← DESPLAZAMIENTO CLAVE
    marginTop - 8
);
// ===== EJE X: HORA REAL + LINEAS VERTICALES =====
ctx.font = "11px Arial";

if(time.length > 1){
    const marks = 4;

    for(let i = 0; i <= marks; i++){
        let idx = Math.round(i * (time.length - 1) / marks);
        let x = marginLeft + i * (w - marginLeft - marginRight) / marks;

        // --- LINEA VERTICAL (MISMO ESTILO QUE HORIZONTAL) ---
        ctx.strokeStyle = "#222";
        ctx.lineWidth = 1;
        ctx.beginPath();
        ctx.moveTo(x, marginTop);
        ctx.lineTo(x, h - marginBottom);
        ctx.stroke();

        // --- TEXTO DE HORA ---
        ctx.fillStyle = "#888";
        ctx.fillText(fmtTime(time[idx]), x - 22, h - 5);
    }
}

```

```

    }
  };
}

/* === GRAFICA DE DOS LÍNEAS (UV: PV + SP) === */
function MultiPlot(id, maxVal) {
  this.colorPV = "#ffffff";
  this.colorSP = "#00eaff";
  this.cvs = document.getElementById(id);
  this.ctx = this.cvs.getContext('2d');
  this.max = maxVal;
  this.data = {
    pv: new Array(60).fill(0),
    sp: new Array(60).fill(0)
  };

  this.add = function(pv, sp) {
    this.data.pv.push(pv); this.data.pv.shift();
    this.data.sp.push(sp); this.data.sp.shift();
    this.draw();
  };

  this.draw = function() {
    const w = this.cvs.width = this.cvs.clientWidth;
    const h = this.cvs.height = this.cvs.clientHeight;
    const ctx = this.ctx;

    const marginLeft = 45;
    const marginRight = 10;
    const marginTop = 25;
    const marginBottom = 20;

    const step = (w - marginLeft - marginRight) / 59;
    ctx.clearRect(0,0,w,h);

    // === EJE Y UV: marcas cada 2 unidades ===
    ctx.fillStyle = "#aaa";
    ctx.font = "12px Arial";
    ctx.strokeStyle = "#222";
    ctx.lineWidth = 1;

    const stepUV = 2; // cada 2 unidades
    for (let v = 0; v <= this.max; v += stepUV) {

      // posición Y (invertida)
      let y = marginTop + (1 - v / this.max) * (h - marginTop - marginBottom);

      // texto del valor
      ctx.fillText(v.toFixed(0), 5, y + 4);
    }
  };
}

```

```

// línea guía horizontal
ctx.beginPath();
ctx.moveTo(marginLeft, y);
ctx.lineTo(w - marginRight, y);
ctx.stroke();
}

const plotCurve = (arr, color) => {
  ctx.beginPath();
  ctx.strokeStyle = color;
  ctx.lineWidth = 2;
  for (let i = 0; i < arr.length; i++) {
    let y = marginTop + (1 - arr[i] / this.max) * (h - marginTop - marginBottom);
    if (i === 0) ctx.moveTo(marginLeft, y);
    else ctx.lineTo(marginLeft + i * step, y);
  }
  ctx.stroke();
};

plotCurve(this.data.pv, this.colorPV);
plotCurve(this.data.sp, this.colorSP);

ctx.fillStyle = "#fff";
ctx.font = "14px Arial";
ctx.fillStyle = this.colorPV;
ctx.fillText("PV: " + this.data.pv[59].toFixed(2), marginLeft + 5, 20);
ctx.fillStyle = this.colorSP;
ctx.fillText("SP: " + this.data.sp[59].toFixed(2), marginLeft + 5, 40);

// ===== EJE X: HORA REAL + LINEAS VERTICALES =====
ctx.font = "11px Arial";

if(time.length > 1){
  const marks = 4;

  for(let i = 0; i <= marks; i++){
    let idx = Math.round(i * (time.length - 1) / marks);
    let x = marginLeft + i * (w - marginLeft - marginRight) / marks;

    // --- LINEA VERTICAL (MISMO ESTILO QUE HORIZONTAL) ---
    ctx.strokeStyle = "#222";
    ctx.lineWidth = 1;
    ctx.beginPath();
    ctx.moveTo(x, marginTop);
    ctx.lineTo(x, h - marginBottom);
    ctx.stroke();

    // --- TEXTO DE HORA ---
    ctx.fillStyle = "#888";
    ctx.fillText(fmtTime(time[idx]), x - 22, h - 5);
  }
}

```

```

    }
  }
};
}

/* === INSTANCIAS === */
const pT = new Plot('cT', 60);    // Temperatura → UNA SOLA CURVA
pT.colorPV = "#ff0000";    // ROJO
const pH = new Plot('cH', 100);    // Humedad → UNA SOLA CURVA
pH.colorPV = "#0080ff";    // AZUL
const pU = new MultiPlot('cU', 10); // UV → PV + SP
pU.colorPV = "#ffca28";    // Amarillo
pU.colorSP = "#00eaff";    // Celeste

/* === LOOP DE ACTUALIZACIÓN === */
async function loop(){
  if(!histLoaded) return;    // evitar duplicación

  try {
    const r = await fetch('/datos');
    const d = await r.json();

    pT.add(d.t);
    pH.add(d.h);
    pU.add(d.u, d.sp);

    // ===== GUARDAR HORA REAL =====
    time.push(new Date());
    if(time.length > 60) time.shift();

  } catch(e){}
}

// ===== CARGAR GRAFICOS AL ABRIR LA PAGINA =====
async function cargarHist(){
  try {
    const r = await fetch('/hist');
    const h = await r.json();

    // LIMPIAR buffers antes de cargar histórico
    pT.data = [];
    pH.data = [];
    pU.data.pv = [];
    pU.data.sp = [];

    time.length = 0;    // ← LIMPIAR TIEMPOS

    for(let i = 0; i < 60; i++){
      pT.data.push(h.temp[i]);
      pH.data.push(h.hum[i]);
    }
  }
}

```

```

        pU.data.pv.push(h.uv[i]);
        pU.data.sp.push(h.sp[i]);

        // ===== HORA SIMULADA PARA HISTÓRICO =====
        time.push(new Date(Date.now() - (59 - i) * 2000));
    }

    pT.draw();
    pH.draw();
    pU.draw();

    histLoaded = true;    // histórico cargado

    } catch(e){}
}

window.onload = cargarHist;

    setInterval(loop, 2000);
</script>
</body>
</html>
)rawliteral";

//----- PID -----
float computePID(float setpoint, float measurement)
{
    unsigned long now = millis();
    float dt = (now - lastPIDMillis) / 1000.0;

    if (dt <= 0.0) return duty;
    lastPIDMillis = now;

    float error = setpoint - measurement;

    // --- INTEGRAL ---
    pidIntegral += error * Ki * dt;
    pidIntegral = constrain(pidIntegral, INTEGRAL_MIN, INTEGRAL_MAX);

    // --- DERIVADA FILTRADA ---
    float deriv = (measurement - lastMeasurement) / dt;
    float derivFiltered = derivFilterAlpha * lastDerivFiltered +
        (1.0 - derivFilterAlpha) * deriv;

    lastDerivFiltered = derivFiltered;
    lastMeasurement = measurement;

    // --- PID ---
    float output = (Kp * error) + pidIntegral - (Kd * derivFiltered);

```

```

    // Limitar a PWM
    output = constrain(output, PWM_MIN, PWM_MAX);

    return output;
}

// ----- SETUP -----
void setup() {

    Serial.begin(115200);
    dht.begin();

    Wire.begin(33, 32);           // SDA = 33, SCL = 32, iniciar I2C
    uv.begin(VEML6070_1_T);      // integración 1T (puedes usar 2T o 4T si
    quieres mayor precisión)
    Serial.println("VEML6070 iniciado correctamente");

    WiFi.mode(WIFI_AP);
    WiFi.softAP(AP_SSID, AP_PASS);
    WiFi.setSleep(false);

    Serial.print("AP IP: ");
    Serial.println(WiFi.softAPIP());

    // Inicializar históricos
    for(int i=0; i<HIST; i++){
        histTemp[i] = 0;
        histHum[i]  = 0;
        histUV[i]   = 0;
        histSP[i]   = uvSetpoint;
    }

    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", index_html);
    });

    server.on("/panel", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", panel_html);
    });

    server.on("/graficas", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", graphs_html);
    });

    // ----- SLIDER -----
    server.on("/setUV", HTTP_GET, [](AsyncWebServerRequest *request){
        if (request->hasParam("valor")) {
            float valor = request->getParam("valor")->value().toFloat();
            uvSetpoint = valor;           // valor del slider
            // Serial.print("Slider UV: ");

```

```

        // Serial.println(valor, 2);
    }
    request->send(200, "text/plain", "OK");
});
// ----- Constantes PID -----
server.on("/setPID", HTTP_GET, [](AsyncWebServerRequest *request){
    if(request->hasParam("kp")) Kp = request->getParam("kp")->value().toFloat();
    if(request->hasParam("ki")) Ki = request->getParam("ki")->value().toFloat();
    if(request->hasParam("kd")) Kd = request->getParam("kd")->value().toFloat();

    //Serial.printf("PID Actualizado → Kp=%.2f Ki=%.2f Kd=%.2f\n", Kp, Ki, Kd);
    request->send(200, "text/plain", "OK");
});
// ----- JSON PID -----
server.on("/getPID", HTTP_GET, [](AsyncWebServerRequest *request){
    String json = "{}";
    json += "\"kp\": " + String(Kp, 2) + ",";
    json += "\"ki\": " + String(Ki, 2) + ",";
    json += "\"kd\": " + String(Kd, 2);
    json += "}";
    request->send(200, "application/json", json);
});

// ----- JSON GRAPHICS -----
server.on("/datos", HTTP_GET, [](AsyncWebServerRequest *request){
    String json = "{}";
    json += "\"t\": " + String(cachedTemp, 1) + ",";
    json += "\"h\": " + String(cachedHum, 1) + ",";
    json += "\"u\": " + String(cachedUV, 2) + ","; // enviar índice UV real con 2
decimales
    json += "\"sp\": " + String(uvSetpoint, 2); // enviar setPoint UV con 2 decimales
    json += "}";
    request->send(200, "application/json", json);
});

// Endpoint para devolver el setpoint actual (usado por la página panel)
server.on("/getSetpoint", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/plain", String(uvSetpoint));
});

// Endpoint para históricos
server.on("/hist", HTTP_GET, [](AsyncWebServerRequest *request){
    String json = "{}";
    json += "\"temp\":[";
    for(int i = 0; i < HIST; i++){
        int idx = (histIndex + i) % HIST;
        json += String(histTemp[idx], 1);
        if(i < HIST-1) json += ",";
    }
}

```

```

    json += ",";
    json += "\"hum\":[";
    for(int i = 0; i < HIST; i++){
        int idx = (histIndex + i) % HIST;
        json += String(histHum[idx], 1);
        if(i < HIST-1) json += ",";
    }
    json += ",";

    json += "\"uv\":[";
    for(int i = 0; i < HIST; i++){
        int idx = (histIndex + i) % HIST;
        json += String(histUV[idx], 2);
        if(i < HIST-1) json += ",";
    }
    json += ",";

    json += "\"sp\":[";
    for(int i = 0; i < HIST; i++){
        int idx = (histIndex + i) % HIST;
        json += String(histSP[idx], 2);
        if(i < HIST-1) json += ",";
    }
    json += "]";

    json += "}";
    request->send(200, "application/json", json);
});

server.begin();
Serial.println("Servidor Asincrono Iniciado");

pinMode(pwmPin, OUTPUT);
// ledcSetup(pwmChannel, pwmFreq, pwmResolution);
// ledcAttachPin(pwmPin, pwmChannel);

pinMode(blinkPin, OUTPUT);
}

// ----- LOOP -----
void loop() {

    unsigned long now = millis();

    // --- Lectura del sensor VEML6070 ---
    uint16_t uvRaw = uv.readUV();    // lectura del sensor
    cachedUV = uvRaw * 0.05;        // escala típica → índice UV aproximado, 0.005

    if (now - lastSamplePID >= SAMPLE_INTERVAL_PID) {
        lastSamplePID = now;
    }

```

```

// ---- PID ----
duty = computePID(uvSetpoint, cachedUV);
analogWrite(pwmPin, duty);

Serial.print("SP: "); Serial.print(uvSetpoint, 2);
Serial.print(" | PV: "); Serial.print(cachedUV, 2);
Serial.print(" | PWM: "); Serial.println(duty);
}

if (now - lastSample >= SAMPLE_INTERVAL) {
    lastSample = now;

    float t = dht.readTemperature();
    float h = dht.readHumidity();

    if (!isnan(t)) cachedTemp = t;
    if (!isnan(h)) cachedHum = h;

    // Guardar nueva muestra cada 2 segundos
    histTemp[histIndex] = cachedTemp;
    histHum[histIndex] = cachedHum;
    histUV[histIndex] = cachedUV;
    histSP[histIndex] = uvSetpoint;
    histIndex = (histIndex + 1) % HIST;
}

if (cont_blink < 50)
{
    cont_blink = cont_blink + 1;
    if (cont_blink < 25)
        digitalWrite(blinkPin, HIGH);
    else
        digitalWrite(blinkPin, LOW);
}
else
{
    cont_blink = 0;
}
delay(5);
}

```

12.2 Evidencia fotográfica del prototipo

Figura 29.

Fabricación de estructura metálica

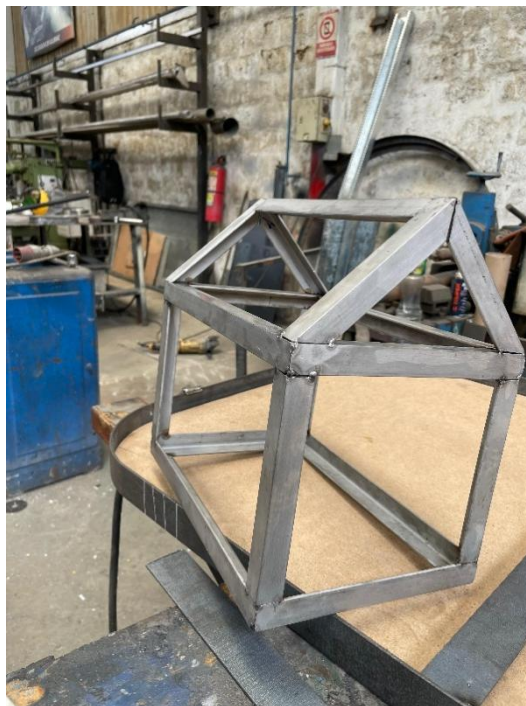


Figura 30.
Diseño de base para estructura

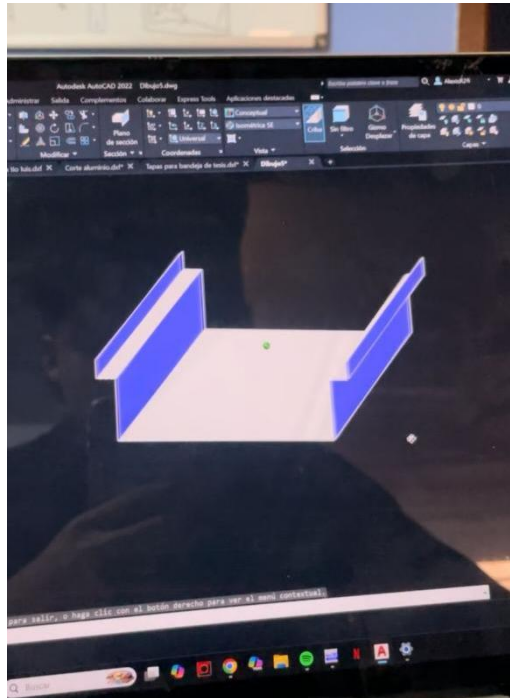


Figura 31.
Montaje de paredes de acrílico



Figura 32.
Programación del ESP32

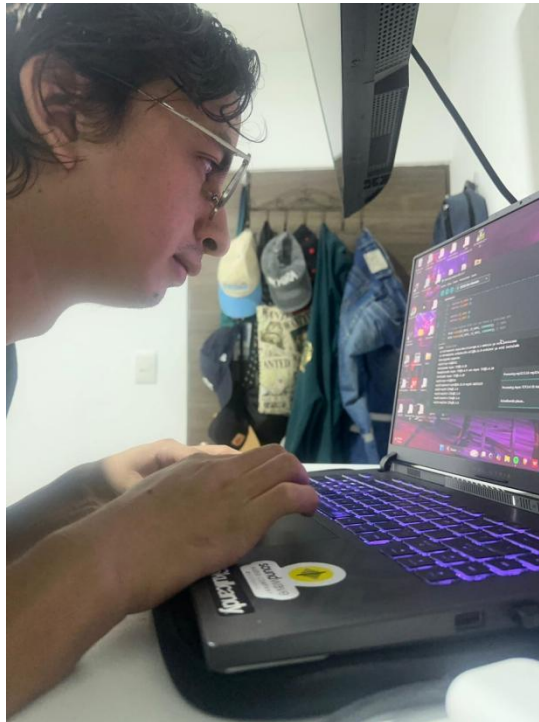


Figura 33.
Conexión de elementos electrónicos

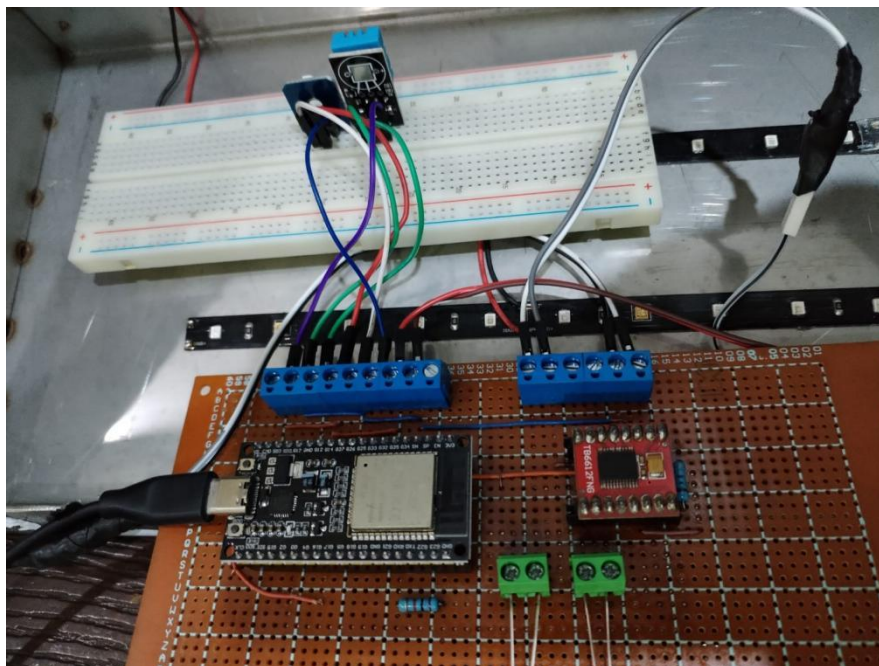


Figura 34.
Pruebas de funcionamiento

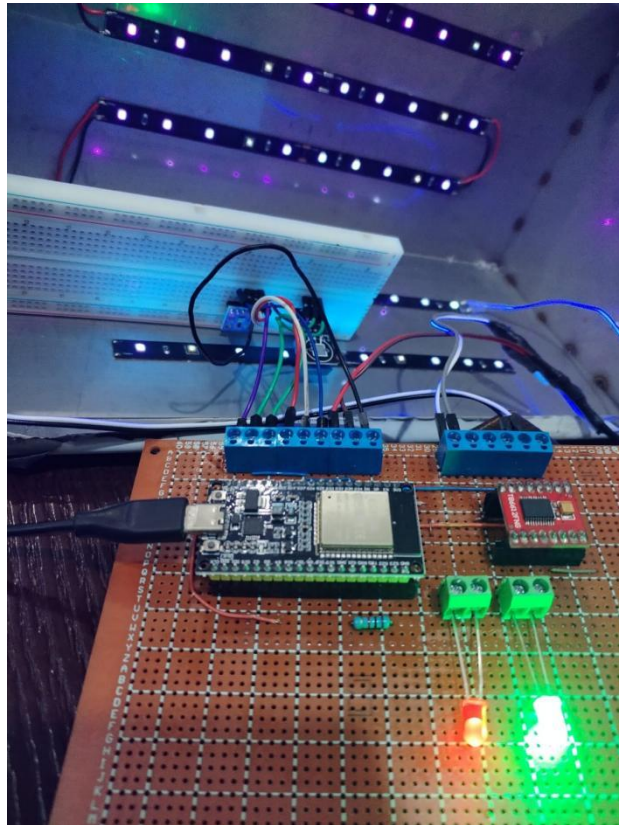


Figura 35.
Montaje de circuito en maqueta

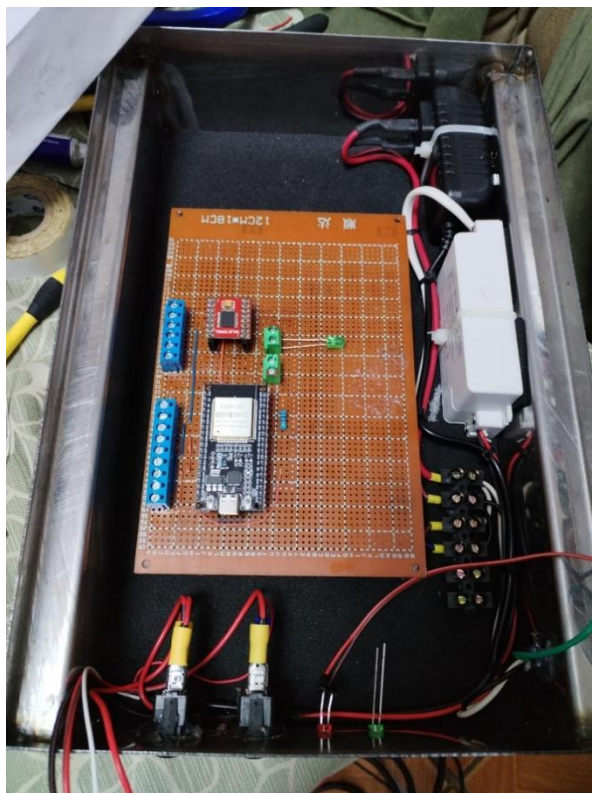


Figura 36.
Pruebas de funcionamiento



Figura 37.
Verificación de funcionalidad

