Universidad Nacional de Ingeniería

Área de Conocimiento de Tecnología de la Información y Comunicación

Diseño de un Prototipo de Incubadora Avícola Automatizada Basada en IoT para la Optimización Productiva en la Comunidad Mario Corea, Chinandega

Trabajo Monográfico para optar al título de Ingeniero Electrónico

Elaborado por:

Br. Asling Massiel Ney Ruiz
Carnet:2019-1126U

Br. Luis Fernando Alvarado Abarca
Carnet:2019-1084U

Br. Justin Ernesto Vallejos Loaisiga
Carnet:2019-0110U

Tutor:

Ing. Marlovio José Sevilla Hernández

22 de abril del 2026,
Managua, Nicaragua

Área de Conocimiento de
Tecnología de la Información
y Comunicación

Managua 21 de abril de 2026

MSc. Claudia Benavides Rugama
Directora DACTIC
Su Despacho

Estimada MSc. Benavides.

Me dirijo a usted de la manera más atenta para hacer de su conocimiento, que el trabajo monográfico aprobado para el egresado del programa académico de Ingeniería Electrónica Br. Luis Fernando Alvarado Abarca (Carnet: 2019-1084U), Asling Massiel Ney Ruiz (2019-1126U) y Justin Ernesto Vallejos Loaisiga (2019-0110U) los cuales tiene como título **Diseño de un prototipo de Incubadora Avícola Automatizada Basada en IoT para la optimización productiva en la comunidad de Mario Corea Chinandega**, del cual soy el tutor, luego de revisar la documentación correspondiente, considero que este ha llegado a su feliz término y se encuentra listo para realizar el proceso de defensa.

Agradeciendo su amable atención, quedo a la espera de sus instrucciones aprovechando la ocasión para reiterarle mis más sinceras muestras de consideración y respeto.

NOTA: Se anexan tres ejemplares del trabajo monográfico, para la defensa monográfica.

Atentamente,

Ing. Marlovio José Sevilla Hernández
Profesor Auxiliar
Tutor

Cc. Archivo

Móvil: (505) 8588 8333

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

www.uni.edu.ni

Dirección de Área de
Conocimiento de Tecnología
de la Información y Comunicación

Managua, 26 de enero 2026

Br. Asling Massiel Ney Ruíz ***2019-1126U***
Br. Luis Fernando Alvarado Abarca ***2019-1084U***
Br. Justin Ernesto Vallejos Loáisiga ***2019-0110U***
Egresados Programa académico Ingeniería Electrónica
Sus manos.-

Estimados Egresados:

Reciban cordiales saludos y éxito en sus actividades.

*Por medio de la presente, les comunico la aprobación del trabajo monográfico titulado: "**Diseño de incubadora avícola automatizada basada en IoT para la optimización productiva en la Comunidad Mario Corea, Chinandega"**, el cual cumple con los requisitos establecidos en el capítulo II de la normativa de trabajos monográficos de la UNI como forma de culminación de estudios, por lo que queda oficialmente aprobado por esta Dirección.*

El docente responsable de acompañarle en el proceso de desarrollo de su tema monográfico es el Ing. Marlovio José Sevilla Hernández.

A partir de la fecha de aprobación, tienen como máximo seis meses para realizar el trabajo monográfico incluyendo la defensa.

Atentamente,

MSc. Claudia Lucía Benavídez Rugama
Directora Área de Conocimiento de
Tecnología de la Información y Comunicación

Cc. Ing. Marlovio José Sevilla Hernández – Tutor
MSc. Luisa Massiel Mercado Gutiérrez - Secretaria Académica.
Archivo DACTIC.

Móvil: (505) 8889 0903

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

claudia.benavidez@dactic.uni.edu.ni-
www.uni.edu.ni

Agradecimiento

A Dios, por darnos la sabiduría y fortaleza para culminar este proceso. A la memoria de quienes ya no están, pero cuya luz sigue guiando nuestro camino. Este logro es fruto del esfuerzo compartido entre Asling Massiel Ney Ruiz, Luis Fernando Alvarado Abarca y Justin Ernesto Vallejos Loaisiga, con el firme propósito de poner la tecnología al servicio del desarrollo humano.

A nuestros padres y familias, quienes con su amor, sacrificio y apoyo incondicional han sido el pilar fundamental para alcanzar esta meta profesional. A nuestro tutor, el Ing. Marlovio José Sevilla Hernández, por su guía experta y por compartir sus conocimientos en el área de la electrónica y el control.

A todas las familias productoras nicaragüenses que, a través de la innovación y el trabajo en el campo, luchan diariamente por la seguridad alimentaria y el desarrollo de nuestra nación.

Resumen

Esta investigación presenta un diseño de incubadora avícola automatizada que integra tecnologías de Internet de las Cosas (IoT) y control PID para optimizar la producción en la comunidad rural Mario Corea, Chinandega. El diseño busca mejorar la tasa de eclosión frente a los sistemas tradicionales y se alinea estratégicamente con el Plan Nacional de Lucha contra la Pobreza (2022-2026) de Nicaragua.

El sistema automatiza variables críticas mediante un microcontrolador ESP32 y sensores DHT22, fijando la temperatura en 37.8 °C y regulando la humedad relativa inicialmente entre el 50% y 60%. Además, implementa una lógica de volteo automático con servomotor para el desarrollo embrionario y ventilación controlada por PWM para el intercambio gaseoso, enviando telemetría en tiempo real a través del protocolo MQTT hacia un dashboard remoto.

La viabilidad del diseño fue validada mediante simulaciones en MATLAB/Simulink, donde el modelo térmico alcanzó estabilidad óptima en aproximadamente 1,000 segundos, y en el entorno Wokwi, que certificó la ejecución del firmware en C++ y la gestión de alertas. Desde la perspectiva económica, el prototipo requiere una inversión de C$ 7,247.90 y, con una tasa de eclosión proyectada del 87%, permite recuperar el capital en 3.5 meses, consolidándose como una solución accesible para pequeños productores.

Palabras clave: Incubación avícola, Internet de las Cosas (IoT), Control PID, ESP32, Seguridad alimentaria, Automatización rural.

Glosario

- PID (Proporcional-Integral-Derivativo): Es el controlador de lazo cerrado utilizado para regular variables continuas como la temperatura y la humedad, permitiendo alcanzar la estabilidad mediante tres acciones combinadas

.

- PWM (Modulación por Ancho de Pulso): Técnica utilizada para controlar la potencia de los actuadores, como la velocidad del ventilador o la intensidad de la resistencia, permitiendo un ajuste gradual en lugar de un simple encendido/apagado

.

- FOPDT (First Order Plus Dead Time): Modelo matemático de primer orden con tiempo muerto que describe el comportamiento térmico de la incubadora para su simulación en entornos como MATLAB

.

- GPIO (General Purpose Input/Output): Pines de entrada y salida de propósito general del microcontrolador utilizados para conectar sensores y actuadores

.

- I2C (Inter-Integrated Circuit): Protocolo de comunicación serial utilizado específicamente para conectar periféricos como la pantalla OLED al microcontrolador

.

- IoT (Internet de las Cosas): Red de objetos físicos conectados que recolectan y transmiten datos para automatizar procesos y permitir el monitoreo remoto

.

- MQTT (Message Queuing Telemetry Transport): Protocolo de mensajería ligero diseñado para comunicaciones con bajo ancho de banda y enlaces inestables, fundamental para el envío de telemetría en este proyecto

.

- QoS (Quality of Service): Niveles de calidad en la entrega de mensajes dentro del protocolo MQTT para garantizar que los datos lleguen al receptor

.

- LPWAN (Low Power Wide Area Network): Redes de bajo consumo y largo alcance diseñadas para sistemas IoT en zonas rurales, donde se incluyen protocolos como LoRaWAN y Sigfox

.

- NB-IoT (Narrowband IoT): Variante de tecnología IoT basada en redes celulares (LTE) para dispositivos de bajo consumo

.

- ESP32: Microcontrolador de alto rendimiento con Wi-Fi y Bluetooth integrados que funciona como la unidad central de procesamiento del sistema

.

- DHT22: Sensor digital de precisión utilizado para medir simultáneamente la temperatura y la humedad relativa

.

- OLED (Organic Light-Emitting Diode): Tipo de pantalla utilizada para la interfaz de usuario local donde se visualizan los datos en tiempo real

.

- SMPS (Switch Mode Power Supply): Fuente de alimentación conmutada de alta eficiencia utilizada para regular la energía del sistema minimizando el calor residual

.

- EMC (Compatibilidad Electromagnética): Se refiere a la capacidad de los equipos para funcionar correctamente en su entorno electromagnético sin introducir perturbaciones

.

- PNCL-DH: Plan Nacional de Lucha contra la Pobreza y para el Desarrollo Humano (2022-2026) de Nicaragua, marco estratégico en el que se inscribe este proyecto

.

- ROI (Return on Investment): Retorno de inversión; cálculo financiero que determina la rentabilidad del proyecto y el tiempo necesario para recuperar el capital invertido

.

Índice de Contenido

1. Introducción 1

2. Objetivos 2

2.1. Objetivo General 2

2.2. Objetivos Específicos 2

3. Marco Teorico 3

3.1. Conceptos Fundamentales de la Incubación Avícola 3

3.1.1. Definición de incubación natural y artificial. 3

3.1.1.1. Incubación Artificial. 3

3.1.2. Importancia de la incubación para la producción avícola. 3

3.1.3. Fases del proceso de incubación 4

3.1.3.1. Fase de Precalentamiento 4

3.1.3.2. Fase de Desarrollo Embrionario 4

3.1.4. Lineamientos del Plan Nacional de Lucha contra la Pobreza y el Hambre (2022–2026) 5

3.1.4.1. Lineamiento 3: Desarrollo de Talentos Humanos e Innovación 5

3.1.4.2. Lineamiento 4: Más y Mejor Producción en el Campo 5

3.2. Internet de las Cosas (IoT) 6

3.2.1. Concepto general de IoT 6

3.2.1.1. Sensores y Actuadores 6

3.2.1.2. Actuadores (Transductores de Salida) 7

3.2.1.3. Microcontroladores 7

3.2.1.4. Fuentes de alimentación 8

3.2.1.4.1. Fuente de alimentación conmutada 8

3.2.1.5. Conectividad (La Capa de Red) 9

3.2.1.6. Plataformas de Datos y Servicios 9

3.2.1.7. Servicios 9

3.3. Aplicaciones de IoT en agricultura y avicultura. 9

3.3.1. Aplicaciones Específicas en la Avicultura 10

3.3.2. Ventajas: monitoreo en tiempo real, predicción de fallas, análisis de datos. 10

3.3.2.1. Monitoreo en Tiempo Real 10

3.3.2.2. Predicción de Fallas 11

3.3.2.3. Análisis de Datos 11

3.4. Caracterización de la Comunidad Mario Corea (Palo Verde) 11

3.4.1. Ubicación de la Comunidad que se contempla 12

3.4.2. Vulnerabilidad y Necesidad de Innovación Tecnológica 12

3.5. Componente de Telecomunicaciones 13

3.5.1. Protocolos de Comunicación para IoT en Zonas Rurales 13

3.5.1.1. LoRaWAN 13

3.5.1.2. NB-IoT 13

3.5.1.3. Sigfox 13

3.5.1.4. MQTT-SN 13

3.5.1.5. MQTT (Message Queuing Telemetry Transport): 13

3.5.1.6. Niveles de calidad de servicio (QoS) 14

3.5.2. Arquitectura Cliente-Servidor vs. Publish-Subscribe 14
3.5.2.1. El Modelo Cliente-Servidor 14
3.5.2.2. El Modelo Publish-Subscribe 15

3.6. Variables Críticas en la Incubación Artificial 15
3.6.1. Temperatura: rangos óptimos, influencia en el desarrollo embrionario. 16
3.6.2. Humedad relativa: efectos sobre la pérdida de agua del huevo. 16
3.6.3. Ventilación y oxigenación: importancia para el intercambio gaseoso. 17
3.6.4. Giro o volteo: frecuencia y consecuencias de una rotación inadecuada. 17
3.6.5. Beneficios de la automatización: estabilidad ambiental, reducción de errores humanos, mejora de la tasa de eclosión. 17

3.7. Sistemas de Monitoreo y Plataformas de Análisis de Datos 18
3.7.1. Tipos de plataformas para visualizar datos (dashboards, aplicaciones). 18
4.5.1.1. Generación de alertas 19
3.7.2. Generación de alertas y predicción de fallas. 19

3.8. Modelos Conceptuales para el Diseño de un Sistema Automatizado 19
3.8.1. Estructura básica de un modelo conceptual: entradas, procesos, salidas. 19
3.8.2. Representación de la interacción entre sensores, actuadores y software. 20
3.8.3. Modelado de interfaces y señales 20
3.8.4. Mecanismos de control (Control PID, automatización por eventos). 20

3.9. Contexto del Sector Avícola en Nicaragua 21
3.9.1. Importancia del pollo y el huevo en la dieta nacional. 21
3.9.2. Problemas del sector: dependencia de importaciones, costos, falta de equipamiento. 21
3.9.3. Necesidad de alternativas tecnológicas locales 22

3.10. Impactos Esperados de Tecnologías IoT en la Producción Avícola 22
3.10.1. Impacto económico: reducción de costos, mayor eficiencia. 22
3.10.2. Impacto social: empleo, capacitación, empoderamiento comunitario. 22
3.10.3. Impacto ambiental: eficiencia energética, reducción de pérdidas. 23

4. Metodología 23

4.1. Enfoque metodológico 23

4.2. Alcance de la investigación 24

4.3. Diseño de la investigación 24

4.4. Contexto del estudio 24

4.5. Procedimiento metodológico 24
4.5.1. Fase 1: Revisión documental y análisis técnico 25
4.5.2. Fase 2: Definición de variables de estudio 25
4.5.3. Fase 3. Diseño de la arquitectura del sistema 25
4.5.4. Fase 4. Simulación del sistema 25
4.5.5. Fase 5. Análisis de resultados 26

4.6. Herramientas y dispositivos utilizados en la simulación 26

5. Desarrollo 27

5.1. Cálculo para el valor de la resistencia calefactora ambiental 27

5.2. Energía requerida dentro del sistema de incubación 27
5.3. Energía requerida por los huevos de gallina 29
5.4. Energía perdida a través de las paredes 30
5.5. Determinación de Sensor 32
5.6. Determinación de sistemas de acoplamiento y drivers de potencia 33
5.6.1. Módulo de Conmutación por Aislamiento Óptico(Relé Optoacoplador). 33
5.6.2. Driver de Potencia de Estado Sólido (Puente H L298N) 34
5.7. Robustez Del Diseño 35
5.8. Análisis De Carga Y Cálculos De Potencia 35
5.9. Subtotales por Voltaje 37
5.9.1. Justificación de la Fuente 38
5.10. Determinación Del Sistema De Alimentación Y Gestión Energética 38
5.10.1. Justificación de la Tecnología Conmutada 38
5.10.2. Arquitectura de Distribución de Voltaje (Regulación en Cascada) 39
5.10.3. Capacidad de Corriente y Factor de Reserva 39
5.11. Determinación de Microcontrolador 40
5.12. Regulador de voltaje 41
5.12.1. Funcionamiento de una fuente de alimentación conmutada 42
5.12.2. Filtro de compatibilidad electromagnética (EMC) 42
5.13. Diagrama de Bloques 43
5.14. Gestión de Energía y Respaldo Crítico (PWR) 44
5.15. Adquisición y Procesamiento (MCU & SNS) 44
5.16. Etapa de Actuación de Potencia (DRV & RLY) 44
5.17. Telemetría y Alertas (IOT & ALM) 45
5.18. Simulacion del sistema implementado en Matlab, Wokwi y AdaFruit IO 45
5.18.1. Modelado matemático del sistema térmico y humedad de la incubadora 45
5.18.1.1. Planteamiento general del sistema 45
5.18.1.2. Modelo matemático para el sistema Térmico 46
5.18.1.3. Modelo matemático para el sistema de humedad 48
5.18.1.4. Sintonización por Ziegler-Nichols en lazo cerrado 49
5.18.1.4.1. Calculo analítico del lazo térmico 50
5.18.1.4.2. Cálculo analítico del lazo de humedad 50
5.18.1.4.3. Ajuste fino conservador para incubación biológica 51
5.18.1.5. Implementación en Matlab de Sistema Térmico y Sistema de Humedad 51
5.18.1.6. Implementación y estructura en Matlab 52
5.18.1.7. Consideraciones del Subsistema de Humedad 53
5.18.1.8. Interpretación de la Respuesta Dinámica del Control Térmico 53
5.18.1.9. Modelado de la Planta y Configuración del Lazo 56
5.18.1.10. Compensación de Temperatura Ambiente (Bloque sum+Constant 32) 57
5.18.1.11. Lazo de Realimentación (Feedback) 57
5.18.1.12. Análisis de Resultados y Estabilidad del Sistema 58

5.18.1.13. Validación en Simulaciones en Matlab 59
5.18.2. Validación en Entorno de Simulación Wokwi 59
5.18.3. Implementación del Algoritmo PID en Código C++ 61
5.18.4. Lógica de volteo automático y la progresión temporal 62
5.18.5. Lógica de Simulación Temporal y Volteo de Huevos 62
5.18.6. Gestión del Tiempo y Progresión de Días 62
5.18.7. Mecanismo Volteo Automático y Nacimiento. 63
5.18.8. Lógica de Control de Actuadores y Simulación de Humedad 63
5.18.9. Validación de Conectividad IoT mediante MQTT en Simulación Wokwi 64
5.18.10. Variables Críticas publicadas 66
5.18.11. Interfaz de Usuario Local 67
5.18.12. Simulación a través del entorno de Wokwi 68
5.18.13. Validación de Transmisión de datos en Adafruit Io 69
5.18.13.1. Captura 1: Dashboard Adafruit IO - Día 1 (Validación Inicial) 69
5.18.13.1.1. Interpretación de resultados mostrados 70
5.18.13.2. Captura 2: Dashboard Adafruit IO - Día 3 (Validación de Persistencia) 71
5.18.13.2.1. Interpretación de Resultados 72
5.19. Identificación de la Falla en la Válvula (Relé 3) 72
5.20. Reacción del Circuito ante el Desvío 73
5.21. Respuesta Sonora y Visual 74
5.22. Resultados de la simulación: 74
5.23. Presupuesto 74
5.23.1. Componentes Electrónicos 75
5.23.2. Materiales Estructurales 77
5.23.3. Herramientas y Equipos 78
5.23.4. Costos Adicionales 79
5.23.5. Resumen general de gastos 80
5.23.6. Análisis Comparativo 81
5.23.7. Análisis de Viabilidad Ecónomica 82
5.23.8. Recomendaciones de compra y proveedores 83
5.23.9. Sostenibilidad del proyecto 83
6. Conclusiones 84
7. Recomendaciones 86
8. Bibliografias 87
9. Anexos 89

Índice de Figuras

Figura 1 Fuente de Alimentación Lineal y Fuente de Aimentación Conmutada. Fuente: Obtenida del Articulo “Diferencias entre una Fuente de alimentación lineal y conmutada” 8
Figura 2 La imagen muestra el área de intervención del proyecto, caracterizada por una densidad de 30 hogares dedicados a la crianza de aves y cultivos básicos bajo condiciones climáticas de alta temperatura ambiental. Fuente: Google Maps 12
Figura 3. Arquitectura de Sensor DHT22. Fuente: Articulo de Revista Española de Electrónica 32
Figura 4. Optoacoplador PC817. Fuente: Nikon Shutter release - Code mod help needed. 34
Figura 5. L298N Modulo Driver Motor A pasos. Fuente UNITS ELECTRONICS Tienda de Componentes Electronicos 34
Figura 6. Diagrama de PinOut del ESP32. Fuente: ESP32 Pinout Diagram Overview Scribd 41
Figura 7. Regulador Step-Down XL4015. Fuente: Kit De 2 Módulos Reductores Dc-dc Xl4015 Ajustables 5a 75w Con Voltímetro Led, Incluye Pilares, Alta Eficiencia 96%, Voltaje Salida 1.25-36v, Protección Contra Cortocircuito Y Sobrecalentamiento MERCADOLIBRE.....42
Figura 8. Filtro de compatibilidad electromagnética. Fuente: Cómo funcionan las fuentes de alimentación conmutadas I en FIDESTEC. 42
Figura 9. Incubadora IOT carry Circuito electrónico detallado. Fuente: Sistema disenado como imagen referencial. 43
Figura 10. Incubadora Carry diagrama de bloques. Fuente: Imagen Propia creada en Powerpoint 43
Figura 11. Fragmento del script en MATLAB utilizado para el modelado térmico de la incubadora y la obtención de los parámetros analíticos de sintonización PID por Ziegler–Nichols. Fuente: Elaboración propia y simulaciones desarrollados en Matlab 52
Figura 12. Comportamiento del subsistema térmico mediante control PID conservador, se presenta la respuesta del sistema, análisis ampliado del régimen transitorio y la capacidad de rechazo ante una perturbación térmica externa. Fuente: Modelado efectuados en MATLAB 53
Figura 13. Resultados de las métricas de desempeño del lazo térmico en MATLAB. Fuente: Propia Obtenida durante simulacion 54
Figura 14. Simulacion de la Planta. Fuente: Propia obtenida en el entorno de simulacion MATLAB Simulink 56
Figura 15. Retardo de Transporte (Transport Delay): Ajustado a 5 segundos para simular la latencia real del sensor DHT22. Fuente: Delay de la onda transportadora. Fuente: Propia obtenida en el entornon de simulaicion de MATLAB Simulink.....56

Figura 16. Parámetros de la función de transferencia. Fuente: Propia obtenida en el entorno de simulacion de MATLAB Simulink 57
Figura 17. Simulacion de las Curvas. Fuente: Propia obtenida en el entorno de simulacion MATLAB Simulink 58
Figura 18. Validación de la propuesta de diseño mediante prototipado virtual. Fuente: Propia obtenida en el entorno de simulacion de WOKWI 60
Figura 19. Código fuente con la implementación de las constantes PID para el control térmico. Fuente: Propia 61
Figura 20. Configuración de pines, canales PWM para el ventilador e inicialización de red WiFi. Fuente: Propia en el entorno de simulacion WOKWI 61
Figura 21. Simulación Wokwi basada en el análisis del código fuente y los resultados obtenidos. Fuente: Propia en la simulación de Wokwi. 62
Figura 22. Relé como simulador del driver. Fuente: Propia 63
Figura 23. Monitor Serial del ESP32 en Wokwi mostrando información de conectividad WiFi. Fuente: Propia 65
Figura 24. Estructura de tópicos Hive MQTT para la telemetría de temperatura, humedad y estado de actuadores. Fuente: Propia obtenida en el entorno de simulacion Wokwi 65
Figura 25. MQTT operativo: Los tópicos están siendo publicados cada 10 segundos según la lógica. Fuente propia: Obtenido de entorno de simulación WOKWI 66
Figura 27. Variables Criticas. Fuente: Obtenida del entorno de simulación WOKWI .66
Figura 27. Variables Criticas. Fuente: Obtenida del entorno de simulación WOKWI .66
Figura 28. Muestreo de código de pantalla. Fuente: Propia obtenida del entorno de simulacion WOKWI 67
Figura 29. Interfaz local SSD1306 mostrando el estado de incubación en tiempo real. Fuente: Propia obtenida del entorno de simulacion WOKWI 67
Figura 30. Simulación en entorno Wokwi y Recepción de datos en Broquer MQTT. Fuente: Propia 68
Figura 31. Dashboard Adafruit IO - Día 1 de incubación. Fuente: Propia obtenido del entorno de AdaFruit 69
Figura 32. Dashboard Adafruit IO - Día 3 de incubación. Fuente: Propia obtenido del entorno de AdaFruit 71
Figura 33. Identificacion de falla en la valvula (Rele 3). Fuente: Propia desde el entorno de simulacion de Wokwi 72

Índice de Tablas

Tabla 1 Análisis Comparativo de Sensores Digitales 7
Tabla 2 Protocolos de Comunicación para IoT en Zonas Rurales 14
Tabla 3 Modelo Publish-Subscribe 15
Tabla 4 Temperatura: rangos óptimos, influencia en el desarrollo embrionario 16
Tabla 5. Cálculos De Potencia 36
Tabla 6 Tabla clásica de Ziegler-Nichols 49
Tabla 7. Componentes Electronicos 75
Tabla 8. Materiales Estructurales 77
Tabla 9. Herramientas y Equipos 78
Tabla 10. Costos Adicionales 79
Tabla 11. Resumen General de Gastos 80
Tabla 12. Análisis Comparativo 81
Tabla 13. Análisis de Viabilidad Ecónomica 82

Índice de Ecuaciones

Ecuaciones 1 Formula para calcular el factor calorífico 27
Ecuaciones 2 Ecuación de la densidad absoluta 28
Ecuaciones 3 Ecuación de la densidad absoluta 28
Ecuaciones 4 Formula de Energía 30
Ecuaciones 5 Potencia Requerida para un sistema 30
Ecuaciones 6. Energía térmica que se pierde a través de una pared 31
Ecuaciones 7. Requerimiento energético optimo es la suma de las potencias calculadas 32
Ecuaciones 8. Ley de Watt 36
Ecuaciones 9. Factor de Reserva 40
Ecuaciones 10. ecuación diferencial que rige la dinámica de la temperatura 46
Ecuaciones 11. Ganancia equivalente en el lazo 47
Ecuaciones 12. Constante de tiempo para la humedad 48
Ecuaciones 13. Función de Transferencia 50

1. Introducción

El diseño del presente prototipo de incubadora avícola con IoT se fundamenta en la estrategia nacional de lucha contra la pobreza y las desigualdades, establecida en el Plan Nacional de Lucha contra la Pobreza y para el Desarrollo Humano 2022-2026 (PNCL-DH 2022-2026). El Plan prioriza la acción de "Más y mejor producción en el campo", buscando fortalecer la cadena avícola familiar para mejorar la rentabilidad de estos emprendimientos y, consecuentemente, elevar la calidad de vida de la población (Gobierno de Reconciliación y Unidad Nacional, 2021, p. 109).

En el contexto local, la comunidad Mario Corea en el Ingenio Monte Rosa, Chinandega, tipifica los desafíos de la economía familiar. Con alrededor de 30 hogares enfrentando el desempleo, la subsistencia depende de la producción agropecuaria, que combina la siembra de alimentos con la cría de aves de corral. La eficiencia de la producción avícola es un factor clave para el sustento económico, pero se ve limitada por el uso de sistemas de incubación tradicionales. Estos sistemas carecen de la capacidad de mantener y monitorear con precisión parámetros ambientales críticos, lo que se traduce en bajas tasas de eclosión e ineficiencia productiva. Esta situación no solo acentúa la vulnerabilidad económica, sino que también dificulta una gestión de recursos responsable, con impacto en el medio ambiente.

Esta iniciativa se fundamenta en las estrategias nacionales de generación y transferencia de tecnologías aplicadas a la producción (Gobierno de Reconciliación y Unidad Nacional, 2021), así como en el impulso de soluciones digitales que incrementen los rendimientos productivos. El prototipo permitirá un control ambiental eficiente, promoviendo el uso sostenible de los recursos y contribuyendo a mejorar la calidad de vida de las familias productoras de Chinandega.

2. Objetivos

2.1. Objetivo General

- Diseñar un prototipo de incubadora avícola automatizada mediante técnicas de Internet de las Cosas (IoT) y control PID, para la optimización de la tasa de eclosión y el monitoreo remoto de variables críticas en la comunidad Mario Corea.

2.2. Objetivos Específicos

- Diagnosticar las condiciones técnicas y ambientales de la comunidad Mario Corea, identificando los requerimientos de conectividad y variables críticas (temperatura, humedad, volteo) para la incubación artificial.

- Modelar la arquitectura de hardware y software del sistema, seleccionando sensores de precisión, actuadores y protocolos de comunicación (MQTT/Wi-Fi) adecuados para entornos rurales.

- Implementar el algoritmo de control de temperatura y humedad en el microcontrolador, integrando la visualización de datos en una plataforma IoT (Dashboard).

3. Marco Teorico

3.1. Conceptos Fundamentales de la Incubación Avícola

3.1.1. Definición de incubación natural y artificial.

La incubación natural es el método biológico original por el cual el ave clueca genera el calor necesario para el desarrollo continuo del embrión. Este proceso no depende de la tecnología, sino de una adaptación física, la zona de cría. No obstante, aunque es un proceso orgánico, este método presenta limitaciones considerables en términos de escala productiva y consistencia en el control de grandes volúmenes de huevos.

3.1.1.1. Incubación Artificial.

Es la metodología tecnológica que replica las condiciones naturales de cría mediante dispositivos electrónicos, buscando la máxima tasa de eclosión a gran escala, algo que es poco práctico biológicamente. Estos equipos demandan un control preciso y continuo de los parámetros críticos. Al respecto, Agriculture Institute (2023) señala: "Los sensores de temperatura mantienen el calor preciso necesario para el desarrollo embrionario, mientras que los controles de humedad garantizan niveles adecuados de humedad durante todo el período de incubación.". El diseño de incubadora utilizando tecnología IoT se centra en asegurar este control constante, clave para superar las limitaciones de los métodos tradicionales.

3.1.2. Importancia de la incubación para la producción avícola.

La incubación avícola tiene un gran impacto económico al ser el proceso que determina la productividad. Garantizar un ambiente preciso para el desarrollo embrionario no solo asegura una alta tasa de eclosión, sino que también establece la sostenibilidad de la producción en términos de calidad y cantidad, lo cual es vital para la rentabilidad a largo plazo.

3.1.3. Fases del proceso de incubación

El ciclo de incubación artificial, que dura aproximadamente 21 días para los huevos de gallina, se divide en varias fases. Cada una de ellas exige que el sistema de control mantenga parámetros ambientales específicos. Es la precisión con la que se manejan estas etapas es lo que realmente determina la viabilidad de los embriones.

3.1.3.1. Fase de Precalentamiento

Esta primera fase es la preparación clave de la incubadora. Lo que se busca es que el ambiente interno esté totalmente estable antes de meter los huevos, evitando así que sufran un choque térmico a través de variaciones bruscas de temperatura y que el vapor de agua se condense en la cascara, ocasionando poros taponeados justo al inicio de su desarrollo.

- Duración: Se recomienda un periodo de estabilización de entre 12 y 24 horas.
- Parámetros: La temperatura debe fijarse y sostenerse, usualmente entre 37.5 ◦ C y 37.8 ◦ C, y después se enfrían a temperatura ambiente antes de colocarlos en las incubadoras.
- La humedad inicial debe rondar el 50% al 60% para que empiece la evaporación de manera gradual.

3.1.3.2. Fase de Desarrollo Embrionario

La fase de desarrollo embrionario es la etapa más extensa y crítica del proceso de incubación, ya que durante ella ocurre la formación de los órganos y el crecimiento del embrión. Por esta razón, la estabilidad de las condiciones internas resulta esencial, pues pequeñas variaciones en los parámetros ambientales pueden afectar seriamente su viabilidad.

En esta etapa, la temperatura debe mantenerse con alta precisión, debido a que el metabolismo embrionario depende directamente del calor suministrado. Asimismo, el crecimiento del embrión incrementa la producción de calor metabólico y CO_2, por lo que se requiere una ventilación adecuada para asegurar el intercambio gaseoso.

De igual forma, el volteo periódico de los huevos, realizado varias veces al día, es indispensable para evitar la adherencia del embrión a las membranas y favorecer una distribución uniforme del calor y los nutrientes.

3.1.4. Lineamientos del Plan Nacional de Lucha contra la Pobreza y el Hambre (2022–2026)

El diseño de esta incubadora con tecnología IoT no es solo un ejercicio de ingeniería. Su existencia está enmarcada en la estrategia de desarrollo del país, establecida por el Plan Nacional de Lucha contra la Pobreza y para el Desarrollo Humano 2022–2026 (PNCL-DH). Es crucial entender que el proyecto utiliza la tecnología como motor para alcanzar metas económicas y sociales de Nicaragua.

3.1.4.1. Lineamiento 3: Desarrollo de Talentos Humanos e Innovación

El proyecto se alinea de manera directa con este lineamiento, que prioriza el uso de la ciencia y la tecnología para impulsar el desarrollo. El Plan pide" Fortalecer la investigación y la innovación en temas prioritarios para el país" (Pág. 94). Lo que estamos haciendo es tomar la Electrónica y la tecnología IoT y aplicarlos para solucionar un problema del sector productivo. Usar sensores de alta precisión y conectividad remota para automatizar una incubadora familiar es la forma tangible de trasladar la innovación digital a la base productiva, cumpliendo con la exigencia de generar soluciones tecnológicas propias y aplicables.

3.1.4.2. Lineamiento 4: Más y Mejor Producción en el Campo

El objetivo final del diseño es el incremento de la productividad, lo cual ataca directamente este lineamiento. El Plan insiste en "Generar y Trasnferir Tecnologías aplicadas a la Producción, Que incrementan los rendimientos" (Pág. 113). Aquí es donde el enfoque tecnológico se vuelve indispensable: un sistema basado en IoT elimina el error humano y garantiza que la temperatura y la humedad los parámetros más críticos se mantengan perfectos.

La tecnología en esta incubadora es la herramienta más eficiente que tenemos para lograr la meta. Al asegurar una producción más estable y predecible, el diseño apoya el mandato de "fortalecer a las familias productoras" (Pág. 109), convirtiendo un prototipo en una contribución práctica para la eficiencia económica y social.

3.2. Internet de las Cosas (IoT)

3.2.1. Concepto general de IoT

El Internet de las Cosas (IoT) es el siguiente gran paso en la evolución de Internet. Ya no se trata solo de conectar documentos o personas, sino de conectar objetos físicos de nuestro mundo para que puedan "hablar" entre sí y con nosotros. El IoT es la base de incubadora inteligente, y funciona gracias a la interacción de cuatro componentes esenciales:

3.2.1.1. Sensores y Actuadores

Los sensores son los "ojos y oídos" de la incubadora. Son dispositivos inteligentes (como los de temperatura y humedad) que detectan y miden el estado del ambiente físico. La Capa de Detección (Sensing Layer) se integra con todos los objetos disponibles para detectar su estado. Los sensores miden la temperatura interna y la humedad, convirtiendo estas condiciones físicas en datos digitales que el sistema puede entender.

Tras un análisis técnico de los dispositivos disponibles para el control climático, se evaluaron tres modelos principales: SHT30, DS18B20 y el sensor seleccionado, DHT22.

Tabla 1 Análisis Comparativo de Sensores Digitales

Sensor	Variables	Precisión	Comunicación	Ventaja Principal
SHT30	Temp / Humedad	Alta (±0.3 °C)	I^2C	Calibración de fábrica industrial.
DS18B20	Solo Temperatura	Media (±0.5 °C)	OneWire	Sumergible y resistente al ruido.
DHT22	Temp / Humedad	Media (±0.5 °C)	Single-Bus	Equilibrio costo/precisión decimal.

3.2.1.2. Actuadores (Transductores de Salida)

Los actuadores son los componentes responsables de modificar el estado del sistema siguiendo las órdenes enviadas por el microcontrolador. Transforman la energía eléctrica en otra forma de energía (térmica, mecánica, neumática).

- Relés (Actuadores de Conmutación): Funcionan como interruptores controlados eléctricamente. Permiten que una señal de baja potencia (3.3V del ESP32) controle cargas de alta potencia (110VAC del calefactor), proporcionando aislamiento óptico para proteger la lógica del circuito.
- Servomotores (Actuadores de Posición): Son dispositivos que permiten un control preciso de la posición angular.
- Ventiladores: Utilizados para la convección forzada, permiten homogeneizar la temperatura y la humedad dentro de la cámara, evitando zonas muertas térmicas.

3.2.1.3. Microcontroladores

Los microcontroladores (como un Arduino o ESP32) son el pequeño cerebro que procesa la información que viene del sensor. Son esenciales porque toman el dato del sensor y busca minimizar recursos y costos, algo vital para que el proyecto sea viable.

3.2.1.4. Fuentes de alimentación

Una fuente de alimentación eléctrica es un dispositivo encargado de suministrar energía eléctrica a un sistema o equipo electrónico, transformando la energía proveniente de la red eléctrica en valores de tensión y corriente adecuados para su correcto funcionamiento. De acuerdo con la tecnología empleada en su funcionamiento, las fuentes de alimentación pueden clasificarse principalmente en fuentes lineales y fuentes conmutadas, nos enfocaremos en uso de fuente conmutada.

3.2.1.4.1. Fuente de alimentación conmutada

Las fuentes de alimentación conmutadas regulan la energía mediante dispositivos electrónicos de potencia que operan a alta frecuencia en modo de conmutación, lo que permite una transferencia controlada e intermitente de energía. Surgieron como una alternativa más eficiente a las fuentes lineales, reduciendo problemas de peso, tamaño y bajo rendimiento energético.

Actualmente se emplean ampliamente en equipos electrónicos y sistemas de control, debido a su alta eficiencia y diseño compacto. No obstante, su uso puede generar interferencias electromagnéticas y ruido eléctrico.

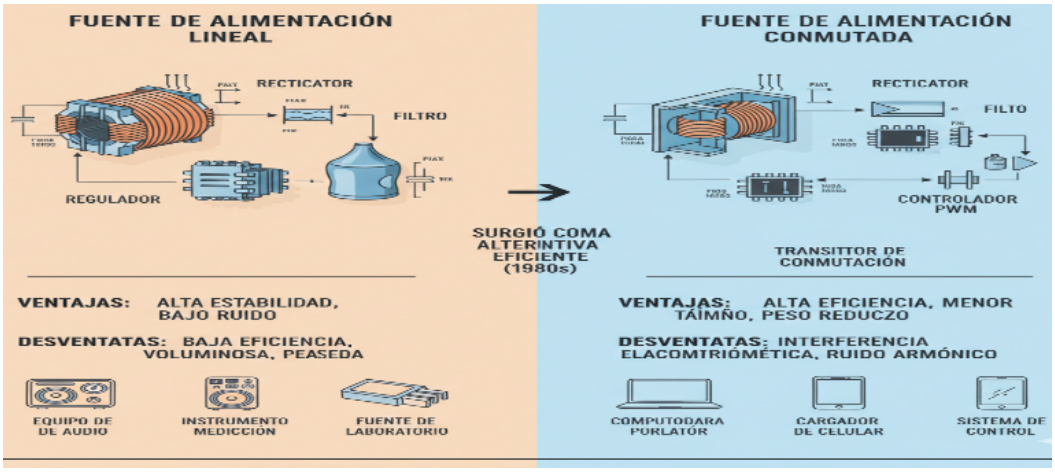


Figura 1 Fuente de Alimentación Lineal y Fuente de Aimentación Conmutada. Fuente: Obtenida del Articulo "Diferencias entre una Fuente de alimentación lineal y conmutada"

3.2.1.5. Conectividad (La Capa de Red)

La conectividad es el sistema nervioso que permite que el microcontrolador envíe sus datos a la nube o a su aplicación móvil. La Capa de Red es, literalmente, la infraestructura que soporta las conexiones (alámbricas o inalámbricas) entre todos los componentes. Es lo que usa el microcontrolador para enviar la lectura de 37.8 ◦ C a su teléfono a través de Wi-Fi o una red móvil. Si la incubadora no tiene una conexión estable (WLAN o red móvil), todo el sistema IoT se rompe.

3.2.1.6. Plataformas de Datos y Servicios

Las Capas de Servicio e Interfaz, una vez que los datos llegan a Internet, necesitan ser almacenados, gestionados y presentados de forma útil. Esto ocurre en la nube (Capa de Servicio) y la plataforma de Datos que es aquí es donde se almacena el historial de temperatura y humedad de las últimas 21 días.

3.2.1.7. Servicios

Esta capa es la encargada de crear y administrar los servicios que necesitamos como usuarios. Por ejemplo, es donde se ejecuta el código que le dice a la aplicación: "Si la temperatura supera los 37.8 ◦ C, envía una alerta al usuario". Interfaz (Frontend): Por último, la Capa de Interfaz es la parte con la que el usuario interactúa (aplicación móvil o un dashboard web), permitiéndole ver los datos y controlar el sistema de forma remota. La clave del IoT es que estos cuatro elementos trabajan juntos para convertir un objeto inerte (la incubadora) en un sistema inteligente que opera de forma autónoma, pero que se puede monitorear desde cualquier lugar.

3.3. Aplicaciones de IoT en agricultura y avicultura.

El Internet de las Cosas (IoT) ha trascendido la domótica y las grandes industrias para convertirse en una herramienta indispensable en el sector agropecuario, un campo donde la precisión ambiental significa directamente mayor rendimiento y rentabilidad. Se trata de una extensión de los servicios que ya ofrece el Internet (Gokhale, Bhat, & Bhat, 2018), pero llevada a los campos y granjas.

3.3.1. Aplicaciones Específicas en la Avicultura

La avicultura, tanto a nivel industrial como familiar, depende totalmente de parámetros ambientales rigurosos. El IoT ofrece la solución más eficiente para asegurar el bienestar animal y la eficiencia productiva:

Monitoreo Ambiental: Se utilizan sensores para registrar la temperatura, la humedad y los niveles de gases tóxicos. Un exceso de calor o la mala calidad del aire impacta directamente la salud y el peso final de las aves.

Incubación Inteligente: Este es el punto de mayor impacto directo en nuestro proyecto. Al aplicar el IoT para automatizar el control de las variables ambientales críticas del proceso de incubación, el sistema garantiza la constancia en la temperatura y la humedad. Esto elimina el error humano en un proceso delicado, maximizando la tasa de eclosión y asegurando que las decisiones del productor estén siempre basadas en datos precisos.

El IoT transforma la agricultura y la avicultura en un sistema basado en la información, lo que se traduce directamente en mayor eficiencia, reducción de costes y mejores rendimientos productivos.

3.3.2. Ventajas: monitoreo en tiempo real, predicción de fallas, análisis de datos.

3.3.2.1. Monitoreo en Tiempo Real

Esta funcionalidad constituye la columna vertebral del control ambiental de la incubadora. El sistema de sensores y la plataforma de conectividad eliminan la dependencia de la supervisión visual o de instrumentos analógicos inexactos.

Precisión y Oportunidad: El productor tiene acceso a un dashboard digital que refleja los valores exactos de temperatura y humedad en el instante mismo en que son medidos.

Intervención Cero Latencia: Ante cualquier desviación de los parámetros óptimos, el sistema genera una alerta instantánea, permitiendo una intervención rápida y precisa.

3.3.2.2. Predicción de Fallas

El sistema IoT trasciende la simple alerta de errores; su verdadero poder radica en la capacidad de anticipar el fallo de componentes. Esto se logra a través de la monitorización continua del comportamiento operativo de los actuadores y los elementos de calefacción.

Detección de Anomalías: El microcontrolador registra la tendencia en los ciclos de trabajo de la resistencia o el ventilador.

Mantenimiento Preventivo: En lugar de esperar el colapso total de un componente lo que podría arruinar un ciclo de producción, el sistema emite una alerta de servicio antes de que el fallo se manifieste.

3.3.2.3. Análisis de Datos

El beneficio a largo plazo del sistema IoT se materializa en la capacidad de generar y almacenar la "huella digital" productiva de cada ciclo de incubación. Inteligencia Productiva: El sistema almacena los datos históricos de 21 días (temperatura promedio por hora, desviación estándar de la humedad, tiempos de volteo, etc.). Al finalizar el ciclo productivo, el productor no solo obtiene polluelos, sino un informe detallado que explica el rendimiento.

3.4. Caracterización de la Comunidad Mario Corea (Palo Verde)

En el contexto local, la comunidad Mario Corea, ubicada en el municipio de El Viejo, Chinandega, específicamente en las cercanías del Ingenio Monte Rosa (conocida también como Palo Verde), tipifica los desafíos de la economía familiar rural. Esta comunidad presenta una densidad poblacional de aproximadamente 30 hogares,

donde el tamaño de los núcleos familiares varía significativamente entre 3 y 8 personas.

La subsistencia de estas familias depende fundamentalmente de una economía diversificada, Entre las actividades principales destacan crianza de aves de corral y cerdos, siembra de cultivos básicos como maiz, platanos, entre otras verduras.

3.4.1. Ubicación de la Comunidad que se contempla



Figura 2 La imagen muestra el área de intervención del proyecto, caracterizada por una densidad de 30 hogares dedicados a la crianza de aves y cultivos básicos bajo condiciones climáticas de alta temperatura ambiental. Fuente: Google Maps

3.4.2. Vulnerabilidad y Necesidad de Innovación Tecnológica

Las familias de Mario Corea enfrentan una alta vulnerabilidad ante las fluctuaciones de los precios del mercado avícola y el desempleo local. La dependencia de sistemas manuales o empíricos profundiza esta debilidad económica, limitando la capacidad de los hogares para sostener su actividad. Ante esta realidad, surge la necesidad de alternativas tecnológicas locales .el diseño de la incubadora busca mejorar la eficiencia productiva.

En esta zona, la producción avícola familiar es un factor clave para el sustento económico; no obstante, se ve limitada por el uso de sistemas de incubación tradicionales que carecen de precisión, lo que genera bajas tasas de eclosión e ineficiencia productiva.

3.5. Componente de Telecomunicaciones

3.5.1. Protocolos de Comunicación para IoT en Zonas Rurales

En zonas rurales, los sistemas IoT enfrentan grandes distancias, bajo acceso a infraestructura, limitaciones energéticas y costos reducidos. Por ello, se emplean protocolos de bajo consumo y largo alcance (LPWAN). Dentro de los protocolos más utilizados se encuentran:

3.5.1.1. LoRaWAN

- Largo alcance (hasta 15 km en áreas rurales).
- Bajo consumo energético.
- Ideal para agricultura de precisión, monitoreo ambiental.
- Baja tasa de datos (0.3–50 kbps).

3.5.1.2. NB-IoT

- Basado en redes celulares (LTE).
- Buena penetración en interiores.
- Dependiente de operadores móviles.
- Consumo moderado.

3.5.1.3. Sigfox

- Ultra bajo consumo.
- Muy baja tasa de datos.
- Dependiente de red propietaria.

3.5.1.4. MQTT-SN

- Variante de MQTT optimizada para redes inalámbricas y sensores.
- Ideal para enlaces inestables o de baja potencia.

3.5.1.5. MQTT (Message Queuing Telemetry Transport):

MQTT es un protocolo ligero de mensajería diseñado para sistemas con bajo ancho de banda, alta latencia o enlaces inestables, como IoT. Características claves:

- Modelo **publish–subscribe**
- Basado en TCP/IP
- Encabezados mínimos (2 bytes)
- Muy eficiente energéticamente

Tabla 2 Protocolos de Comunicación para IoT en Zonas Rurales

Protocolo	Alcance	Consumo	Infraestructura
LoRaWAN	Muy alto	Muy bajo	Gateway propio
NB-IoT	Alto	Bajo	Red celular
Sigfox	Muy alto	Muy bajo	Red privada
MQTT-SN	Variable	Bajo	Gateway

3.5.1.6. Niveles de calidad de servicio (QoS)

- QoS 0: Entrega una vez (sin confirmación)
- QoS 1: Entrega al menos una vez
- QoS 2: Entrega exactamente una vez
- Publisher: envía mensajes
- Subscriber: recibe mensajes
- Broker: gestiona la comunicación (Mosquitto, HiveMQ)

3.5.2. Arquitectura Cliente-Servidor vs. Publish-Subscribe

En el desarrollo de sistemas de comunicación para el Internet de las Cosas (IoT), es fundamental distinguir entre el modelo tradicional de red y el modelo orientado a eventos utilizado por el protocolo MQTT.

3.5.2.1. El Modelo Cliente-Servidor

Es la arquitectura estándar de la web (Protocolo HTTP). En este esquema, el Cliente o dispositivo debe solicitar activamente la información al Servidor, el cual procesa la petición y devuelve una respuesta.

Limitación técnica: Existe un acoplamiento temporal; si el servidor no está disponible en el momento exacto de la petición, el dato se pierde. Además, los encabezados de HTTP son pesados, lo que genera un consumo innecesario de recursos en microcontroladores como el ESP32.

3.5.2.2. El Modelo Publish-Subscribe

A diferencia del anterior, este modelo desacopla al emisor (Publisher) del receptor (Subscriber) a través de un mediador central denominado Broker.

- Desacoplamiento Espacial: Los dispositivos no necesitan conocer sus direcciones IP mutuas, solo necesitan conocer la dirección del Broker.
- Desacoplamiento Temporal: El dispositivo puede publicar un dato y entrar en modo de bajo consumo o concentrarse en el algoritmo PID; el Broker se encargará de entregar el mensaje a los interesados cuando estos estén disponibles.
- Eficiencia de Red: ¿Al no requerir una petición constante de “hay datos nuevos?", el tráfico en la red disminuye drásticamente, permitiendo una comunicación fluida incluso en entornos de baja conectividad como el sector Mario Corea.

Tabla 3 Modelo Publish-Subscribe

Característica	**Cliente-Servidor**	**Publish-Subscribe**
Acoplamiento	Alto	Bajo
Escalabilidad	Limitada	Alta
Latencia	Media	Baja
IoT	Poco eficiente	Óptimo

3.6. Variables Críticas en la Incubación Artificial

La incubación artificial constituye un proceso técnico que busca reproducir de manera controlada las condiciones naturales necesarias para el desarrollo

embrionario de las aves. Su eficiencia depende del manejo preciso de un conjunto de variables ambientales que inciden de forma directa en la viabilidad del embrión y en la tasa final de eclosión.

3.6.1. Temperatura: rangos óptimos, influencia en el desarrollo embrionario.

La temperatura es considerada la variable más crítica dentro del proceso de incubación artificial, debido a su influencia directa sobre el metabolismo y el desarrollo del embrión. El mantenimiento de rangos térmicos estables permite que los procesos biológicos se desarrollen de forma adecuada a lo largo del período de incubación. En el contexto de la comunidad Mario Corea, donde predominan sistemas tradicionales sin control preciso, la inestabilidad térmica representa una de las principales limitantes productivas, lo que resalta la necesidad de incorporar mecanismos de control automatizado mediante sensores y algoritmos de regulación, como el control PID.

Tabla 4 Temperatura: rangos óptimos, influencia en el desarrollo embrionario

Fase del Proceso	**Rango Óptimo (°C)**	**Punto de Consigna (Setpoint)**	**Margen de Tolerancia**
Incubación (Día 1-18)	37.5°C — 38.0°C	**37.8°C**	± 0.5°C
Eclosión (Día 19-21)	36.8°C — 37.2°C	**37.0°C**	± 0.3°C
Límite de Alarma	< 35.5°C o > 39.5°C	N/A	Crítico (Mortalidad)

3.6.2. Humedad relativa: efectos sobre la pérdida de agua del huevo.

La humedad relativa durante la incubación regula la pérdida progresiva de agua del huevo, un proceso esencial para el desarrollo adecuado del embrión y la formación de la cámara de aire. Un nivel de humedad inadecuado puede provocar deshidratación excesiva o retención de líquidos, afectando la capacidad del polluelo

para completar el proceso de eclosión. En sistemas no automatizados, el control empírico de la humedad suele ser impreciso, incrementando la variabilidad de los resultados productivos.

3.6.3. Ventilación y oxigenación: importancia para el intercambio gaseoso.

La ventilación cumple un papel fundamental en el intercambio gaseoso entre el embrión y el ambiente, facilitando la entrada de oxígeno y la eliminación de dióxido de carbono generado durante el desarrollo embrionario. Una ventilación deficiente puede provocar acumulación de gases y estrés fisiológico, comprometiendo la viabilidad del embrión. La automatización del sistema permite regular de manera más eficiente la circulación de aire, asegurando condiciones adecuadas de oxigenación y contribuyendo a la optimización del proceso productivo.

3.6.4. Giro o volteo: frecuencia y consecuencias de una rotación inadecuada.

El giro periódico de los huevos durante la incubación evita la adhesión del embrión a las membranas internas y favorece un desarrollo uniforme. Una frecuencia inadecuada o la ausencia de volteo puede generar deformaciones embrionarias y reducir significativamente la tasa de eclosión. La incorporación de mecanismos automatizados de volteo, integrados a un sistema IoT, garantiza una rotación constante y programada, reduciendo fallas operativas y mejorando la eficiencia del sistema de incubación.

3.6.5. Beneficios de la automatización: estabilidad ambiental, reducción de errores humanos, mejora de la tasa de eclosión.

La automatización de la incubación avícola permite mantener un control continuo y preciso de las variables críticas, minimizando las fluctuaciones ambientales y la intervención manual. La estabilidad térmica y de humedad, junto con la ventilación

y el volteo automatizados, contribuyen a reducir los errores humanos inherentes a los métodos tradicionales. En el marco del presente proyecto, la implementación de tecnologías IoT posibilita el monitoreo remoto y la visualización de datos en tiempo real, fortaleciendo la toma de decisiones y promoviendo una mejora sostenida en la tasa de eclosión, en coherencia con los objetivos de optimización productiva y uso eficiente de los recursos.

3.7. Sistemas de Monitoreo y Plataformas de Análisis de Datos

3.7.1. Tipos de plataformas para visualizar datos (dashboards, aplicaciones).

Las plataformas para visualización de datos en sistemas IoT permiten supervisar, organizar e interpretar en tiempo real la información generada por sensores y actuadores. Entre las más utilizadas se encuentran los dashboards web y las aplicaciones de monitoreo, las cuales facilitan la representación de variables críticas mediante indicadores, gráficos temporales, alertas y estados del sistema. En aplicaciones de incubación automatizada, estas plataformas permiten observar parámetros como temperatura, humedad relativa, ciclos de volteo y condición de los actuadores, favoreciendo el seguimiento continuo del proceso y la detección oportuna de anomalías.

Dentro de este tipo de herramientas, Adafruit IO constituye una plataforma de visualización y monitoreo ampliamente empleada en proyectos IoT, ya que permite mostrar datos en paneles gráficos, indicadores numéricos y controles interactivos de forma accesible. Su integración con protocolos ligeros como MQTT la hace adecuada para sistemas que requieren transmisión periódica de datos y supervisión remota, especialmente en aplicaciones de automatización de bajo costo. En contextos con conectividad variable, este tipo de plataformas resulta de gran utilidad para mantener el registro histórico de variables y apoyar la toma de decisiones sobre el comportamiento del sistema.

4.5.1.1. Generación de alertas

Las alertas automáticas son notificaciones que se activan cuando una variable monitoreada supera umbrales definidos o presenta un comportamiento anómalo. Pueden basarse en reglas estáticas o en análisis dinámico, y su finalidad es detectar condiciones críticas, reducir el tiempo de respuesta y prevenir fallas mayores.

3.7.2. Generación de alertas y predicción de fallas.

Los sistemas IoT pueden implementar umbrales (alertas por temperatura/ humedad fuera de rango), reglas basadas en eventos (fallo de sensor, desconexión), y técnicas de detección de anomalías (z-score, EWMA, o modelos de machine learning supervisado y no supervisado) para prevenir pérdidas. En mantenimiento predictivo, el uso de telemetría y modelos de degradación permite anticipar fallas de componentes (resistencias de calor, motores de volteo, humidificadores) y programar reemplazos o ajustes [Lee et al., 2015; Jardine et al., 2006].

3.8. Modelos Conceptuales para el Diseño de un Sistema Automatizado

3.8.1. Estructura básica de un modelo conceptual: entradas, procesos, salidas.

Un modelo conceptual de sistema describe, a alto nivel, cómo un sistema transforma entradas en salidas mediante procesos que satisfacen una función o necesidad. Esta visión funcional es la base del model-based systems engineering (MBSE): identificar entradas, procesos (funciones/transformaciones) y salidas (resultados) para definir requisitos, interfaces y pruebas. La representación ayuda a separar alcance del sistema, límites y relaciones con actores externos.

Componentes básicos:

- Entradas (Inputs): señales físicas (temperatura, vibración), datos (base de datos, configuración), energía, comandos de usuario.

- Procesos (Funciones): adquisición y pre-procesamiento de señales, filtrado, extracción de características, lógica de decisión, control y comunicación.
- Salidas (Outputs): acciones físicas (actuadores), alarmas, registros históricos, comandos a otros sistemas y visualizaciones.

3.8.2. Representación de la interacción entre sensores, actuadores y software.

- Capa física: sensores y actuadores conectados a nodos de adquisición/actuación (I/O).
- Capa de control embebido: microcontrolador/PLC que adquiere señales, ejecuta control en tiempo real y acciona salidas.
- Capa de integración/mediación: gateways o middleware (ej. protocolos OPC UA, MQTT) para comunicación con sistemas superiores.
- Capa de supervisión/visualización: SCADA, dashboards o aplicaciones web (Shiny/Dash) para historización, alarmas y operación

3.8.3. Modelado de interfaces y señales

Diagramas de bloques y modelos de comportamiento (p. ej. SysML para MBSE) para representar las conexiones físicas y lógicas entre sensores, controladores y actuadores, y los flujos de datos/estado. Los perfiles SysML y extensiones para interacción física facilitan mapear señales físicas a modelos de simulación (Simulink, modelos dinámicos).

3.8.4. Mecanismos de control (Control PID, automatización por eventos).

El PID es el controlador de lazo cerrado más usado en la industria para regular variables continuas (nivel, temperatura, velocidad). Combina tres acciones: P (proporcional), I (integral) y D (derivativo) que, correctamente sintonizadas, permiten estabilidad y respuesta adecuada.

Diseño y sintonización son métodos clásicos: Ziegler–Nichols, método por respuesta a escalón, sintonización automática en controladores comerciales. Y sus consideraciones: límite de saturación, anti-reset windup para I, filtrado del término D y robustez frente a ruido.

3.9. Contexto del Sector Avícola en Nicaragua

El sector avícola desempeña un papel estratégico dentro de la economía familiar rural en Nicaragua, constituyéndose como una fuente relevante de alimentos y de ingresos para numerosas comunidades. No obstante, este sector enfrenta limitaciones estructurales que afectan su sostenibilidad y competitividad, especialmente en zonas rurales con acceso limitado a tecnologías productivas, como ocurre en el sector Mario Corea, ubicado en el departamento de El viejo, Chinandega .

3.9.1. Importancia del pollo y el huevo en la dieta nacional.

El pollo y el huevo representan alimentos de alto valor nutricional y accesibilidad relativa dentro de la dieta nacional, siendo fundamentales para la seguridad alimentaria de la población. En comunidades como Mario Corea, la producción avícola familiar contribuye de manera directa a la disponibilidad de proteínas de origen animal, así como al ingreso económico derivado de la venta de excedentes. La mejora de los procesos productivos, particularmente de la incubación, incide directamente en la estabilidad alimentaria y económica de los hogares rurales.

3.9.2. Problemas del sector: dependencia de importaciones, costos, falta de equipamiento.

El sector avícola enfrenta desafíos asociados a la dependencia de insumos importados, el incremento de los costos de producción y la limitada disponibilidad de equipamiento tecnológico adaptado a contextos rurales. Estas condiciones restringen la capacidad de los pequeños productores para mejorar su productividad y competitividad. La utilización de sistemas de incubación tradicionales, con baja

eficiencia y altos índices de pérdida, profundiza la vulnerabilidad económica de las familias dedicadas a la avicultura.

3.9.3. Necesidad de alternativas tecnológicas locales.

Ante las limitaciones del sector, surge la necesidad de desarrollar alternativas tecnológicas locales que respondan a las condiciones técnicas, económicas y sociales de las comunidades rurales. El diseño de un prototipo de incubadora avícola automatizada basado en IoT se enmarca en esta necesidad, al proponer una solución adaptable, eficiente y alineada con las estrategias nacionales de innovación productiva y transferencia tecnológica.

3.10. Impactos Esperados de Tecnologías IoT en la Producción Avícola

3.10.1. Impacto económico: reducción de costos, mayor eficiencia.

La estabilización de parámetros críticos aumenta la tasa de eclosión y reduce mermas, mejorando los ingresos netos. El monitoreo remoto disminuye tiempos improductivos y optimiza consumo energético mediante control más fino (p. ej., modulando calefacción/ventilación). La trazabilidad de datos posibilita decisiones de inversión informadas (calidad de insumos, mantenimiento oportuno). Estudios de IoT agro–industrial reportan mejoras en rendimientos y reducción de costos operativos por automatización y análisis de datos.

3.10.2. Impacto social: empleo, capacitación, empoderamiento comunitario.

La adopción de IoT promueve capacitación técnica local en sensores, control y uso de datos; puede generar microemprendimientos de servicio/mantenimiento. La apropiación tecnológica fortalece la autonomía y el capital social, alineada con políticas de lucha contra la pobreza. La capacitación basada en datos (históricos, alertas) facilita aprendizaje práctico.

3.10.3. Impacto ambiental: eficiencia energética, reducción de pérdidas.

Un control preciso reduce consumo de energía y derroche (sobrecalentamiento, humedad excesiva), disminuyendo huella ambiental. La detección temprana de desviaciones evita lotes fallidos, reduciendo residuos biológicos. Integrar políticas de ventilación eficiente y aislamiento térmico mejora desempeño ambiental del sistema. En agro–IoT, la optimización basada en datos contribuye a sostenibilidad al reducir insumos y pérdidas [FAO, 2014; Wolfert et al., 2017]. Conexión con OG/OE: sustenta OE4 (eficiencia energética) y OE5 (impacto ambiental).

4. Metodología

4.1. Enfoque metodológico

La presente investigación adopta un enfoque cuantitativo, dado que se fundamenta en la medición, análisis y control de variables físicas propias del proceso de incubación artificial, tales como la temperatura, la humedad relativa y el tiempo de volteo de los huevos. Estas variables son expresadas numéricamente y evaluadas mediante simulación. De acuerdo con Hernández Sampieri, Fernández Collado y Baptista Lucio (2018), el enfoque cuantitativo se caracteriza por el uso de datos medibles, el análisis estadísticos y la verificación objetiva de los fenómenos estudiados.

Asimismo, la investigación es de tipo aplicada, ya que busca dar solución a una problemática concreta del sector avícola rural mediante el diseño de un prototipo tecnológico funcional basado en IoT y control PID. En este sentido, Hernández Sampieri et al. (2018) señalan que la investigación aplicada se orienta a la generación de conocimiento con fines prácticos y de intervención directa en la realidad.

4.2. Alcance de la investigación

El alcance del estudio es descriptivo, puesto que se pretende especificar y caracterizar el comportamiento de las variables críticas del proceso de incubación avícola, así como describir el desempeño del sistema automatizado bajo condiciones controladas de simulación. Según Hernández Sampieri et al. (2018), los estudios descriptivos buscan detallar propiedades, características y perfiles de procesos, fenómenos u objetos sometidos a análisis, sin establecer relaciones causales profundas.

4.3. Diseño de la investigación

El diseño metodológico corresponde a un diseño no experimental y de corte transversal. No se manipulan deliberadamente las variables independientes en un entorno real con organismos vivos; en su lugar, el análisis se desarrolla mediante simulaciones computacionales y pruebas técnicas del prototipo en un entorno controlado. Hernández Sampieri et al. (2018) indican que los diseños no experimentales se basan en la observación de fenómenos tal como ocurren en su contexto natural o simulado, sin intervención directa del investigador.

4.4. Contexto del estudio

El estudio se contextualiza socialmente en la comunidad Mario Corea, ubicada en el Ingenio Monte Rosa, Chinandega, como referente del problema productivo y socioeconómico abordado. No obstante, la validación técnica del sistema se realiza en un entorno de laboratorio y simulación, utilizando herramientas de software especializadas para el análisis del control y la arquitectura IoT.

4.5. Procedimiento metodológico

El procedimiento metodológico se desarrolla de manera secuencial, conforme a las etapas sugeridas por Hernández Sampieri et al. (2018):

4.5.1. Fase 1: Revisión documental y análisis técnico

Se realizó una revisión sistemática de literatura científica, técnica y normativa relacionada con incubación avícola, control PID, e Internet de las Cosas. Esta fase permitió identificar rangos óptimos de operación, antecedentes nacionales e internacionales, y criterios técnicos para el diseño del sistema.

4.5.2. Fase 2: Definición de variables de estudio

Las variables de la investigación se clasifican de la siguiente manera:

- Variables independientes: temperatura interna (°C), humedad relativa (%), tiempo y frecuencia de volteo.
- Variables dependientes: estabilidad del sistema (error estacionario), desempeño del controlador PID y comportamiento dinámico del sistema.

4.5.3. Fase 3. Diseño de la arquitectura del sistema

Se diseñó un modelo conceptual del sistema basado en el esquema entradas–procesos–salidas, definiendo la interacción entre sensores, microcontrolador, actuadores y plataforma IoT. Esta arquitectura sirvió como base para la simulación del comportamiento del sistema automatizado.

4.5.4. Fase 4. Simulación del sistema

La simulación constituye el eje central del proceso metodológico y se desarrolló en las siguientes etapas:

1. Modelado matemático del sistema térmico de la incubadora.
2. Implementación del controlador PID para temperatura y humedad.
3. Sintonización del PID mediante métodos clásicos, como Ziegler–Nichols.
4. Simulación de perturbaciones ambientales.
5. Evaluación del error, tiempo de respuesta y estabilidad del sistema.
6. Integración del flujo de datos hacia una plataforma IoT simulada.

4.5.5. Fase 5. Análisis de resultados

Los resultados obtenidos de la simulación fueron analizados de manera descriptiva, utilizando tablas y gráficas para comparar los valores de referencia con los valores controlados, evaluando el desempeño general del sistema.

4.6. Herramientas y dispositivos utilizados en la simulación

Para el desarrollo de la simulación se consideraron los siguientes elementos de software:

- MATLAB/Simulink R2021b
- Wokwi Online Simulator
- MQTT Explorer v0.4.0-beta
- Broker: HiveMQ Cloud
- Adafruit Io(Opcional dashboard)
- ESP32 DevKit v1
- DHT22 (±0.5°C, ±2%)
- OLED SSD1306 128x64 I2C
- 3 Módulos Relé
- Servo MG996R
- Buzzer + 3 LEDs
- Bibliotecas:
- WiFi.h, PubSubClient.h, DHTesp.h
- ESP32Servo.h, Adafruit_GFX.h Adafruit_SSD1306.h

5. Desarrollo

5.1. Cálculo para el valor de la resistencia calefactora ambiental

La resistencia calefactora es un elemento encargado de generar calor y mantener condiciones térmicas controladas dentro de un sistema. Su función principal es evitar que la temperatura descienda por debajo del valor requerido, garantizando la estabilidad del ambiente. En una incubadora, este dispositivo resulta esencial para conservar la temperatura óptima necesaria durante el proceso de incubación y eclosión de los huevos. Se consideran los siguientes aspectos fundamentales:

- Energía necesaria para calentar y mantener el aire interno de la incubadora dentro del intervalo de temperatura comprendido entre 24℃ y 37.8℃.
- Energía requerida por los huevos durante su fase de desarrollo embrionario previo a la eclosión.
- Pérdidas energéticas a través de las paredes y demás superficies del sistema de incubación, incluyendo transferencia por conducción, convección y radiación.

A continuación, se mostrará los cálculos de los valores previamente mencionados para determinar el valor de la resistencia calefactora ambiental.

5.2. Energía requerida dentro del sistema de incubación

$$\mathrm{Q} = \mathrm{mc}\Delta \qquad (1)$$

Ecuaciones 1 Formula para calcular el factor calorífico

Donde:

Q =es el símbolo de la transferencia de calor (cantidad de calor).

m = es la masa de la sustancia.

c = es la representación del calor (capacidad calorífica específica).

ΔT = es el cambio de temperatura ($T2 - T1$)

Considerando que la temperatura ambiental oscila entre 24℃ y 34℃ (297.15°K y 307.15°K), tomaremos como punto medio 29℃ (302.15°K) como temperatura ambiental promedio, esta tendrá el valor de $T1$, mientras $T2$ es la temperatura final del aire (temperatura óptima para que el huevo eclosione correctamente) que es de 37.8℃ (310.95°K).

El valor de "C" que es el calor especifico a volumen constante del aire es de

$$C = 718(J\ /K\text{x}Kg)$$

El valor de m (masa) se determinará a través de la ecuación de la densidad absoluta.

$$p = (m/\ V) \qquad (2)$$

Ecuaciones 2 Ecuación de la densidad absoluta

Donde:

$p = Densidad$

$m = masa$

$V = volumen\ de\ la\ sustancia$

Despejando el valor de la masa nos queda de la siguiente manera:

$$m = p.\ V \qquad (3)$$

Ecuaciones 3 Ecuación de la densidad absoluta

El valor de p se determina a través de las diferentes densidades presentadas en el ambiente. Teniendo en cuenta que un huevo de gallina mide en promedio entre 5.5 cm a 6 de largo y un diámetro de 4 a 4.5 cm, se deja mínimo 1 cm de espacio entre los huevos se deja que el valor de V son las dimensiones del recinto de la incubadora las cuales viene determinado de la siguiente formula V=A.B.H, donde A es el largo = 0.4 m, B es el ancho = 0.34 m y H es la altura = 0.22 m.

$$V= 0.4\ m * 0.34\ m * 0.22\ m = 0.02992\ m3$$

Conociendo el valor de Volumen de la sustancia, procedemos a determinar m (masa) y la ecuación quedando de la siguiente manera:

$$m = p.\ V = (1.1360\ kg)\ x(0.02992\ m3) = 0.03398\ kgm3$$

Conociendo la masa se procede a calcular la energía requerida para calentar el aire:

$$Q = m * (T2 - T1) * C = 0.03398kg * (310.95°K - 302.15°K) * (718(J / Kg*k) = 214.69\ J$$

Masa de aire: $m = 0.03398\ kg$

Calor específico del aire: $c = 718\ J/(Kg{\cdot}K)$

Incremento de temperatura: $\Delta T = 12.8°C$ *(de 25°C a 37.8°C)*

Tiempo: $t = 3600\ s$ *(1 hora)*

El tiempo total del calentamiento del aire no debe ser tan brusco, ya que el choque térmico sobre los huevos puede dañarlos, razón por la cual el tiempo total de calentamiento debe ser alrededor de una hora, entonces:

$$Tiempo\ total = 3{,}600\ s$$

$$Q = 214.69\ J$$

Por lo tanto, la ecuación queda de la siguiente forma:

$$Potencia\ requerida = Q/\ t = 214.69\ J/3600\ s = 59\ W$$

5.3. Energía requerida por los huevos de gallina

Para calcular la energía requerida por los huevos, se ocupará la ecuación de calor especifico, aproximando “C” que es el calor especifico a volumen constante del huevo a el calor especifico a presión constante del agua: C = 4.18 $(KJ/Kg*°C)$

Para el caso de “m” según Jiménez y Veloza (2008) explican que el huevo es 72% agua, por lo tanto, la incubadora tendrá una capacidad de 30 huevos, con un peso aproximado de 70g, siendo un peso total de 2.1Kg. En el caso de la variación de temperatura, tomaremos como punto medio 29℃ (302.15°K) como temperatura ambiental promedio, esta tendrá el valor de $T1$, mientras $T2$ es la temperatura final del aire (temperatura óptima para que el huevo eclosione correctamente) que es de 37.8 °C (310.95°K).

$$Q = m * (T2 - T1) * C \qquad (4)$$

Ecuaciones 4 Formula de Energía

$$Q = 2.1\ kg * (310.95°K - 302.15°K) * (4.18\ (KJ/K * Kg)) = 77.2\ KJ$$

Finalmente, para que los huevos alcancen los 37.8°C deben ser lo suficientemente grandes para evitar efectos nocivos de un cambio brusco de temperatura, por lo tanto, los huevos alcanzan el equilibrio térmico después de 1 horas (3,600 s).

$Tiempo\ total$ = 3,600 s

Q = 77.2 KJ

Por lo tanto, la ecuación queda de la siguiente forma:

$$Potencia\ requerida = Q/\ t \qquad (5)$$

Ecuaciones 5 Potencia Requerida para un sistema

Sustituyendo los datos obtenidos en la Ecuación (5):

$$Potencia\ requerida = Q/\ t = 77.2\ KJ/3{,}600\ s = 21.4\ \text{W}$$

5.4. Energía perdida a través de las paredes

La energía térmica que se pierde a través de una pared de la incubadora se determina mediante.

$$Q = m * (T2 - T1) * C$$

$$Q = \frac{A(T2 - T1)}{\frac{1}{h1} + \frac{e1}{k1} + \frac{e2}{k2} + \frac{1}{h2}} \quad (6)$$

Ecuaciones 6. Energía térmica que se pierde a través de una pared

Donde,

$T1 = 29°C = 302.15°K\ (Temperatura\ promedio)$

$T2 = 37.8°C = 310.95°K\ (Temperatura\ adecuada\ dentro\ de\ la\ incubadora)$

$h1 = 10\ m2°K\ (coeficiente\ de\ conveccion\ externo$

$\mathrm{h2} = 20\ \mathrm{m2}°K\ (coeficiente\ de\ conveccion\ interno)$

$e1 = 0.8\ mm = 0.0008m\ (espesor\ de\ la\ lamina\ de\ acero)$

$e2 = 1" = 0.0254m\ (espesor\ de\ aislamiento)$

$K1 = 106\ m°K\ (conductividad\ termica\ del\ zinc)$

$K2 = 0.024\ m°K\ (conductividad\ termica\ del\ aire)$

$A = 0.63m2\ (Area\ de\ una\ cara\ lateral\ externa)$

Sustituyendo los valores de diseño, se obtiene una pérdida aproximada de 4.82 W por pared. Considerando las cuatro paredes laterales de la cabina, la pérdida total de calor es:

$$Qtotal: 4.82 * 4: 19.28W$$

Para calcular la potencia perdida debido al material del que está compuesta la puerta frontal se utiliza la siguiente ecuación, tomando en consideración el área de la puerta de vidrio de la incubadora: A= 0.25m*0.20m=0.05m^2, con un grosor de 3mm (0.003m).

Por lo tanto, el calor perdido a través de las paredes laterales de la incubadora y la puerta de vidrio es la suma de ambas:

$$P = 19.28 + 2.88 = 22.16W$$

El requerimiento energético optimo es la suma de las potencias calculadas anteriormente, es decir:

$$Ptotal = P_aire + P_huevos + P_pérdidas \quad (5)$$

Ecuaciones 7. Requerimiento energético optimo es la suma de las potencias calculadas

Sustituyendo los datos obtenidos en la Ecuación (5):

$$Ptotal = 59W + 21.4W + 22.16W = 102.56W$$

Dicho esto, cualquier resistencia por encima de este valor, funcionará para calentar y mantener en óptimas condiciones los huevos de gallina dentro de la incubadora, siempre y cuando se mantenga dentro del rango de temperatura.

5.5. Determinación de Sensor

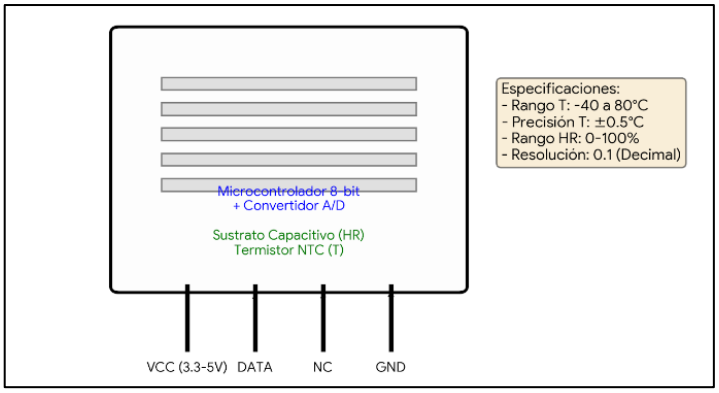


Figura 3. Arquitectura de Sensor DHT22. Fuente: Articulo de Revista Española de Electrónica

Para el control climático del diseño de la incubadora se seleccionó el sensor DHT22, debido a que ofrece una relación costo-beneficio adecuada para el contexto rural del proyecto. Su elección responde a que permite medir temperatura y humedad con una precisión suficiente para el sistema de control, a un costo accesible y con fácil disponibilidad comercial, lo que favorece su implementación en pequeños sistemas productivos. Además, la obtención de lecturas con

resolución decimal lo hace apropiado para el algoritmo PID, contribuyendo a mantener un margen de error reducido durante el desarrollo embrionario.

- Eficiencia de Hardware: Monitorea temperatura y humedad en un solo pin del ESP32, optimizando recursos.
- Resolución Crítica: Ofrece una resolución de 0.1°C, permitiendo que el sistema mantenga los 37.8°C con la precisión que el modelo matemático requiere.
- Robustez Digital: Su protocolo Single-Bus es inmune al ruido de la red de 110V de la zona Mario Corea, a diferencia de los sensores analógicos.

5.6. Determinación de sistemas de acoplamiento y drivers de potencia

En el diseño de sistemas embebidos, la etapa de acoplamiento de potencia es crítica. Los microcontroladores como el ESP32 operan en un dominio de baja energía (3.3V / 12mA por pin), mientras que los actuadores de la incubadora (calefactor, humidificador y ventilación) operan en dominios de potencia media y alta (12VDC o 120VAC). Para cerrar esta brecha, se han determinado dos controladores específicos basados en criterios de aislamiento, eficiencia térmica y protección contra transitorios.

5.6.1. Módulo de Conmutación por Aislamiento Óptico(Relé Optoacoplador).

Para el control del calefactor y el humidificador se seleccionó un módulo relé de 4 canales, no solo por su capacidad de conmutación, sino también por la seguridad y confiabilidad que aporta su aislamiento optoelectrónico mediante PC817, el cual separa eléctricamente el circuito de control del ESP32 de las cargas de mayor tensión y reduce el riesgo de daño ante fallas como cortocircuitos. Además, el módulo incorpora protección frente a picos de voltaje generados por cargas inductivas, como bombas o válvulas del subsistema de humedad, lo que ayuda a preservar los contactos y a extender la vida útil del sistema durante los 21 días de operación continua. Asimismo, su lógica de activación por corriente en

configuración active-low favorece un accionamiento más estable y con menor ruido eléctrico en las líneas de control, mejorando la respuesta general del sistema.

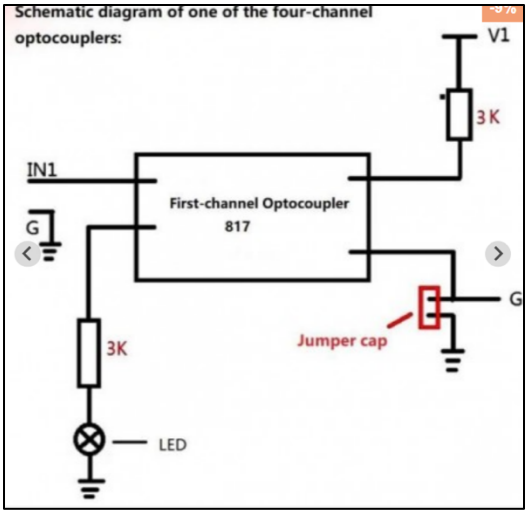


Figura 4. Optoacoplador PC817. Fuente: Nikon Shutter release - Code mod help needed.

5.6.2. Driver de Potencia de Estado Sólido (Puente H L298N)

Para la gestión del ventilador y sistemas de movimiento, el Driver L298N. Su integración se justifica por la necesidad de manejar corrientes de arranque que el ESP32 no puede suministrar.

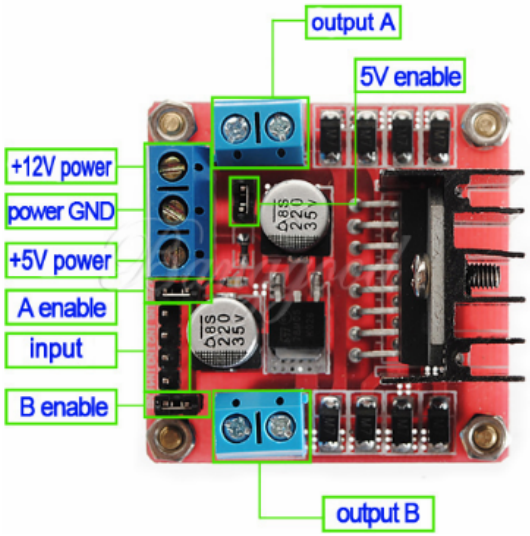


Figura 5. L298N Modulo Driver Motor A pasos. Fuente UNITS ELECTRONICS Tienda de Componentes Electronicos

En lo que respecta control de Inercia y Velocidad a diferencia de un relé, el L298N permite el control por PWM (Modulación por Ancho de Pulso). Esto es fundamental para el ventilador interno, permitiendo ajustar el flujo de aire de forma gradual para evitar cambios bruscos de temperatura al abrir la incubadora.El módulo integra un disipador de calor de aluminio, vital para disipar la energía térmica generada durante la conmutación de los 0.42A del ventilador, asegurando que el driver no sufra degradación térmica por el uso ininterrumpido.

5.7. Robustez Del Diseño

El uso de estos drivers constituye la capa de defensa del diseño del proyecto. La determinación técnica se basa en tres pilares:

Integridad de Señal: Al separar la potencia de la lógica, evitamos que los "rebotes" eléctricos afecten las lecturas del sensor DHT22, garantizando que el setpoint de 37.8°C sea real y no producto de ruido.

Mantenibilidad: Los drivers utilizan borneras de tornillo, lo que permite reemplazar actuadores o fuentes de poder sin necesidad de desoldar componentes, facilitando el mantenimiento técnico en campo.

Seguridad Industrial: El uso de drivers dedicados permite que el sistema cumpla con estándares básicos de seguridad electrónica, separando las etapas de Control (Bajo Voltaje) y Actuación (Alto Voltaje).

5.8. Análisis De Carga Y Cálculos De Potencia

Para garantizar la fiabilidad del sistema durante los 21 días de operación ininterrumpida, se ha realizado un análisis del balance de cargas en las etapas de potencia (AC) y control (DC). Este cálculo justifica la selección del Módulo Relé de 10A, la Fuente de Alimentación de 15A, y los reguladores secundarios necesarios para una incubadora de 30 huevos.

Cálculo de Corriente en la Etapa de Potencia (AC) El actuador principal es la resistencia térmica (o foco calefactor). Para una incubadora de 30 huevos, se estima una carga de 100W práctico, teóricamente nos dió 102.56 watt, El algoritmo PID no mantiene la resistencia encendida al 100% del tiempo. En operación estable, el duty cycle promedio es del 60-70%, lo que reduce el consumo efectivo para alcanzar y mantener los 37.8°C.

$$Ley\ de\ Watt\ (P=V\cdot I): \qquad (8)$$

Ecuaciones 8. Ley de Watt

Utilizando la Ley de Watt (P=V·I):

$I = P/V$

$100W / 120VAC = 0.83A$

Corriente Nominal de la Carga: 0.83A.

Capacidad del Relé: 10A.

Factor de Seguridad: El relé está operando a menos del 10% de su capacidad máxima, lo que garantiza que los contactos internos no sufran de arco eléctrico excesivo ni soldadura por calor, extendiendo la vida útil del controlador significativamente. Balance de Cargas en la Etapa de Control (DC) Es vital asegurar que la fuente de alimentación pueda suplir a todos los componentes simultáneamente:

Tabla 5. Cálculos De Potencia

Componente	Voltaje	Consumo Típico	Consumo Pico	Notas Técnicas
ESP32 (Wi-Fi Activo)	3.3V (de 5V)	160 mA	240 mA	Mediante regulador AMS1117 desde 5V
Sensor DHT22	3.3V	1.5 mA	2.5 mA	Comunicación cada 2s
Pantalla OLED SSD1306	3.3V	20 mA	50 mA	I2C, actualización 1 Hz

Módulo Relé (4 Canales)	5V	240 mA	300 mA	4× (bobina 15mA + LED 45mA + opto)
Servomotor MG995	6V	500 mA	1200 mA	Torque 13 kg·cm ,6V, para 30 huevos
Ventilador DC 12V	12V	300 mA	420 mA	Control PWM mediante L298N
Driver L298N	12V	50 mA	80 mA	Consumo del circuito de control
Resistencia Calefactora	12V	8.33 A	10 A	100W (P=V×I, 12V×8.33A≈100W)
LEDs Indicadores (×3)	5V	45 mA	60 mA	Rojo/Azul/Verde con resistencias
Buzzer Activo	5V	30 mA	50 mA	Alarma de seguridad
Regulador Buck 12V→6V	12V in	50 mA	80 mA	Alimentación dedicada para servo

5.9. Subtotales por Voltaje

• 3.3V Logic: 181.5 mA típico, 292.5 mA pico (ESP32 + DHT22 + OLED)

• 5V Peripherals: 315 mA típico, 410 mA pico (Relés + LEDs + Buzzer)

• 6V Servo: 500 mA típico, 1200 mA pico (Servo MG995 dedicado)

• 12V Power: 8.73 A típico, 10.58 A pico (Resistencia + Ventilador + L298N + Buck).

Con todo lo anterior mostrado se tiene un total estimado (referenciado a 12V): Consumo típico: ~9.5 A *12V (114W) y un Consumo pico: ~12 A * 12V (144W)

5.9.1. Justificación de la Fuente

• Voltaje: 12V DC

• Corriente: 15A (margen de seguridad 25% sobre pico)

• Potencia: 180W

• Tipo: SMPS (Switching Mode Power Supply)

• Eficiencia: >85%

• Protecciones: OCP, OVP, SCP (sobrecorriente, sobrevoltaje, cortocircuito)

5.10. Determinación Del Sistema De Alimentación Y Gestión Energética

La estabilidad de un sistema de incubación artificial depende directamente de la calidad de su suministro eléctrico. Un rizado de voltaje o una caída de tensión momentánea pueden provocar lecturas erróneas en el sensor DHT22 o el reinicio del ESP32, rompiendo el ciclo de incubación. Por ello, se ha determinado el uso de una Fuente Conmutada (SMPS) de 12VDC / 15A.

5.10.1. Justificación de la Tecnología Conmutada

Eficiencia Energética: Las fuentes SMPS operan con una eficiencia superior al 85%, generando menos calor residual, lo cual es vital para no alterar la temperatura ambiente externa de la incubadora.

Rango de Entrada Universal: Capacidad de operar entre 100V y 240VAC, protegiendo al sistema contra las fluctuaciones comunes de la red eléctrica comercial.

Protección Integrada: Estas fuentes incluyen de fábrica protección contra cortocircuitos, sobrecargas y sobretensiones (OVP), actuando como la primera línea de defensa para el hardware costoso (ESP32 y sensores).

5.10.2. Arquitectura de Distribución de Voltaje (Regulación en Cascada)

Dado que el sistema requiere distintos niveles de tensión, se ha implementado una arquitectura de regulación por etapas:

Bus de 12VDC: Alimenta directamente al ventilador de flujo de aire y al driver L298N. Este voltaje garantiza que el ventilador opere a sus RPM nominales para una homogeneización térmica efectiva.

Bus de 5VDC: Obtenido a través del regulador interno del driver L298N o un convertidor Buck externo. Este nivel alimenta la bobina de los relés y el servomotor de volteo.

Bus de 3.3VDC: Regulado internamente por el ESP32 para alimentar la lógica del procesador y el sensor DHT11.

5.10.3. Capacidad de Corriente y Factor de Reserva

El cálculo de consumo total arrojó una demanda máxima de 12A. La elección de una fuente de 15A responde a un criterio de Diseño Robusto con una operación a Carga Óptima.

$$\textit{Carga nominal (pico)} = 12.0\ A$$
$$\textit{Capacidad de la fuente} = 15.0\ A$$
$$\textit{Factor de utilización} = 12A / 15A = 0.80 = 80\%$$
$$\textit{Reserva disponible} = 15A - 12A = 3.0\ A$$

Operar una fuente de alimentación al 80% de su capacidad nominal (NO al límite del 100%) proporciona múltiples ventajas técnicas y operacionales:

Una fuente que no trabaja al límite tiene una vida útil significativamente mayor y una temperatura de operación mucho más baja. El factor de reserva es un indicador crítico de robustez del sistema de alimentación. Se calcula como:

$$Factor\ de\ reserva = (CF - CPp) / Consumo\ pico \quad (9)$$

Ecuaciones 9. Factor de Reserva

$$Factor\ de\ reserva = (15A - 12A) / 12A$$
$$Factor\ de\ reserva = 3A / 12A$$
$$Factor\ de\ reserva = 0.25 = 25\%$$

Gestión de Corrientes de Arranque : El servomotor y el ventilador generan picos de consumo al iniciar su movimiento. La reserva de 3 A adicionales absorbe estos picos sin generar caídas de voltaje que afectarían la conectividad WiFi del ESP32

5.11. Determinación de Microcontrolador

En el mercado de microcontroladores, existen diversas opciones según las necesidades de conectividad, procesamiento, consumo de energía y costo. ESP8266 y ESP32 (de Espressif) se destacan por su conectividad Wi-Fi (y Bluetooth en el caso del ESP32), alto rendimiento (con un procesador de hasta 240 MHz) y bajo costo, lo que los hace perfectos para proyectos IoT como una incubadora automática, donde se requiere monitoreo remoto y control en tiempo real.

ESP32 es una opción ideal para este diseño proyecto de incubadora automática con IoT, considerando las características del diseño de la incubadora: Conectividad integrada (Wi-Fi y Bluetooth), que facilita el monitoreo remoto y la integración con plataformas IoT. Rendimiento superior (procesador dual-core a 240 MHz), que permite manejar múltiples tareas y sensores de manera eficiente.

Gran número de pines GPIO y soporte para PWM, que permite controlar diversos sensores y actuadores necesarios para la incubación. Bajo consumo energético y opciones de gestión de energía, ideales para sistemas autosuficientes o con

baterías. Costo accesible, lo que te permite mantener el proyecto económico y accesible para pequeños agricultores.

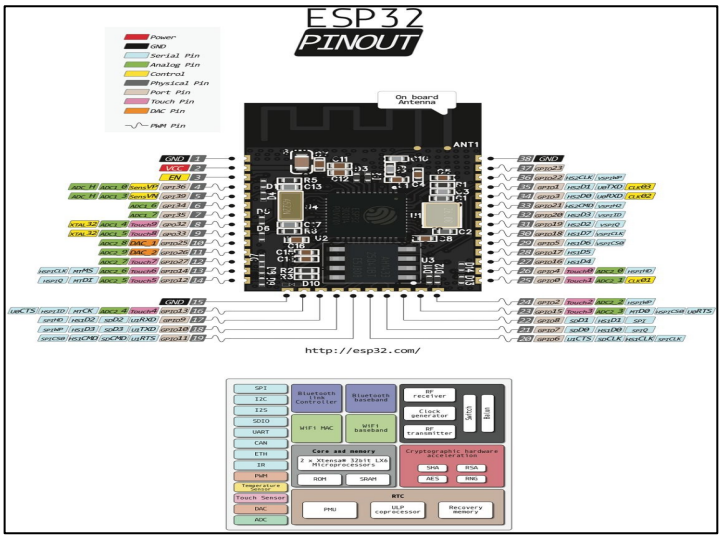


Figura 6. Diagrama de PinOut del ESP32. Fuente: ESP32 Pinout Diagram Overview Scribd

5.12. Regulador de voltaje

Se sugiere un regulador de voltaje reductor, dado que se utiliza una fuente de 12V y necesitamos para los demás componentes un voltaje de entrada inferior a este, por ende, se recomienda el regulador que nos proporcionaría los voltajes inferiores a 12V y necesarios para la activación de los demás componentes electrónicos. El regulador Step-Down XL4015 tiene como función entregar un voltaje de salida constante inferior al voltaje de entrada frente a variaciones del voltaje. Soporta corrientes de salida de hasta 5A, con un voltaje de entrada entre los 4 a 36V y un voltaje de salida entre 1.25V a 32V.

Este regulador permite obtener un voltaje regulado a partir de una fuente con un voltaje mayor, por ejemplo: obtener 5V, 3.3V, 1.8V a partir de una fuente o batería de 12V. Es capaz de manejar una carga de hasta 5A máximo, cuando se emplee para una corriente superior a 3.5 A es recomendable adicionar un disipador de calor.

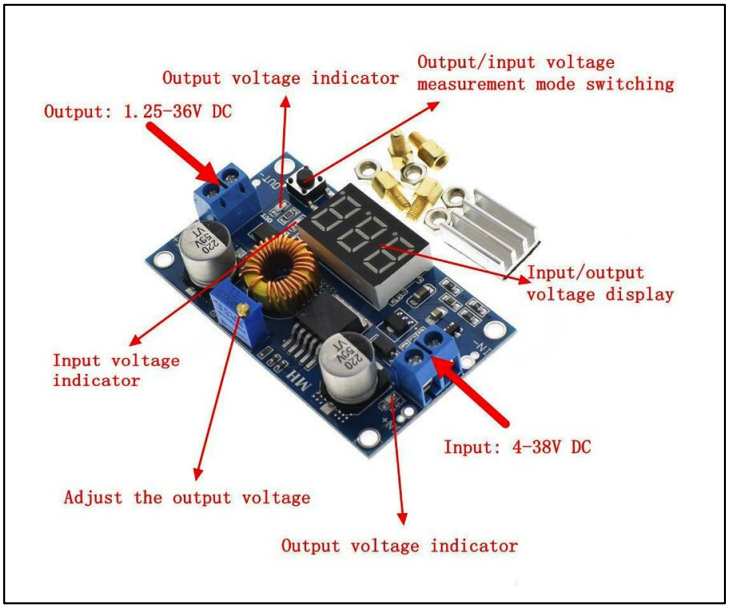


Figura 7. Regulador Step-Down XL4015. Fuente: Kit De 2 Módulos Reductores Dc-dc Xl4015 Ajustables 5a 75w Con Voltímetro Led, Incluye Pilares, Alta Eficiencia 96%, Voltaje Salida 1.25-36v, Protección Contra Cortocircuito Y Sobrecalentamiento MERCADOLIBRE.

5.12.1. Funcionamiento de una fuente de alimentación conmutada

Una fuente de alimentación conmutada puede analizarse dividiéndola en diferentes bloques funcionales. Cada uno de estos bloques cumple una función específica dentro del proceso de conversión de energía. De manera general, las fuentes de alimentación conmutadas, también conocidas como SMPS (Switch Mode Power Supply), operan mediante la conmutación rápida de dispositivos electrónicos de potencia, permitiendo una regulación eficiente de la tensión de salida.

5.12.2. Filtro de compatibilidad electromagnética (EMC)

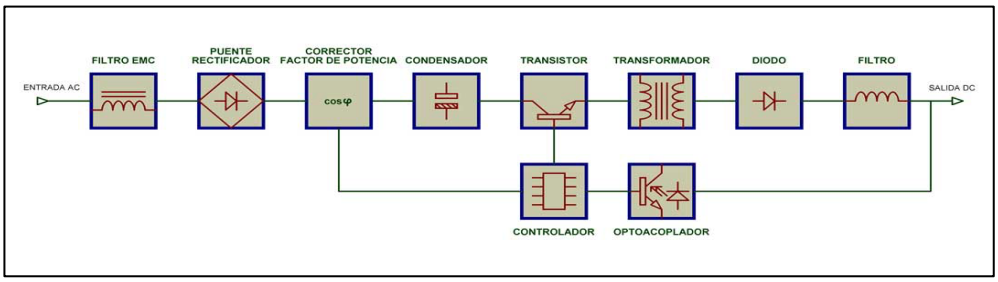


Figura 8. Filtro de compatibilidad electromagnética. Fuente: Cómo funcionan las fuentes de alimentación conmutadas I en FIDESTEC.

Este bloque tiene como finalidad atenuar las perturbaciones eléctricas provenientes de la red, tales como ruido, armónicos y transitorios. Asimismo, impide que las interferencias generadas por la propia fuente se propaguen hacia la red eléctrica. Permite el paso de la corriente en un solo sentido, transformando la tensión alterna de entrada en una señal de corriente continua pulsante, la cual mantiene la forma de la señal alterna original, pero sin cambios de polaridad.

5.13. Diagrama de Bloques

En la figura vemos una representación funcional del diseño de la incubadora en diagrama de bloques, detallando mejor el sistema veamos la siguiente figura:

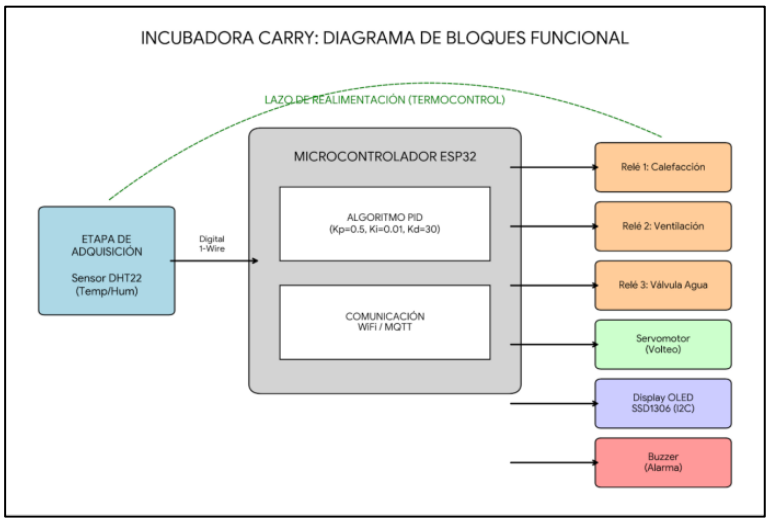


Figura 10. Incubadora Carry diagrama de bloques. Fuente: Imagen Propia creada en Powerpoint

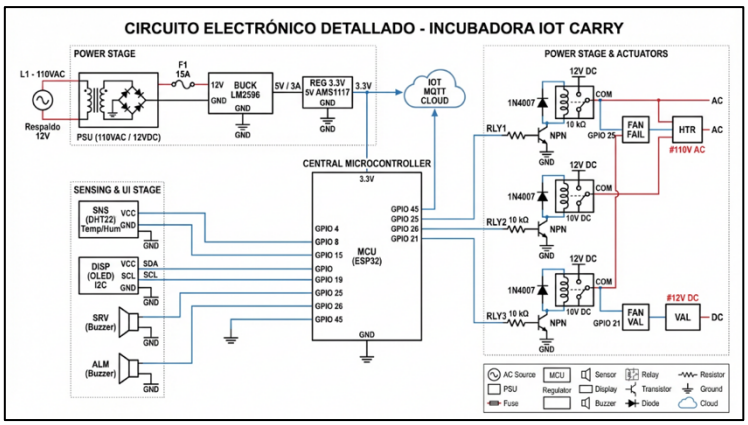


Figura 9. Incubadora IOT carry Circuito electrónico detallado. Fuente: Sistema disenado como imagen referencial.

El esquema de la figura representa un sistema de control de precisión industrializado del diseño. Se divide en cuatro sub-etapas integradas que garantizan la estabilidad de los 37.8°C y el 55% de HR bajo cualquier condición ambiental.

5.14. Gestión de Energía y Respaldo Crítico (PWR)

El sistema utiliza una arquitectura de alimentación redundante. La PSU (Fuente Conmutada) rectifica los 110VAC para entregar 12VDC limpios.

ATS (Automatic Transfer Switch): Este conmutador monitorea la línea de la PSU. Si el voltaje cae, conmuta instantáneamente a la BAT (Batería 12V). Esto evita que el MCU se reinicie y pierda el acumulado del tiempo de incubación.

Etapa de Regulación (REG): Se utilizan convertidores para segmentar voltajes: 12V para potencia (Motores/Relés), 5V para periféricos y 3.3V para la lógica del ESP32.

5.15. Adquisición y Procesamiento (MCU & SNS)

El ESP32 (MCU) es el núcleo donde reside el firmware. Sensor DHT22: Conectado mediante un bus digital. Su alta resolución decimal permite que el Algoritmo PID realice micro-ajustes en la potencia del calefactor, evitando el sobreimpulso térmico.

Bus I2C: Utilizado para el DISP (OLED), permitiendo que el operario en el sector Mario Corea visualice el estado sin necesidad de conexión a internet.

5.16. Etapa de Actuación de Potencia (DRV & RLY)

Aquí es donde el control lógico se convierte en acción física:

Driver L298N (PWM): A diferencia de un relé común, este driver permite al MCU variar la velocidad del FAN (Ventilador), lo representaremos con un rele en la fase de simulación. Esto es vital para "suavizar" la distribución del calor y evitar corrientes de aire agresivas sobre los huevos.

Relés (RLY1 y RLY2): Actúan como interruptores de seguridad. El RLY1 controla la resistencia u elemento de calefacción, mientras que el RLY2 gestiona la VAL (Válvula de Agua) para la humedad. Ambos aislados galvánicamente para proteger al ESP32 de ruidos eléctricos.

Servomotor (SRV): Controlado por pulsos de posición para el volteo mecánico programado.

5.17. Telemetría y Alertas (IOT & ALM)

MQTT Cloud: El sistema publica datos cada poco segundo. Si ocurre un fallo de energía y entra la batería, el MCU envía una alerta crítica a través del Broker MQTT para avisar al usuario.

ALM (Buzzer): Genera una señal auditiva de advertencia local en caso de que la temperatura supere los 39.5°C (límite de seguridad térmica).

5.18. Simulacion del sistema implementado en Matlab, Wokwi y AdaFruit IO

5.18.1. Modelado matemático del sistema térmico y humedad de la incubadora

El diseño de un sistema de control para una incubadora avícola exige una representación matemática rigurosa de las dinámicas térmicas y de humedad. A continuación, detallamos el desarrollo analítico desde los primeros principios, s basado en balances de energía y de materia, con el fin de obtener las funciones de transferencia que describen a la planta.

5.18.1.1. Planteamiento general del sistema

El sistema de incubación está diseñado para albergar una capacidad máxima de 30 huevos de gallina, lo cual representa una masa biológica aproximada de 2.1 kg.

Para garantizar un desarrollo embrionario óptimo, se establecen puntos de operación estrictos: una temperatura interna objetivo de 37.8 °C y una humedad relativa del 55 % HR.

Físicamente, el habitáculo posee unas dimensiones internas de 0.40 m de largo, 0.34 m de ancho y 0.22 m de alto, conformando un volumen útil de $0.02992m^3$. En términos energéticos, los cálculos previos de la estructura determinan una demanda calorífica total cercana a 102.56 W para contrarrestar las pérdidas del recinto y elevar la temperatura de la masa interna. Consecuentemente, el sistema emplea un actuador de calefacción con una potencia de referencia de 100 W, la cual será modulada de manera continua por el sistema de control.

5.18.1.2. Modelo matemático para el sistema Térmico

El comportamiento térmico de la incubadora se modela analíticamente mediante un balance de energía aplicado al volumen de control. La ecuación diferencial que rige la dinámica de la temperatura es:

$$Ceq \cdot dT(t)/dt = Pmax \cdot uT(t - LT) - UA\big(T(t) - Ta(t)\big) \qquad (10)$$

Ecuaciones 10. ecuación diferencial que rige la dinámica de la temperatura

Donde:

Ceq es la capacidad térmica equivalente del sistema

$(J/K).T(t)$ es la temperatura interna medida por el sensor (°C).

$Ta(t)$ es la temperatura ambiente exterior (°C).

$P\max$ es la potencia máxima del elemento calefactor (100 W).

$uT(t)$ es la señal de control normalizada del actuador [0, 1].

LT es el tiempo muerto del lazo térmico (s).

UA es la conductancia térmica global equivalente (W/K).

A partir de las dimensiones internas, el área total de transferencia de calor A se calcula como:

Agrupando las pérdidas por conducción a través de las paredes, puerta frontal de visualización, uniones constructivas y la infiltración de aire, se introduce una conductancia térmica equivalente razonable de $UA = 1.25W/K$. Esto permite definir la ganancia estática térmica (KT) con una entrada normalizada (0 a 1) como:

b

Si la implementación en el lazo de control maneja una señal de actuación expresada en porcentaje (0 % a 100 %), la ganancia equivalente se escala linealmente:

$$\% \, KT = 80/100 = 0.8°C/ \qquad (11)$$

Ecuaciones 11. Ganancia equivalente en el lazo

Para la dinámica transitoria, se fija una constante de tiempo $K\tau T = 150s$ un valor empírico representativo de la inercia térmica del aire interno y el calentamiento superficial.

De aquí se deduce que $Ceq = \tau T \cdot UA = 187.5J/K$. Adicionalmente, se modela un tiempo muerto $LT = 5s$, justificado físicamente por el retardo de muestreo del sensor digital, el transporte convectivo del calor y el tiempo de respuesta del elemento calefactor. Aplicando la Transformada de Laplace al balance de energía linealizado, se obtiene un modelo de Primer Orden con Tiempo Muerto (FOPDT): Y en su forma equivalente para porcentaje de actuación:

Es de vital importancia aclarar que los parámetros del modelo $(K \, \tau \, y \, L)$ emanan del modelado dinámico basado en las ecuaciones diferenciales del sistema y su comportamiento temporal, y no directamente del cálculo estacionario de la potencia requerida. El balance termodinámico justifica el actuador físico, mientras que la función de transferencia captura la evolución de la variable frente a los cambios de estado temporal.

5.18.1.3. Modelo matemático para el sistema de humedad

Se establece un punto de consigna de humedad relativa $H_sp = 55$ % A la temperatura de incubación de $37.8°C$, la humedad absoluta de saturación del aire se estima. Por consiguiente, la humedad absoluta de referencia necesaria para el equilibrio es:

Considerando el volumen útil $V = 0.02992m^3$, la masa de vapor de agua presente en la cámara en régimen estacionario es de apenas $0.753g$ lo que evidencia la alta sensibilidad del sistema ante perturbaciones.

El balance de masa de vapor se enuncia así, donde:

$p_v(t)$ es la humedad absoluta interna (g/m^3).

$\dot{m}_w, \max$ es la tasa máxima de inyección de vapor del humidificador

$u_H(t)$ es la señal de control normalizada del humidificador entre 0 y 1

q es el caudal efectivo de renovación de aire y

$Pva(t)$ es la humedad absoluta exterior.

Ls es el tiempo muerto del lazo de humedad (s)

Asumiendo una tasa de renovación de aire

$q = 0.43m^3/h(1.194 \times 10^{-4}m^3/s)$,

La constante de tiempo para la humedad resulta ser:

$$\tau H = V/q \approx 250s\,. \qquad (12)$$

Ecuaciones 12. Constante de tiempo para la humedad

Considerando un humidificador con un flujo máximo $\dot{m}_{w,max} = 3.6g/h(0.001g/s)$ la ganancia de la planta de humedad en términos de fracción relativa es:

$$K_H = \dot{m}_{w,\,max}/\left(q \cdot \rho_{v,sat}\right) = 0.001/(1.194 \times 10^{-4} \cdot 45.77) \approx 0.183 fracción/u$$

Expresado como porcentaje de humedad relativa, esto equivale a 18.3 %HR/u. Si el actuador opera bajo un escalado porcentual, la ganancia es % K_H = 0.183 HR. Añadiendo un tiempo muerto $LH = 8s$, atribuido a la difusión del vapor y la respuesta del sensor, se formula la función de transferencia:

$$G_H(s) = 18.3e - 8s/(250s + 1)$$

$$\%G_H(s) = 0.183e - 8s/(250s + 1)$$

5.18.1.4. Sintonización por Ziegler-Nichols en lazo cerrado

El método empírico analítico de Ziegler-Nichols (segundo método) determina los parámetros del PID llevando el sistema al límite de la estabilidad. El procedimiento requiere anular las acciones integral y derivativa (Ki = 0, Kd = 0) e incrementar progresivamente la ganancia proporcional Kp hasta observar oscilaciones sostenidas de amplitud constante. En este punto crítico, se miden la ganancia última (Ku) y el periodo de oscilación (Pu). La tabla clásica de sintonización define las reglas:

Tabla 6 Tabla clásica de Ziegler-Nichols

Controlador	Kp	Ti	Td
P	$0.5K_u$	α	0
PI	$0.45K_u$	$\frac{P_u}{1.2}$	0
PID	$0.6K_u$	$\frac{P_u}{2}$	$\frac{P_u}{8}$

Para la estructura paralela, las ganancias se calculan como:

$$K_i = Kp/Ti \text{ y}$$

$$Kd = Kp \cdot Td.$$

5.18.1.4.1. Calculo analítico del lazo térmico

Partimos de la Función de Transferencia

$$G(s) = \frac{K}{\tau s+1} e^{-Ls} \text{ sustituyendo, } GT(s) = \frac{80e^{-5s}}{150+1} \quad (13)$$

Ecuaciones 13. Función de Transferencia

Se impone la condición de fase -180° (π radianes). La condición de fase crítica es:

Resolviendo tenemos frecuencia de cruce:

$$\omega_u \approx 0.3183 rad/s$$

Entonces el periodo crítico:

$$Pu = \frac{2\pi}{\omega_u} \approx 19.74s$$

La ganancia última queda en diagrama Bode:

$$K_u = \frac{\sqrt{1 + \left((150\omega u)\right)^2}}{80} \approx 0.5970$$

Aplicando las reglas del PID, se obtienen los valores de sintonía clásica:

$$K_{pT} = 0.6 \cdot 0.5970 = 0.3582$$

$$T_{iT} = 19.74/2 = 9.868s \rightarrow K_{iT} = 0.3582/9.868 = 0.0363$$

$$T_{dT}dT = 19.74/8 = 2.467s \rightarrow K_{dT} = 0.3582 \cdot 2.467 = 0.8838$$

5.18.1.4.2. Cálculo analítico del lazo de humedad

Mediante un proceso análogo para la función

$$GH(s) = 18.3e^{-8s}/(250s + 1): a\tan(250\omega u) + 8\omega u = \pi$$

Se calcula:

$$\omega u \approx 0.1989 rad/s, P_u \approx 31.60 s \; y \; K_u \approx 2.725.$$

Resultando en el PID clásico:

$$K_{pH} = 0.6 \cdot 2.725 = 1.635$$

$$T_{iH} = 31.60/2 = 15.80 s \rightarrow K_{iH} = 1.635/15.80 = 0.1035$$

$$T_{dH} = 31.60/8 = 3.95 s \rightarrow K_{dH} = 1.635 \cdot 3.95 = 6.457$$

5.18.1.4.3. Ajuste fino conservador para incubación biológica

El método de Ziegler-Nichols favorece una respuesta rápida, pero incurre típicamente en sobre impulsos considerables (aproximadamente 25%). En una aplicación de incubación avícola, un sobre impulso superior a 1 °C representa un grave riesgo de mortalidad embrionaria. Por lo tanto, es mandatorio aplicar un ajuste fino, reduciendo la agresividad del controlador.

Aplicando un factor de mitigación del 50 % sobre las ganancias calculadas, se garantiza una curva de amortiguamiento más suave y segura. Los parámetros ajustados finales son:

Temperatura ajustada:

$$K_p = 0.179, K_i = 0.0182, K_d = 0.442$$

Humedad ajustada:

$$K_p = 0.817, K_i = 0.0517, K_d = 3.228$$

5.18.1.5. Implementación en Matlab de Sistema Térmico y Sistema de Humedad.

El propósito fundamental de la simulación en Matlab radica en la validación teórica de los modelos matemáticos previamente deducidos, así como en la comprobación

del desempeño del algoritmo de control. El uso del entorno de MATLAB y su herramienta Simulink proporciona una plataforma robusta para emular la dinámica de la planta, permitiendo observar las interacciones en lazo cerrado frente a variaciones operativas y perturbaciones exógenas.

5.18.1.6. Implementación y estructura en Matlab

Para llevar a cabo el análisis dinámico, se elaboró un script de simulación en MATLAB encargado de parametrizar el entorno de trabajo. En dicho documento computacional se declararon las variables fundamentales del sistema, se construyeron las funciones de transferencia utilizando la aproximación de retardos de Padé para el manejo del tiempo muerto, y se ejecutó algorítmicamente la sintonización del controlador mediante los postulados analíticos del método de Ziegler-Nichols en lazo cerrado.

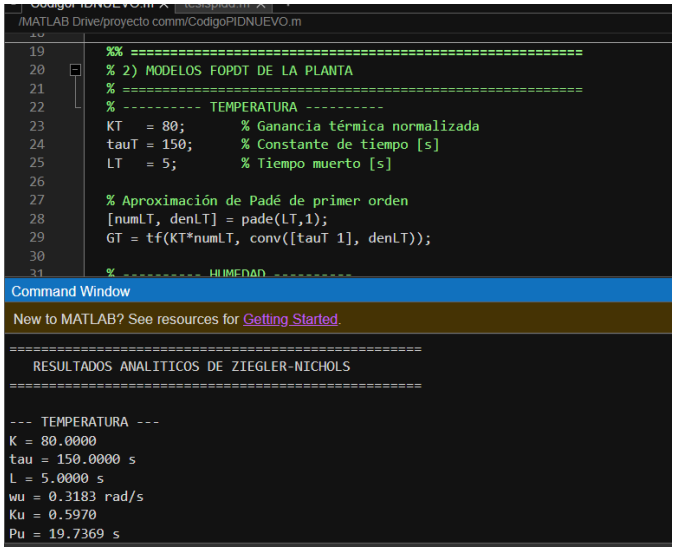


Figura 11. Fragmento del script en MATLAB utilizado para el modelado térmico de la incubadora y la obtención de los parámetros analíticos de sintonización PID por Ziegler–Nichols. Fuente: Elaboración propia y simulaciones desarrollados en Matlab

Adicionalmente, el script invoca funciones de simulación temporal para someter la planta a perturbaciones típicas, recopilando de forma programática las métricas de desempeño para su posterior muestreo de grafica. Cabe destacar que, con el objetivo de preservar la fluidez de la lectura del presente documento, no se incluye el código fuente en este apartado; el script íntegro desarrollado para esta investigación se encuentra detallado y documentado en la sección de anexos.

5.18.1.7. Consideraciones del Subsistema de Humedad

Cabe señalar que el análisis no se limitó al subsistema térmico, ya que para la humedad relativa también se realizó el modelado matemático, la determinación y validación de los parámetros de la planta, así como la sintonización del controlador PID. Sin embargo, en este capítulo se priorizó la simulación del subsistema térmico, debido a que la calefacción constituye la perturbación más crítica y la variable de mayor influencia sobre la estabilidad interna de la incubadora. En consecuencia, la humedad se incorpora como validación analítica de parámetros y sintonización, mientras el análisis dinámico principal se centra en la respuesta del control térmico.

5.18.1.8. Interpretación de la Respuesta Dinámica del Control Térmico

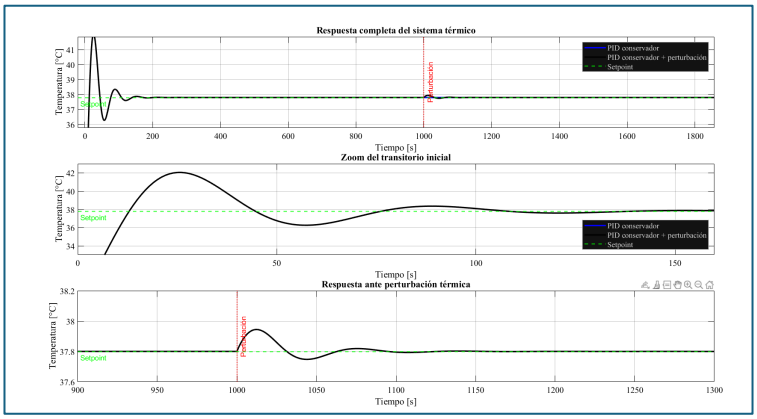


Figura 12. Comportamiento del subsistema térmico mediante control PID conservador, se presenta la respuesta del sistema, análisis ampliado del régimen transitorio y la capacidad de rechazo ante una perturbación térmica externa. Fuente: Modelado efectuados en MATLAB

La figura muestra que el subsistema térmico de la incubadora, controlado por PID conservador, presenta una respuesta estable y convergente hacia el setpoint de 37.8 °C. En el transitorio inicial se observa un sobre impulso moderado acompañado de oscilaciones amortiguadas, las cuales disminuyen progresivamente hasta que la temperatura se estabiliza alrededor del valor de referencia. Este comportamiento indica que el controlador logra corregir el error inicial y conducir el sistema hacia el régimen permanente sin pérdida de estabilidad.

Asimismo, ante la perturbación térmica aplicada en (t = 1000) s, la respuesta del sistema evidencia una desviación pequeña y temporal respecto al punto de operación, seguida de una recuperación rápida hacia el setpoint. Esto permite afirmar que el controlador posee capacidad de rechazo ante perturbaciones externas y mantiene el equilibrio térmico requerido dentro de la incubadora. En consecuencia, la respuesta obtenida valida el desempeño del control propuesto y confirma que el lazo térmico constituye una base sólida para el análisis dinámico del sistema.

Para complementar el análisis del sistema, se utilizaron criterios integrales de error que permiten evaluar el desempeño acumulado del controlador en el tiempo. El IAE, $\int |e(t)| dt$, mide el error absoluto total; el ISE, $\int^{e^2} \blacksquare t) dt$, penaliza con mayor intensidad los errores grandes; y el ITAE, $\int^{t} |e(t)| dt$, asigna mayor peso a los errores que persisten por más tiempo.

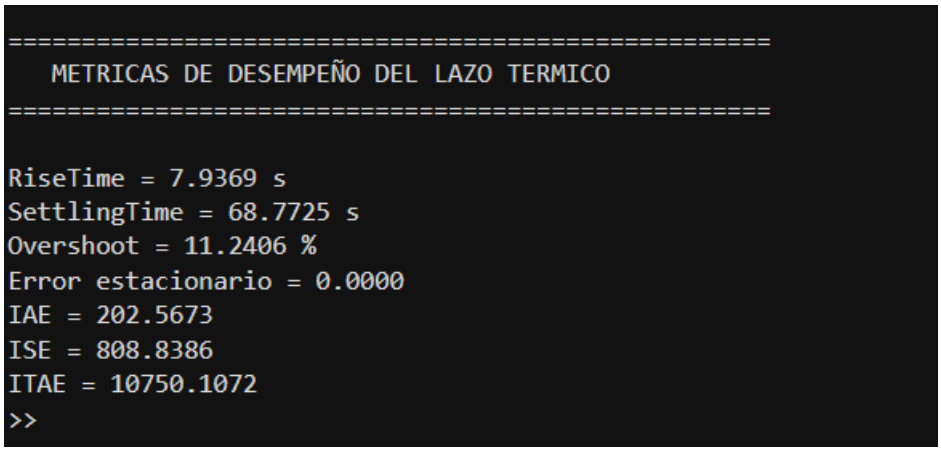


Figura 13. Resultados de las métricas de desempeño del lazo térmico en MATLAB. Fuente: Propia Obtenida durante simulacion

La figura demuestra que el controlador PID del subsistema térmico proporciona una respuesta estable y precisa para la regulación de temperatura en la incubadora. El tiempo de subida de 7.9369 s evidencia una respuesta rápida del sistema, mientras que el tiempo de establecimiento de 68.7725 s confirma el adecuado amortiguamiento de las oscilaciones transitorias.

Aunque se presenta un sobreimpulso de 11.2406 %, este se mantiene dentro de un comportamiento controlado y no afecta la estabilidad del lazo. Además, el error estacionario nulo verifica que el controlador alcanza y mantiene con exactitud la temperatura de consigna. Desde la perspectiva del desempeño global, los índices integrales IAE = 202.5673, ISE = 808.8386 e ITAE = 10750.1072 permiten afirmar que el sistema de calefacción opera con una eficiencia adecuada en la corrección del error térmico

Para validar el control PID, se recrea la planta en MATLAB/Simulink siguiendo este flujo de trabajo:

- Un bloque Step o Constant para proveer el setpoint de referencia.

- Un bloque sum (configuración + y -) para calcular el error a partir de la retroalimentación del sensor.

- El bloque PID Controller, seleccionando el modo de forma Parallel, donde se ingresan directamente los valores de Kp, Ki y Kd.

- Un bloque Transfer Fcn combinado con un bloque Transport Delay para modelar el componente FOPDT de la planta, o un único bloque LTI si se define a través de un objeto de espacio de trabajo.

- Un bloque Sum intermedio tras la planta para añadir dinámicamente las perturbaciones ambientales.

- Un bloque Scope para visualizar y registrar los datos de respuesta temporal.

5.18.1.9. Modelado de la Planta y Configuración del Lazo

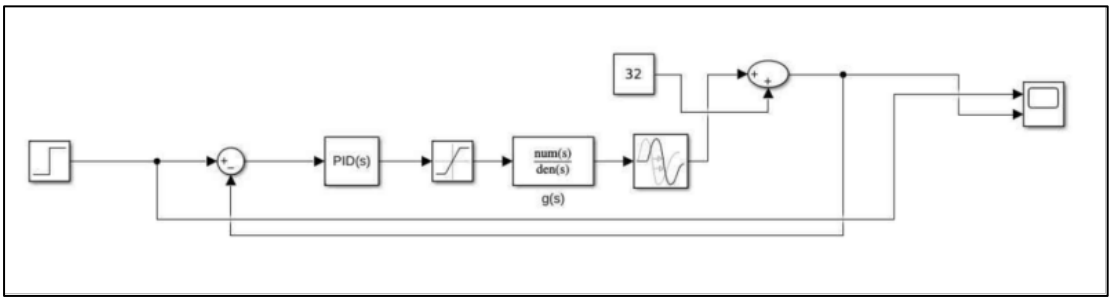


Figura 14. Simulacion de la Planta. Fuente: Propia obtenida en el entorno de simulacion MATLAB Simulink

El flujo de trabajo inició con el ensamblaje del sistema de control en lazo cerrado dentro del entorno MATLAB/Simulink. El diseño se estructuró para simular no solo el comportamiento electrónico, sino también las condiciones físicas reales de una incubadora operando en un clima cálido como el de Chinandega. En el diagrama se observa la integración de los siguientes bloques

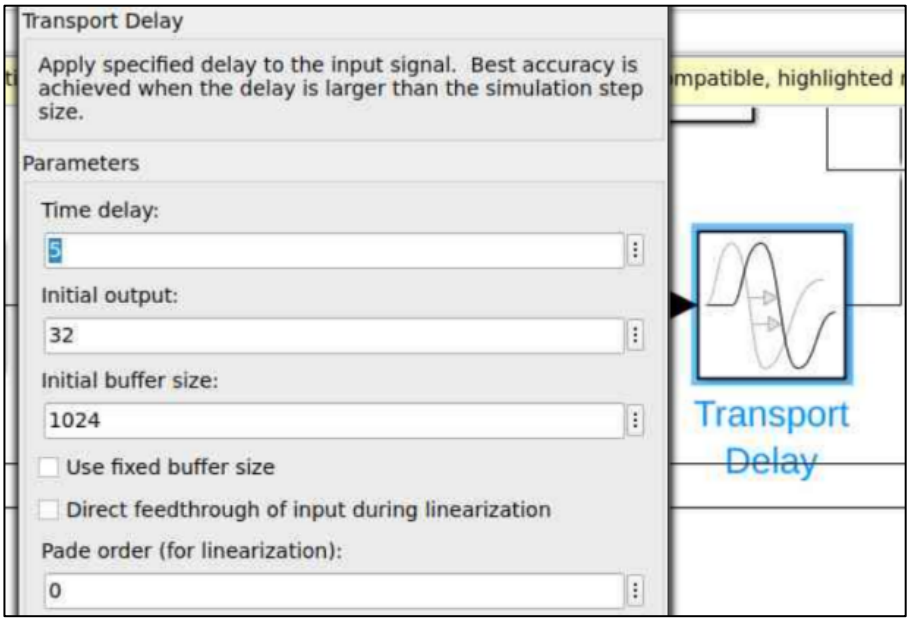


Figura 15. Retardo de Transporte (Transport Delay): Ajustado a 5 segundos para simular la latencia real del sensor DHT22. Fuente: Delay de la onda transportadora. Fuente: Propia obtenida en el entornon de simulaicion de MATLAB Simulink

Parámetros de la función de transferencia fueron calculados mediante el método de curva de reacción. La ganancia (K =0.8) y la constante de tiempo (τ =150) modelan la capacidad de almacenamiento térmico del prototipo, mientras que el retardo de 5 segundos incorpora las limitaciones reales del sensor DHT22.

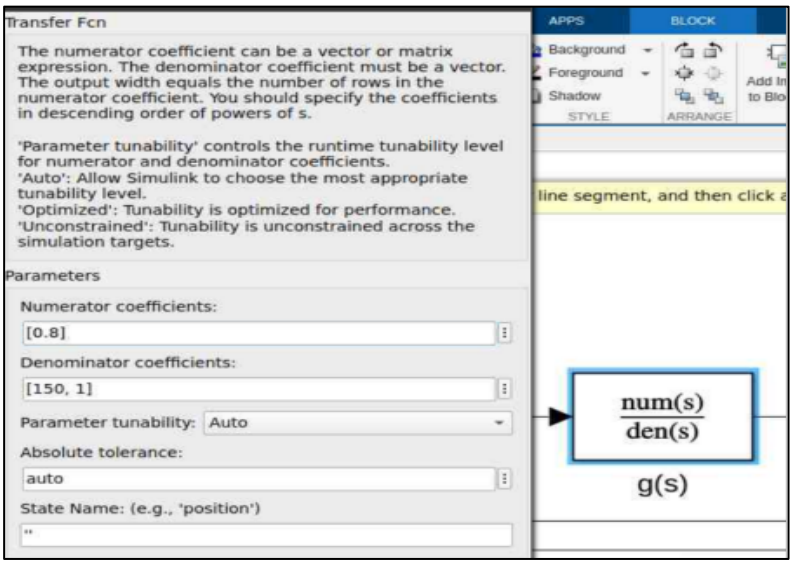


Figura 16. Parámetros de la función de transferencia. Fuente: Propia obtenida en el entorno de simulacion de MATLAB Simulink

5.18.1.10. Compensación de Temperatura Ambiente (Bloque sum+Constant 32)

En los modelos de control clásicos, el sistema suele iniciar en 0. Sin embargo, una incubadora en condiciones reales parte de la temperatura del aire circundante. Descripción: Se integró un bloque Constant con valor de 32 (representando los 32°C promedio de Chinandega) conectado a un bloque sum.

5.18.1.11. Lazo de Realimentación (Feedback)

La realimentación es lo que convierte al sistema en un Lazo Cerrado, permitiendo la corrección automática ante cualquier perturbación.

Descripción: La línea de realimentación se conectó después del bloque de suma de 32°C, cerrando el lazo hacia el comparador inicial (error).

5.18.1.12. Análisis de Resultados y Estabilidad del Sistema

El flujo de trabajo concluyó con la ejecución de la simulación dinámica. La gráfica obtenida en el Scope representa la validación final del diseño electrónico y algorítmico.

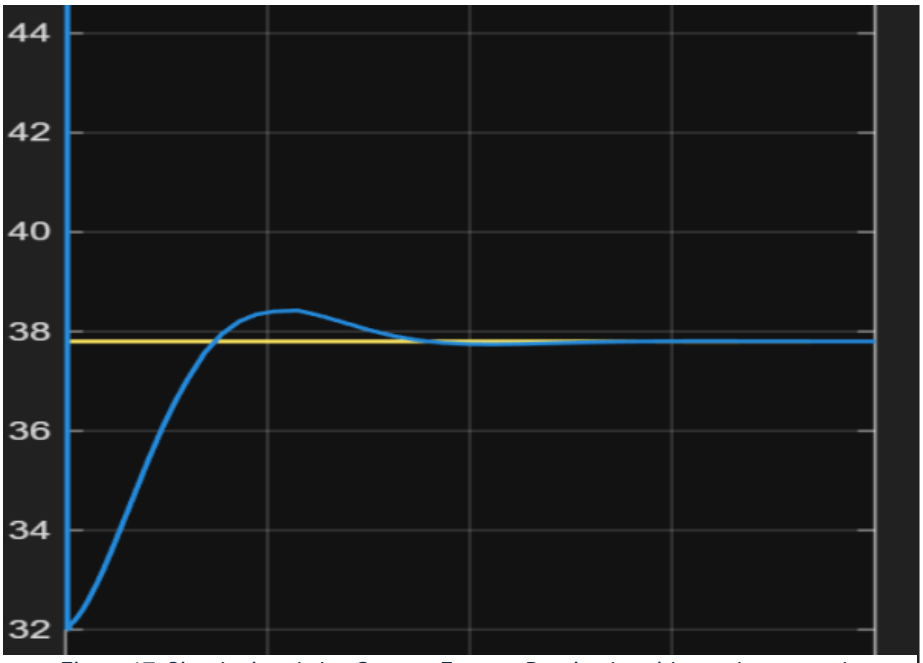


Figura 17. Simulacion de las Curvas. Fuente: Propia obtenida en el entorno de simulacion MATLAB Simulink

Al analizar la curva de respuesta (color azul) frente al setpoint (color amarillo), se identifican:

Punto de Operación Inicial: Se observa que la simulación comienza correctamente en 32°C, validando la efectividad del bloque de compensación ambiental. No presenta saltos térmicos irreales desde los 0°C.

Tiempo de Subida (tr): El sistema alcanza la zona de control en aproximadamente 400 segundos. Este ascenso gradual es ideal para una incubadora, ya que evita el estrés térmico en los embriones.

Sobreimpulso Máximo (Mp): Se registra un sobreimpulso mínimo que alcanza apenas los 38.4°C. Dado que el límite letal se sitúa por encima de los 39°C, este valor de sintonización garantiza un margen de seguridad biológica del 100%.

Tiempo de Asentamiento (ts): El controlador logra estabilizar la temperatura exactamente en 37.8°C cerca de los 1000 segundos. A partir de este punto, la acción integral (ki) elimina cualquier error, manteniendo una línea de temperatura plana y constante.

5.18.1.13. Validación en Simulaciones en Matlab

Los resultados obtenidos en MATLAB/Simulink demuestran que el sistema de control propuesto es asintóticamente estable. El modelo matemático FOPDT sintonizado responde con precisión a las perturbaciones climáticas de la zona de Chinandega, asegurando un error de estado estable nulo y una fluctuación térmica inexistente una vez alcanzado el régimen permanente. Esta validación técnica proporciona la base necesaria para la implementación del código en el microcontrolador físico.

5.18.2. Validación en Entorno de Simulación Wokwi

Al tratarse de una propuesta de diseño técnico, esta etapa representa la consolidación del hardware. Se traslada la arquitectura conceptual del diagrama de bloques de la figura 11, cuya funcionalidad se comprueba íntegramente de manera virtual.

Como fase final del diseño, se realizó la implementación virtual en Wokwi. Esta herramienta permite certificar que la propuesta es viable sin necesidad de construcción física en esta etapa. La implementación en Wokwi es el reflejo físico-virtual de la arquitectura funcional definida anteriormente. Existe una correspondencia exacta entre los bloques y los componentes simulados:

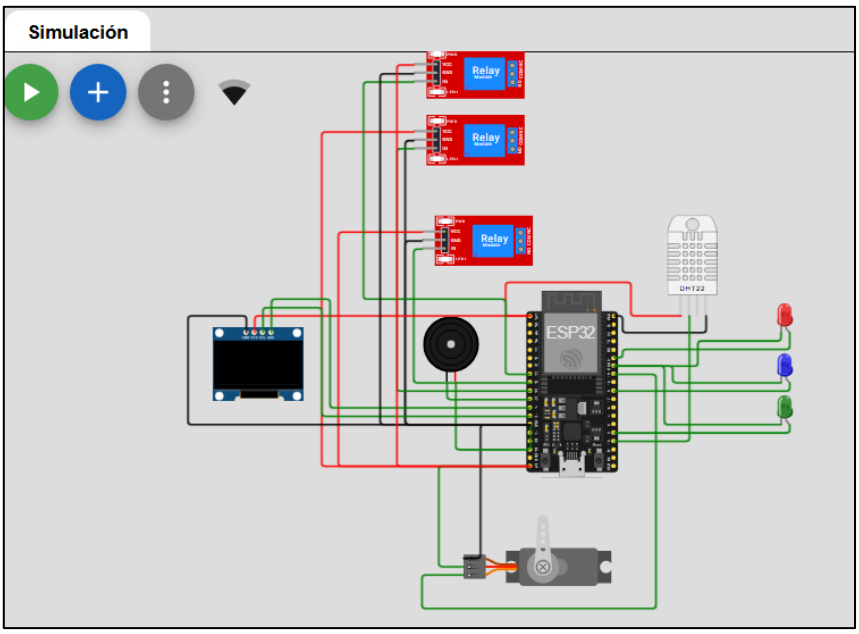


Figura 18. Validación de la propuesta de diseño mediante prototipado virtual. Fuente: Propia obtenida en el entorno de simulacion de WOKWI

- Bloque de Adquisición: Representado por el Sensor DHT22, que provee los datos de entrada al sistema.
- Bloque Controlador: Ejecutado por el ESP32, que procesa el PID y gestiona la comunicación.
- Bloque de Salidas (Actuadores): Relé 1 y 2: Controlan las cargas de calefacción y ventilación.

.

- Relé 3 (Simulación de Driver): En el entorno de simulación, este tercer relé cumple la función del Driver para la válvula de agua, permitiendo validar la lógica de activación para el control de humedad.
- Servomotor, Buzzer y LEDs: Completan el sistema de volteo y alertas visuales/sonoras definidas en la arquitectura original.

En esta etapa se desarrolla la lógica algorítmica que gobierna el hardware simulado. El firmware permite que el **ESP32** actúe como un nodo inteligente, capaz de tomar decisiones autónomas basadas en el modelo matemático PID y reportar estados en tiempo real a través de la red.

5.18.3. Implementación del Algoritmo PID en Código C++

```
29  //========================================
30  estructura PID {
31    doble Kp, Ki, Kd;
32    doble entero = 0 , prevErr = 0 ;
33    doble outMin = 0 , outMax = 100 ;
34
35    doble cálculo ( doble sp, doble pv, doble dt ) {
36      si ( dt <= 0 ) devuelve 0 ;
37      doble err = sp - pv;
38      doble P = Kp * err;
39      entero += err * dt;
40      doble I = Ki * entero;
41      doble D = Kd * ( err - prevErr ) / dt;
42      prevErr = err;
43      doble u = P + I + D;
44      retorna restringir ( u, outMin, outMax ) ;
45    }
46  } ;
```

Figura 19. Código fuente con la implementación de las constantes PID para el control térmico. Fuente: Propia

Se programó el microcontrolador utilizando el lenguaje C++ bajo el framework de Arduino. La clave de este paso es la traducción de las constantes de sintonización obtenidas. El firmware inicializa los sensores y actuadores definidos en el esquema electrónico, estableciendo además la conexión al entorno virtual de red de Wokwi.

```
void setup() {
  Serial.begin(115200);
  Serial.println("Iniciando Incubadora Carry - Fusion Dual MQTT");

  // Pines
  pinMode(PIN_LED_WIFI, OUTPUT);  // Configurar LED WiFi como salida
  pinMode(PIN_RELE_CALOR, OUTPUT);
  pinMode(PIN_RELE_HUMEDAD, OUTPUT);
  pinMode(PIN_LED_TEMP, OUTPUT);
  pinMode(PIN_LED_HUM, OUTPUT);
  digitalWrite(PIN_RELE_CALOR, LOW);
  digitalWrite(PIN_RELE_HUMEDAD, LOW);
  digitalWrite(PIN_LED_TEMP, LOW);
  digitalWrite(PIN_LED_HUM, LOW);
   digitalWrite(PIN_LED_WIFI, LOW);

  // Buzzer PWM
  ledcAttach(PIN_BUZZER, 2000, 8);
  ledcWrite(PIN_BUZZER, 0);

  // DHT
  dht.setup(DHT_PIN, DHTesp::DHT22);

  // OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!oled.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
```

Figura 20. Configuración de pines, canales PWM para el ventilador e inicialización de red WiFi. Fuente: Propia en el entorno de simulacion WOKWI

Gestión de Salidas: Se configuran los pines para el calefactor, humedad y la válvula de agua (relé) como salidas digitales y el protocolo PWM: Se asigna el pin 32 para el control de velocidad del ventilador mediante ledcAttach, permitiendo una homogeneización dinámica del aire.

5.18.4. Lógica de volteo automático y la progresión temporal

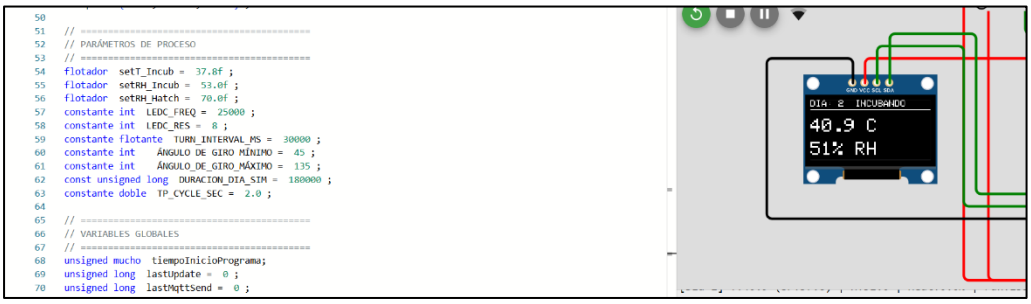


Figura 21. Simulación Wokwi basada en el análisis del código fuente y los resultados obtenidos. Fuente: Propia en la simulación de Wokwi.

5.18.5. Lógica de Simulación Temporal y Volteo de Huevos

El sistema implementa una gestión de tiempo acelerada para permitir la validación del ciclo completo de incubación en un entorno virtual. Esta lógica coordina el cambio de días con la activación mecánica del proceso de volteo.

5.18.6. Gestión del Tiempo y Progresión de Días

El código define la duración de un "día de simulación" mediante una constante, permitiendo observar el comportamiento del sistema a largo plazo en pocos minutos: DURACIÓN_DIA_SIM: Establecida en 180,000 ms (3 minutos).

Cálculo del Día Actual: Se utiliza una función que divide el tiempo total de ejecución entre la duración del día sim, sumando 1 para iniciar en el Día 1.

Visualización en Pantalla: La pantalla OLED muestra dinámicamente el avance del proceso. Por ejemplo, en las capturas se observa la transición del Día 1 al Día 2,

validando que el contador interno funciona correctamente conforme transcurre el tiempo real.

5.18.7. Mecanismo Volteo Automático y Nacimiento.

El volteo de los huevos constituye una función esencial para evitar la adhesión del embrión a la cáscara y favorecer un desarrollo uniforme durante la incubación. En el sistema propuesto, esta acción se ejecuta automáticamente cada 30 000 ms, con un movimiento alternado del servomotor entre 45° y 135°, lo que garantiza un recorrido pendular completo; dicho ciclo se mantiene de forma continua mientras el sistema permanece en la fase de incubación, es decir, antes del día 19.

El código incluye una función de seguridad denominada isHatchingWindow() que detecta si el proceso ha llegado al Día 19. Al alcanzar el día 19, la pantalla OLED cambia el mensaje de "INCUBANDO" a "NACIMIENTO" (o "HATCHING" en la telemetría MQTT).

Seguridad Mecánica: En esta fase, el volteo se detiene automáticamente y el servomotor se bloquea en la posición central (90°) para permitir que los polluelos se posicionen correctamente para la eclosión sin riesgo de lesiones mecánicas.

5.18.8. Lógica de Control de Actuadores y Simulación de Humedad

El lazo principal (loop) ejecuta el control psicrométrico. Es aquí donde se integra un rele como simulador del driver de la válvula de agua.

```
salidaCalor = pidT.compute ( spT, tActual, dt ) ;
outHum = pidRH.compute ( spRH, hActual, dt ) ;

doble  eT = tActual - spT;
ventiladorPWM =  ( eT >  0 )  ?  ( int ) restringir ( 30.0  * eT +  60 ,  0 ,  255 )  :  0 ;
ledcWrite ( PIN_FAN_PWM, fanPWM ) ;

bool  hOn = timeProportionalOn ( outHeat ) ;
bool  rOn = timeProportionalOn ( outHum ) ;
digitalWrite ( PIN_RELAY_HEAT, hOn ?  BAJO  :  ALTO ) ;
digitalWrite ( PIN_RELAY_HUM, rOn ?  BAJO  :  ALTO ) ;
escritura digital ( PIN_LED_OK, hOn ) ;

estadoDeLaVálvula =  ( hActual <  ( spRH -  10.0 ) )  && rOn;
digitalWrite ( PIN_RELAY_VALVE, estadoDeLaVálvula?  BAJO  :  ALTO ) ;
```

Figura 22. Relé como simulador del driver. Fuente: Propia

Gestión de Humedad: Como se observa en la última sección de la imagen, la variable estado de la valvula solo cambia a estado BAJO (encendido) si la humedad actual es al menos 10 puntos inferior al setpoint (hActual < spRH - 10.0). Esto actúa como un seguro para evitar el exceso de humedad en la cámara.

Control de Ventilación PWM: El código calcula dinámicamente el valor de ventiladorPWM basándose en el error de temperatura (eT). Si la temperatura excede el setpoint, el ventilador aumenta su velocidad proporcionalmente para homogeneizar el ambiente.

Tiempo Proporcional (hOn / rOn): Se observa la implementación de la función timeProportionalOn, que permite que relés mecánicos emulen un comportamiento de potencia variable (similar a un dimerizado), lo cual es crítico para que el calor no suba de golpe y se mantenga estable en 37.8°C.

5.18.9. Validación de Conectividad IoT mediante MQTT en Simulación Wokwi

En el desarrollo del prototipo de incubadora IoT, se implementó una arquitectura de comunicación basada en MQTT para validar la conectividad y telemetría en tiempo real, cumplimiento de los requisitos IoT del proyecto.

Limitación de Plataforma: El simulador Wokwi en su versión gratuita no permite el acceso a servidores web HTTP embebidos en el ESP32 desde un navegador externo. Esta restricción técnica no afecta la validación de capacidades IoT, ya que el código genera correctamente la dirección IP (10.10.0.2 en simulación, IP dinámica en hardware real) y las rutas del servidor web (/, /graficas, /set), como se evidencia en el Monitor Serial.

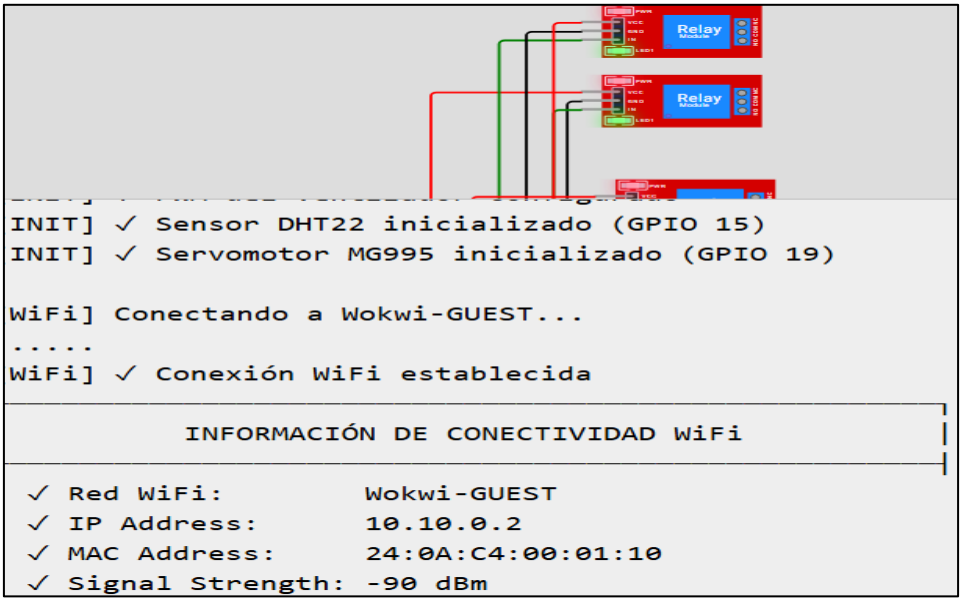


Figura 23. Monitor Serial del ESP32 en Wokwi mostrando información de conectividad WiFi. Fuente: Propia

Monitor Serial de Wokwi (Figura inferior): Muestra la salida del Serial Monitor durante la inicialización del ESP32. Información técnica validada la “Conectividad WiFi: "[WiFi] ✓ Conexión WiFi establecida" a la red Wokwi-GUEST”. Servidor web configurado: Aunque no accesible en Wokwi gratuito, el código inicializa correctamente las rutas HTTP.

```
void publishHiveMQ() {
  unsigned long now = millis();
  if(now - lastHiveMQ >= 10000) {
    lastHiveMQ = now;
    hivemq.publish("incubadora/carry/temp", String(temp, 2).c_str());
    hivemq.publish("incubadora/carry/hum", String(hum, 2).c_str());
    hivemq.publish("incubadora/carry/settemp", String(setTemp, 2).c_str());
    hivemq.publish("incubadora/carry/sethum", String(setHum, 2).c_str());
    hivemq.publish("incubadora/carry/day", String(dayIncubation).c_str());
    hivemq.publish("incubadora/carry/alarm", alarmActive ? "1" : "0");
    hivemq.publish("incubadora/carry/status", "online");
    Serial.println("Publicado en HiveMQ");
  }
}
```

Figura 24. Estructura de tópicos Hive MQTT para la telemetría de temperatura, humedad y estado de actuadores. Fuente: Propia obtenida en el entorno de simulacion Wokwi

Como se observa en el código, los datos se organizan en "topics" específicos (ej: incubadora/carry/temp). Esto permite que el dashboard reciba solo la información necesaria, optimizando el consumo de datos

```
// PUBLICACIÓN MQTT COMPLETA (cada 10s)
if (now - lastMqttSend > 10000) {
  lastMqttSend = now;
  client.publish("carry/telemetria/temp", String(tActual).c_str());
  client.publish("carry/telemetria/hum", String(hActual).c_str());
  client.publish("carry/telemetria/potencia_calor", String(outHeat).c_str());
  client.publish("carry/telemetria/fan_pwm", String(fanPWM).c_str());
  client.publish("carry/telemetria/valvula", valveState ? "1" : "0");
  client.publish("carry/telemetria/dia", String(currentSimDay()).c_str());
  client.publish("carry/telemetria/fase", isHatchingWindow() ? "HATCHING" : "INCUBATION");
  Serial.println("| [MQTT] 📤 Telemetría completa enviada a HiveMQ                    |");
}
}
```

Figura 25. MQTT operativo: Los tópicos están siendo publicados cada 10 segundos según la lógica. Fuente propia: Obtenido de entorno de simulación WOKWI

5.18.10. Variables Críticas publicadas

- Temperatura y Humedad Actual: Para la vigilancia del microclima biológico.
- Potencia de Calor y Fan PWM: Indican el esfuerzo que está haciendo el sistema PID para mantener el equilibrio.
- Estado de Válvula: Registra si la inyección de humedad (Relé 3) está activa.
- Día y Fase: Informa al usuario si el sistema está en periodo de "INCUBACIÓN" o si ha pasado a la fase crítica de "ECLOSIÓN" (lockdown).

```
// PUBLICACIÓN MQTT COMPLETA (cada 10s)
if (now - lastMqttSend > 10000) {
  lastMqttSend = now;
  client.publish("carry/telemetria/temp", String(tActual).c_str());
  client.publish("carry/telemetria/hum", String(hActual).c_str());
  client.publish("carry/telemetria/potencia_calor", String(outHeat).c_str());
  client.publish("carry/telemetria/fan_pwm", String(fanPWM).c_str());
  client.publish("carry/telemetria/valvula", valveState ? "1" : "0");
  client.publish("carry/telemetria/dia", String(currentSimDay()).c_str());
  client.publish("carry/telemetria/fase", isHatchingWindow() ? "HATCHING" : "INCUBATION");
  Serial.println("| [MQTT] 📤 Telemetría completa enviada a HiveMQ                    |");
}
```

Figura 27. Variables Criticas. Fuente: Obtenida del entorno de simulación WOKWI


```
// PUBLICACIÓN MQTT COMPLETA (cada 10s)
if (now - lastMqttSend > 10000) {
  lastMqttSend = now;
  client.publish("carry/telemetria/temp", String(tActual).c_str());
  client.publish("carry/telemetria/hum", String(hActual).c_str());
  client.publish("carry/telemetria/potencia_calor", String(outHeat).c_str());
  client.publish("carry/telemetria/fan_pwm", String(fanPWM).c_str());
  client.publish("carry/telemetria/valvula", valveState ? "1" : "0");
  client.publish("carry/telemetria/dia", String(currentSimDay()).c_str());
  client.publish("carry/telemetria/fase", isHatchingWindow() ? "HATCHING" : "INCUBATION");
  Serial.println("| [MQTT] 📤 Telemetría completa enviada a HiveMQ                    |");
}
```


Figura 27. Variables Criticas. Fuente: Obtenida del entorno de simulación WOKWI

5.18.11. Interfaz de Usuario Local

```
if(now - lastOLED >= 1000) {
  lastOLED = now;
  oled.clearDisplay();
  oled.setTextSize(1);
  oled.setTextColor(SSD1306_WHITE);

  // Linea 1: Titulo (0 pixels from top)
  oled.setCursor(0, 0);
  oled.print("INCUBADORA CARRY");

  // Linea 2: Dia y Fase (8 pixels from top)
  oled.setCursor(0, 8);
  oled.print("Dia:");
  oled.print(dayIncubation);
  oled.print(" ");
  oled.print(isHatchingWindow() ? "ECLOSION" : "INCUBA");

  // Linea 3: Temperatura con status (16 pixels from top)
  oled.setCursor(0, 16);
  oled.print("T:");
  oled.print(temp, 1);
```

Figura 28. Muestreo de código de pantalla. Fuente: Propia obtenida del entorno de simulacion WOKWI

El código no solo envía datos a la nube, sino que genera un servidor web interno y gestiona la pantalla OLED, permitiendo una redundancia en la visualización de datos.

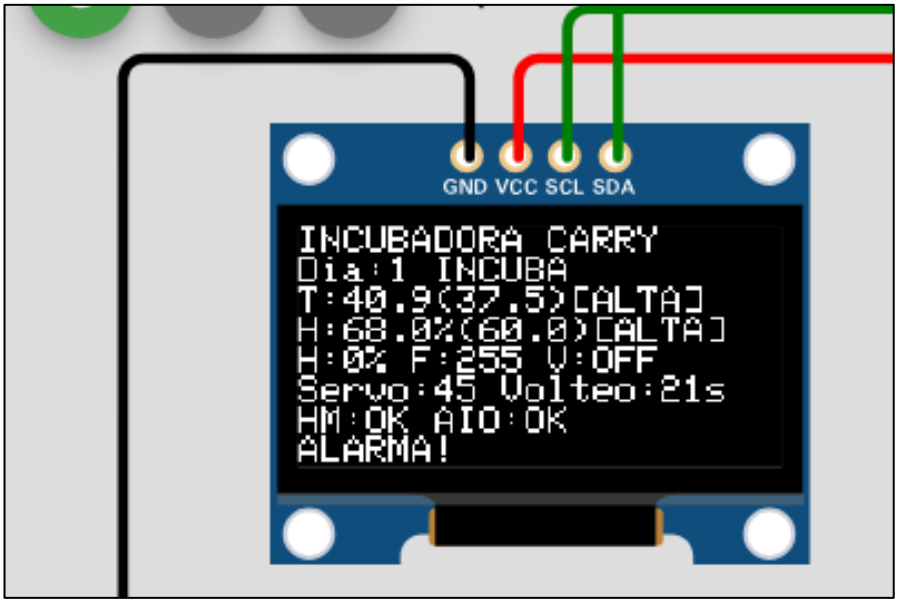


Figura 29. Interfaz local SSD1306 mostrando el estado de incubación en tiempo real. Fuente: Propia obtenida del entorno de simulacion WOKWI

La pantalla muestra el día actual del proceso y las lecturas climáticas exactas, validando que el algoritmo de control está operando correctamente dentro de la simulación.

5.18.12. Simulación a través del entorno de Wokwi

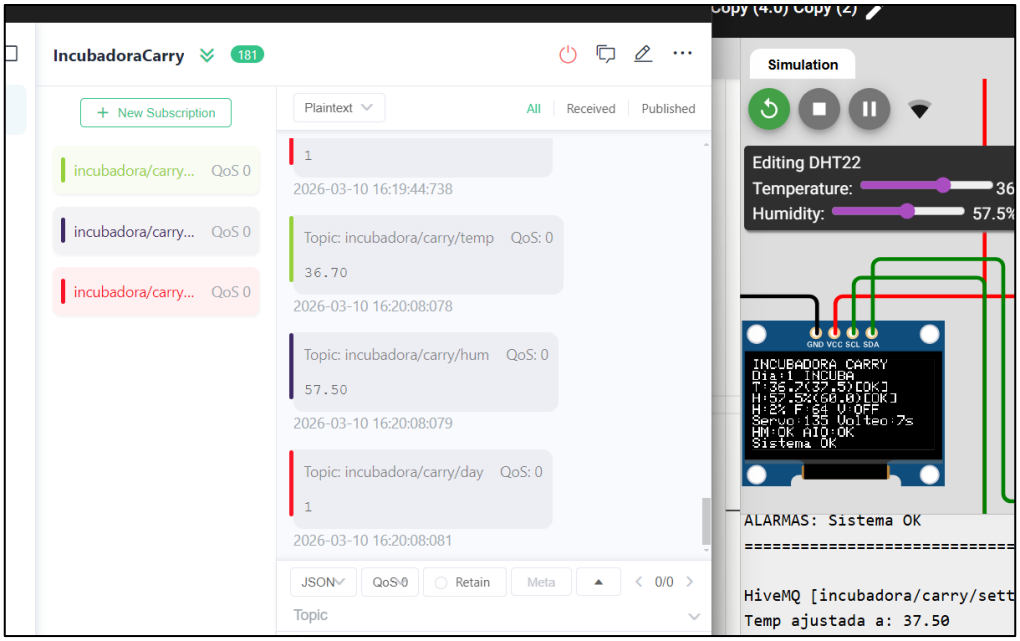


Figura 30. Simulación en entorno Wokwi y Recepción de datos en Broquer MQTT. Fuente: Propia

Esta figura constituye la evidencia final de validación del proyecto diseño de Incubadora Carry, donde se integra la simulación física, la consola de datos y la recepción de telemetría en la nube en tiempo real. A continuación, se detalla la Figura 32:

Se muestra la validación funcional de la arquitectura IoT de la incubadora en el entorno Wokwi y la recepción de datos en el broker MQTT. En el panel izquierdo se observa la llegada de mensajes a tópicos asociados a variables del sistema, lo que confirma la correcta transmisión de información desde el ESP32. En el entorno de simulación también se visualiza el estado operativo del sistema, incluyendo la lectura de temperatura (36.7) y humedad (57.5) en la pantalla OLED, así como la fase de incubación correspondiente.

De forma complementaria, el puerto serial evidencia la lógica de control en tiempo real, mostrando la comparación entre las variables medidas y los valores de referencia. Ante una condición de sobretemperatura(Adicional se cambió el valor de sedpoint en 37.8), la calefacción se desactiva y el ventilador incrementa su acción de enfriamiento, mientras que el mensaje de telemetría confirma el envío exitoso de datos hacia la plataforma en la nube. En conjunto, esta evidencia respalda la robustez del firmware y la correcta integración entre control PID, monitoreo local y comunicación MQTT.

5.18.13. Validación de Transmisión de datos en Adafruit Io

5.18.13.1. Captura 1: Dashboard Adafruit IO - Día 1 (Validación Inicial)

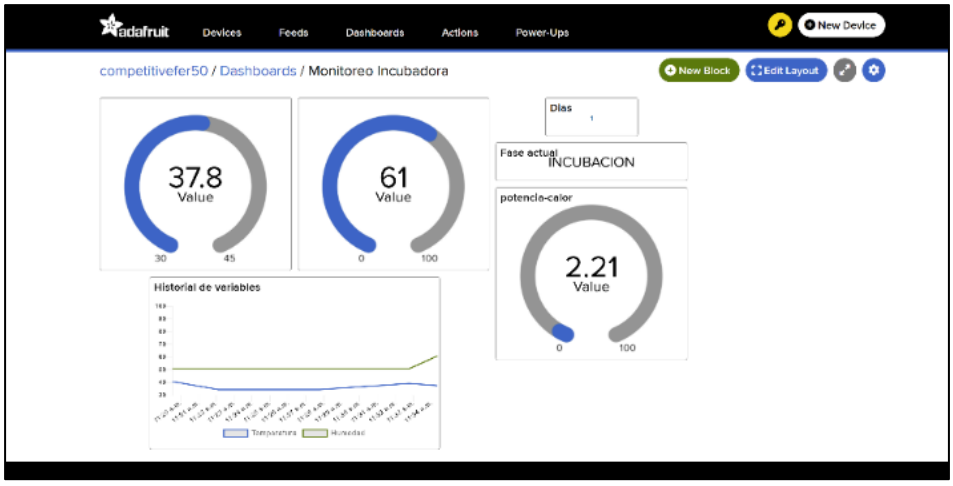


Figura 31. Dashboard Adafruit IO - Día 1 de incubación. Fuente: Propia obtenido del entorno de AdaFruit

Fecha y Contexto: La primera captura corresponde a la prueba inicial del sistema en el Día 1 de simulación de incubación. Esta captura valida la conectividad funcional entre el ESP32 y Adafruit IO, así como la correcta visualización de todas las variables del sistema.

Datos Mostrados en Medidores Circulares (Gauges) son de temperatura: 37.8°C (rango configurado 0-45°C, zona objetivo 37-38°C), humedad: 61% (rango 0-100%,

objetivo 55-65%) y la potencia Calor: 2.21% (rango 0-100%, indica output PID del calentador). Los indicadores de Estado del día: 1 (contador de días de incubación) y la fase actual: INCUBACIÓN (días 1-18, antes del período de eclosión). Teniendo el gráfico Histórico (inferior izquierdo):

- Línea azul: Evolución de temperatura en el tiempo
- Línea rosa: Evolución de humedad en el tiempo
- Eje X: Timestamps en formato dd-mm hh:mm
- Eje Y: Valores de temperatura (°C) y humedad (%)

Elementos de la Interfaz Adafruit IO:

- Botón 'New Block': Permite agregar widgets adicionales al dashboard
- Menú 'Edit Layout': Opción para personalizar disposición de widgets
- Feeds activos: temperatura, humedad, dia, potenciaCaldo, fase

5.18.13.1.1. Interpretación de resultados mostrados

Los datos de la Captura 1 demuestran un sistema funcionando correctamente con temperatura ligeramente elevada (37.8°C vs setpoint 37.5°C), lo que explica la baja potencia de calor (2.21%). El controlador PID está reduciendo activamente la salida para evitar sobrecalentamiento. La humedad se encuentra dentro del rango aceptable (61% ≈ 60% setpoint). El gráfico histórico muestra estabilidad térmica con variaciones mínimas, validando el correcto funcionamiento del control PID dual. La conectividad MQTT con Adafruit IO está completamente operativa.

5.18.13.2. Captura 2: Dashboard Adafruit IO - Día 3 (Validación de Persistencia)

La segunda captura corresponde al Día 3 de simulación, demostrando la persistencia de datos y la capacidad del sistema para mantener históricos durante múltiples días de operación continua. Esta captura también muestra la detección de una anomalía térmica (temperatura significativamente baja).



Figura 32. Dashboard Adafruit IO - Día 3 de incubación. Fuente: Propia obtenido del entorno de AdaFruit

Los datos Mostrados en Medidores Circulares, muestran una temperatura: 26.9°C (rango 30-45°C, valor FUERA del rango objetivo), El gauge muestra indicador visual en zona roja, señalando temperatura BAJA crítica. Esta temperatura está muy por debajo del setpoint de 37.5°C requerido para incubación avícola y humedad: 57.5% (rango 50-60%, en los primeros 18 dias de incubacion).

Indicadores de Estado:

- Día: 3 (confirmando incremento correcto del contador de días)
- Fase actual: INCUBACIÓN (mantiene la fase correcta para días 1-18)

Timestamp Visible**:**

- Fecha/hora: 'March 8th 2026, 6:14:04PM'
- Demuestra registro continuo con marcas de tiempo precisas para cada dato

5.18.13.2.1. Interpretación de Resultados

1. Persistencia de Datos: El dashboard mantiene históricos correctamente después de 3 días de operación simulada

2. Detección de Anomalías: El sistema identifica correctamente una temperatura baja (26.9°C << 37.8°C), mostrando indicador visual de alarma

3. Continuidad de Conectividad: La conexión MQTT se mantiene estable durante múltiples días, validando la confiabilidad del sistema

La temperatura baja registrada (26.9°C) podría corresponder a una condición de prueba deliberada o una falla simulada del calentador. El sistema de alarmas del firmware debería haber detectado esta condición y activado alertas. La humedad se mantiene relativamente estable (57.5% ≈ 60% setpoint), demostrando el funcionamiento independiente del controlador PID de humedad.

5.19. Identificación de la Falla en la Válvula (Relé 3)

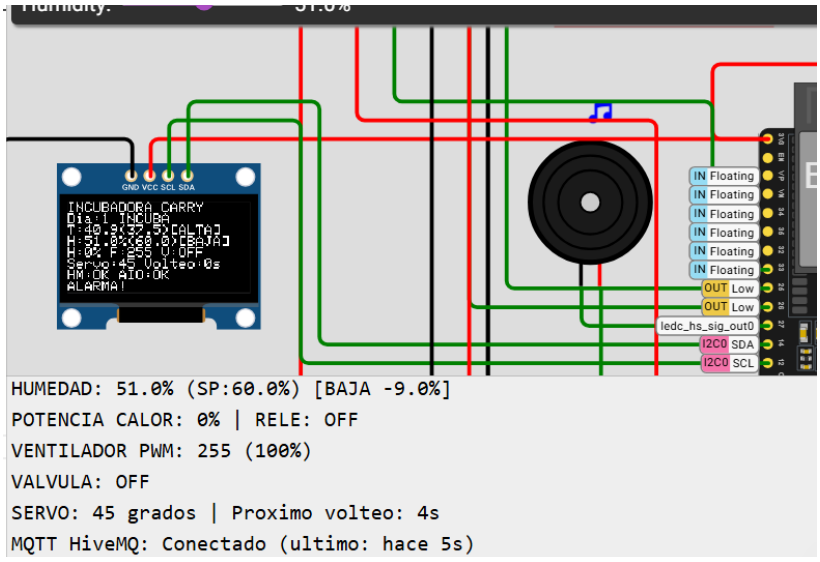


Figura 33. Identificacion de falla en la valvula (Rele 3). Fuente: Propia desde el entorno de simulacion de Wokwi

Según la lógica del código implementado, la válvula de agua (Relay Valve) tiene una condición de activación de "seguridad": solo se activa si existe un déficit de humedad mayor a 10 puntos respecto al setpoint (hActual < spRH - 10.0).

- El Problema en la Simulación: En la imagen, el sistema marca 51.0% RH. Si el setpoint es del 55%, la diferencia es de solo 4 puntos.

- El circuito interpreta que, aunque la humedad está baja, no ha llegado al umbral crítico para activar la válvula (rele). En el monitor serial se lee "Válvula: OFF". Esto es una falla de diseño preventivo para evitar el exceso de humedad, pero en la práctica significa que la humedad no subirá hasta que caiga por debajo del 45%.

5.20. Reacción del Circuito ante el Desvío

Si la humedad continúa disminuyendo o la temperatura sigue aumentando, el sistema de control actúa automáticamente para corregir ambas variables. Cuando la humedad desciende por debajo del umbral definido por histéresis, es decir, 45 % para un setpoint de 55 %, el ESP32 activa el Relé 3 para encender el humidificador o permitir el paso de agua, evitando con ello conmutaciones frecuentes del actuador y favoreciendo la estabilidad del control.

De forma simultánea, si se detecta una sobre temperatura, como el valor de 40.9 °C observado, el sistema prioriza el enfriamiento incrementando la velocidad del ventilador mediante PWM; sin embargo, esta acción puede reducir aún más la humedad relativa, por lo que el subsistema de humedad compensa activando la humidificación. Además, el estado de los actuadores se transmite mediante MQTT, lo que permite supervisar remotamente el comportamiento del sistema e identificar posibles necesidades de ajuste o revisión del suministro de agua.

5.21. Respuesta Sonora y Visual

Debido a que la temperatura está en 40.9°C, el circuito activa el Buzzer (señalado con la nota musical en Wokwi). Esta reacción es integral: el sistema asume que un fallo en la temperatura es más peligroso para el embrión que una desviación leve en la humedad, por lo que enfoca toda su energía en la ventilación.

La simulación permite identificar un comportamiento de zona muerta en el control de humedad. El circuito reacciona protegiendo los actuadores de activaciones constantes, pero ante un desvío crítico, la lógica de seguridad del firmware asegura la apertura de la válvula y la notificación inmediata a través del protocolo MQTT.

5.22. Resultados de la simulación:

- Estabilidad del Firmware: El código se ejecuta de forma fluida, gestionando las entradas del sensor y las salidas de los actuadores.

- Verificación de Señales: Se comprueba que los relés conmutan correctamente según el setpoint y que el servomotor responde a los tiempos de volteo programados.

5.23. Presupuesto

El presente presupuesto detalla los costos de implementación de una incubadora avícola automatizada con tecnología IoT para 30 huevos, diseñada para la comunidad Mario Corea, Chinandega. Los precios están expresados en córdobas nicaragüenses (C$) y reflejan el mercado local de Nicaragua (marzo 2026).

5.23.1. Componentes Electrónicos

En el apartado de componentes electrónicos, se detallan los dispositivos que integran el sistema de control y actuación del prototipo, destinados a garantizar la automatización completa del entorno de incubación.

Tabla 7. Componentes Electronicos

Componente	**Especificación**	**Cant.**	**Precio Unit. (C$)**	**Total (C$)**
Fuente SMPS 12V/15A/180W	Switching, OCP/OVP/SCP	1	585	585
Buck LM2596 12V→5V/3A	Eficiencia 92%, ajustable	1	310	310
ESP32 DevKit	Con regulador AMS1117 integrado	1	320	320
Sensor DHT22	Temp/Humedad ±0.5°C	1	170	170
Display OLED SSD1306	128×64 I2C	1	200	200
Módulo Relé 4 canales	5V optoacoplado 10A	1	252	252
Servomotor MG995	Torque 11kg·cm @ 5V	1	310	310
Driver L298N	Dual H-bridge 2A/canal	1	162	162
Ventilador DC 12V 80mm	300-420mA	1	142	142
Resistencia calefactora 100W	Cerámica, aislada	1	245	245

Buzzer piezoeléctrico 5V	Alarmas y servicio	2	25	50
LEDs 5mm (R, G, B)	Con resistencias 220Ω	3	12	36
Capacitores varios	1000µF/25V, 100nF cerámicos	Lote	65	65
Diodos 1N4007	Flyback protection	5	6	30
Fusibles y portafusibles	15A AC, 3A DC	Set	50	50
Cable alimentación 110VAC	Con enchufe tipo A	1	80	80
Cables AWG 14, 18, 22	10m de cada calibre	Set	225	225
Conectores y terminales	Borneras, terminales, estaño	Set	175	175
PCB perforada universal	O impresa custom	1	125	125
SUBTOTAL ELECTRÓNICA				**C$ 3,532.00**

5.23.2. Materiales Estructurales

La tabla de materiales estructurales describe los insumos necesarios para la fabricación de la cabina física y el soporte de los huevos.

Tabla 8. Materiales Estructurales

Material	**Especificación**	**Cant.**	**Precio Unit. (C$)**	**Total (C$)**
Madera contrachapada 12mm	1 plancha 122×244 cm	1	500	500
Maderapino 2×1"	Refuerzos estructurales, 3m	1	100	100
Vidrio templado	20×15 cm para puerta	1	150	150
Bisagras metálicas	Pequeñas cromadas	2	40	80
Manija metálica tipo L	Cromada	1	32	32
Rejilla ventilación	Metálica 5cm diámetro	2	65	130
Bandeja rejilla galvanizada	35×25 cm para huevos	1	175	175
Tornillos y clavos	Varios tamaños + escuadras	Set	100	100
Pegamento para madera	Resistol/silicón	1	65	65
Barniz sellador	250ml acabado mate	1	125	125
Lijas varios granos	80, 120, 220	Set	40	40
Material aislante térmico	Fibra vidrio/espuma	1 m²	185	185
SUBTOTAL MATERIALES				**C$ 1,682.00**

5.23.3. Herramientas y Equipos

Se incluye una sección dedicada a herramientas y equipos:

Tabla 9. Herramientas y Equipos

Herramienta	Especificación	Cant.	Precio Unit. (C$)	Total (C$)
Multímetro digital	Mide V, A, Ω, continuidad	1	325	325
Soldador estaño 30-60W	Con soporte y accesorios	1	275	275
Juego pinzas/destornilladores	Alicates, cortacables	1	375	375
Sierra/caladora madera	Eléctrica o manual	1	600	600
Taladro eléctrico básico	500-800W con percusión	1	1000	1000
Juego brocas madera/metal	Varios diámetros	1	200	200
SUBTOTAL HERRAMIENTAS				***C$ 2,775.00***

5.23.4. Costos Adicionales

La tabla de costos adicionales integra los gastos operativos vinculados a la mano de obra especializada en electrónica y carpintería, así como la logística de transporte y las pruebas de ajuste iniciales.

Tabla 10. Costos Adicionales

Concepto	**Descripción**	**Cantidad**	**Precio Unit. (C$)**	**Total (C$)**
Mano de obra montaje electrónico	Soldadura, ensamblaje circuitos, programación	1 trabajo	650	650
Mano de obra carpintería	Corte, ensamblaje, acabado madera	1 trabajo	500	500
Transporte y envíos	Compras en Managua, envíos internacionales	Varios	150	150
Pruebas y ajustes	Consumo eléctrico durante pruebas (20 horas)	1 lote	75	75
Imprevistos (10%)	Reserva para componentes adicionales	10%	658.9	658.9
SUBTOTAL COSTOS ADICIONALES				**C$ 2,033.90**

5.23.5. Resumen general de gastos

El resumen general condensa los subtotales previos para determinar la inversión total requerida para el proyecto.

Tabla 11. Resumen General de Gastos

CONCEPTO	**MONTO (C$)**
Subtotal Componentes Electrónicos	C$ 3,532.00
Subtotal Materiales Estructurales	C$ 1,682.00
Subtotal Costos Adicionales	C$ 2,033.90
TOTAL, SIN HERRAMIENTAS	**C$ 7,247.90**
Subtotal Herramientas (opcional - referencia)	*C$ 2,775.00*
TOTAL, CON HERRAMIENTAS (referencia)	***C$ 10,022.90***
Equivalente en USD (tipo de cambio C$36.50/USD)	≈ USD $198.57

5.23.6. Análisis Comparativo

Desempeño del prototipo propuesto frente a alternativas artesanales y comerciales importadas.

Tabla 12. Análisis Comparativo

Concepto	**Incubadora Artesanal**	**Incubadora IoT (Este Proyecto)**	**Incubadora Comercial Importada**
Capacidad	20-30 huevos	30 huevos	30-50 huevos
Sistema de Control	Manual/Termostato básico	PID + IoT (ESP32)	Digital básico
Monitoreo	Local presencial	Remoto WiFi/MQTT	Local/Bluetooth limitado
Costo (C$)	C$ 1,500 - 2,500	C$ 7,248	C$ 8,000 - 15,000
Precisión Temperatura	±2°C	±0.5°C	±1°C
Volteo Automático	Manual o timer simple	Servo controlado (cada 4h)	Motor con timer
Control de Humedad	Manual	Sensor DHT22 (±3% HR)	Sensor básico
Soporte Técnico	No disponible	Sí (local, código abierto)	Limitado, importación
Escalabilidad	Baja	Alta (modular)	Media
Consumo Energético	~150W (sin control)	114W (PID eficiente)	120-200W

5.23.7. Análisis de Viabilidad Ecónomica

La tabla de análisis de viabilidad económica proyecta la rentabilidad real del sistema basándose en una tasa de eclosión esperada del 87%. Los resultados financieros indican que, con un precio promedio de pollito de C$ 70.00, la inversión inicial se recupera en un periodo de 3.5 meses (5 ciclos), alcanzando un retorno de inversión (ROI) anual estimado del 119.7

Tabla 13. Análisis de Viabilidad Ecónomica

CONCEPTO	**VALOR**
Inversión Inicial Total	C$ 7,247.90
Costo por Huevo de Inversión	C$ 241.60
COSTOS OPERACIONALES POR CICLO (21 días)	
Consumo Energético (114W × 24h × 21 días)	57.46 kWh
Tarifa Eléctrica Promedio Nicaragua	C$ 6.50/kWh
Costo Total Energía por Ciclo	C$ 373.46
PROYECCIÓN DE INGRESOS POR CICLO	
Precio Promedio Pollito en Mercado	C$ 70.00
Tasa de Eclosión Esperada (Control PID)	87%
Pollitos Esperados por Ciclo	26 unidades
Ingreso Bruto por Ciclo	C$ 1,820.00
Costo Energía por Ciclo	- C$ 373.46
Ingreso Neto por Ciclo	**C$ 1,446.54**
ANÁLISIS DE RECUPERACIÓN	
Punto de Equilibrio	5.0 ciclos
Tiempo de Recuperación	3.5 meses (≈0 año)
ROI por Ciclo	19.96%
ROI Anual Estimado (6 ciclos/año)	**119.7%**

5.23.8. Recomendaciones de compra y proveedores

Recuperación de inversión: El análisis de viabilidad indica que la inversión inicial se recupera en aproximadamente 5 ciclos de incubación (3.5 meses), considerando una tasa de eclosión del 87% alcanzable con control PID de precisión, y un precio promedio de pollito de C$ 70 en el mercado nicaragüense.

Costo operacional reducido: El consumo energético de 114W con control PID eficiente representa solo C$ 383.46 por ciclo de 21 días, significativamente menor que sistemas sin control automatizado.

5.23.9. Sostenibilidad del proyecto

La estructura de costos presenta equilibrio entre componentes locales (C$ 1,682 en materiales estructurales) y electrónicos especializados (C$ 3,532), permitiendo estrategias de compra mixtas que optimizan tiempos de entrega y presupuesto. El costo adicional de C$ 2,033.90 para mano de obra, contingencias y pruebas garantiza un margen de seguridad del 10% sobre el subtotal material.

Este proyecto representa una inversión razonable con alto potencial de retorno económico, superioridad técnica respecto a alternativas convencionales, y contribución significativa al desarrollo tecnológico y productivo de la comunidad Mario Corea. La combinación de tecnología accesible, costos competitivos y soporte local consolida este proyecto como solución viable y sostenible para la producción avícola automatizada en Nicaragua.

6. Conclusiones

El presente trabajo monografico permitió concluir que el diseño de un prototipo de incubadora avícola automatizada basada en IoT constituye una alternativa técnicamente viable para contribuir a la optimización productiva en la comunidad Mario Corea, Chinandega. El diagnóstico realizado permitió identificar que las variables críticas del proceso de incubación son la temperatura, la humedad relativa y el volteo, así como la necesidad de contar con un sistema de monitoreo y control más preciso que los métodos tradicionales empleados en entornos rurales

Asimismo, se determinó que la arquitectura propuesta de hardware y software responde adecuadamente a los requerimientos del sistema, mediante la selección del microcontrolador ESP32, sensores digitales, actuadores y protocolos de comunicación como MQTT/Wi-Fi, lo que permite la supervisión remota de las variables críticas a través de una plataforma IoT. Esta integración tecnológica fortalece la automatización del proceso y facilita la toma de decisiones basada en datos en tiempo real.

La modelación conceptual del sistema, estructurada bajo el esquema entradas–procesos–salidas, facilitó la comprensión integral de la interacción entre sensores, controlador y actuadores, sentando las bases para la simulación del comportamiento de la incubadora. Se concluye que la implementación del algoritmo PID proporciona una base adecuada para la regulación de temperatura y humedad dentro de la incubadora. La validación realizada en MATLAB confirmó un comportamiento estable del subsistema térmico, con seguimiento del setpoint y capacidad de rechazo ante perturbaciones, mientras que el subsistema de humedad fue sustentado mediante modelado matemático, cálculo de sus parámetros característicos y validación analítica de la sintonización del controlador.

Asimismo, la investigación evidenció que el diseño propuesto se encuentra alineado con los lineamientos del Plan Nacional de Lucha contra la Pobreza y para el

Desarrollo Humano 2022–2026, específicamente en lo referente al fortalecimiento de la producción en el campo, la generación y transferencia de tecnologías aplicadas a la producción, y el impulso a la innovación tecnológica como herramienta para mejorar la calidad de vida de las familias productoras. La incubadora automatizada basada en IoT se proyecta, por tanto, como una alternativa tecnológica viable para contribuir a la seguridad alimentaria, la autosuficiencia productiva y la reducción de la dependencia de insumos externos.

Finalmente, se concluye que la propuesta desarrollada representa una base sólida para la futura implementación física del prototipo, al integrar diagnóstico del contexto, diseño tecnológico, modelado y validación computacional. En este sentido, la incubadora automatizada se proyecta como una solución funcional y adaptable para fortalecer la producción avícola familiar, favorecer la seguridad alimentaria e impulsar la incorporación de tecnología en la comunidad Mario Corea.

7. Recomendaciones

Con base en los resultados obtenidos, se recomienda como primera etapa la construcción física y validación experimental del prototipo, a fin de comparar el comportamiento real del sistema con los resultados obtenidos en la simulación. Esta fase permitirá verificar variables clave como la tasa de eclosión, el consumo energético, la precisión de los sensores y la estabilidad del control durante ciclos completos de incubación.

Asimismo, se sugiere realizar una sintonización fina del controlador PID durante la implementación práctica, considerando factores reales como la inercia térmica de la estructura, las variaciones de la temperatura ambiente y la respuesta de los actuadores. Esto permitirá mejorar la estabilidad del sistema y optimizar su desempeño en condiciones de operación reales.

En cuanto a la conectividad, se recomienda evaluar alternativas complementarias al Wi-Fi, como LoRaWAN o esquemas híbridos, especialmente en zonas rurales donde el acceso a internet puede ser inestable. De igual forma, sería conveniente incorporar mecanismos de almacenamiento local y sincronización posterior de datos, para garantizar la continuidad del monitoreo aun cuando existan interrupciones temporales de conexión.

Finalmente, se recomienda que este tipo de propuestas sea tomado en cuenta por instituciones académicas y organismos vinculados al desarrollo rural, como una base para futuros procesos de transferencia tecnológica e innovación local, orientados a fortalecer la producción avícola familiar mediante soluciones accesibles, sostenibles y adaptadas al contexto.

8. Bibliografias

- Åström, K. J., & Murray, R. M. (2008). Sistemas de retroalimentación: una introducción para científicos e ingenieros. Princeton University Press.

- Barr, M., & Massa, A. (2006). Programación de sistemas embebidos con C y herramientas de desarrollo GNU. O'Reilly Media. Industrial Internet Consortium (IIC).

- Bolton, W. (2015). Controladores lógicos programables (PLC) (4.ª ed.). Newnes. [Recuperado de enlace externo].

- FAO. (2014). Producción avícola a pequeña escala: buenas prácticas para la avicultura familiar. Organización de las Naciones Unidas para la Alimentación y la Agricultura.

- FAO. (2020). Tecnologías digitales para la transformación de la agricultura. Organización de las Naciones Unidas para la Alimentación y la Agricultura. Recuperado de: https://openknowledge.fao.org/server/api/core/bitstreams/def3baab-d313-4158-9741-deb4418ad3ce/content

- Few, S. (2013). Diseño de tableros de información: visualización de datos para el monitoreo de un vistazo (2.ª ed.). Analytics Press.

- Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. International Advanced Research Journal in Science, Engineering and Technology, 5(1), 41–44.

- International Electrotechnical Commission (IEC). (2013). IEC 61131-3: Lenguajes de programación y arquitectura de software para controladores programables (Norma internacional). IEC / PLCopen.

- Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). Revisión sobre diagnóstico y pronóstico de maquinaria mediante mantenimiento basado en la condición. Mechanical Systems and Signal Processing, 20(7), 1483–1510.

- Lashari, M. H., et al. (2023). Sistema IoT para monitoreo ambiental en incubadoras avícolas. Sensors. Recuperado de: http://dspace.unach.edu.ec/bitstream/51000/11103/1/Obando%20M%2C%20Jonathan%20F.%20%282023%29

- Locke, J. (2025, 11 de abril). IoT en la agricultura. Digi International. Recuperado de: https://es.digi.com/blog/post/iot-in-agriculture

- Long, D., Bain, J., et al. (2011). Introducción a la ingeniería de sistemas basada en modelos (MBSE). MIT / Industry Primer.

- Object Management Group (OMG). (2019). SysML versión 1.6 / SysPhS: Extensión de SysML para interacción física y flujo de señales. Object Management Group. Recuperado de: https://www.omg.org/spec/SysML/1.6/About-SysML

- Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M.-J. (2017). Big data en la agricultura inteligente: una revisión. Agricultural Systems, 153, 69–80. Recuperado de: https://www.sciencedirect.com

9. Anexos

9.1. Especificaciones Técnicas

Dependencias y Librerias del Sistema

Librería	Versión/Tipo	Función
WiFi.h	ESP32 Core	Conectividad WiFi
PubSubClient.h	2.8+	Protocolo MQTT
DHTesp.h	1.18+	Sensor DHT22
Wire.h	Arduino Core	Comunicación I2C
Adafruit_GFX.h	1.11+	Gráficos OLED
Adafruit_SSD1306.h	2.5+	Display OLED
ESP32Servo.h	0.13+	Control Servomotor
WebServer.h	ESP32 Core	Servidor Web (opcional)

Configuracion de los pines ESP32

Pin GPIO	Componente	Tipo
GPIO 15	Sensor DHT22	Digital Input
GPIO 14	OLED SDA	I2C Data
GPIO 12	OLED SCL	I2C Clock
GPIO 26	Relé Calefacción	Digital Output
GPIO 25	Relé Humedad	Digital Output
GPIO 27	Buzzer	PWM Output
GPIO 21	LED Temperatura	Digital Output
GPIO 5	LED Humedad	Digital Output
GPIO 19	Servomotor	PWM Output
GPIO 2	LED WiFi	Digital Output

Parámetros PID Estandarizados

Controlador	Kp	Ki	Kd
Temperatura	3.0	0.08	2.0
Humedad	2.0	0.05	1.0

Recomendaciones de compras con proveedores

CATEGORÍA	PROVEEDORES RECOMENDADOS	OBSERVACIONES
Componentes Electrónicos Locales	Electrónica Japonesa (Managua) Comtech Mercado Oriental - Sección Electrónica	ESP32, sensores DHT22, displays OLED, relés Tiempos de entrega: 1-3 días Precios: Medios-altos
Componentes Especializados	AliExpress (China) Amazon (USA)	Servos MG995, drivers L298N, módulos buck Tiempo de envío: 3-6 semanas Precios: Económicos Requiere planificación anticipada
Materiales de Construcción	EPA (múltiples sucursales) Sinsa La Unión	Madera contrachapada, pino, tornillería Vidrio templado Disponibilidad inmediata Precios competitivos
Herramientas	Freund Sinsa EPA	Multímetros, soldadores, herramientas manuales Alquiler de herramientas disponible Garantía en herramientas eléctricas
Fuentes de Alimentación	Casas de repuestos electrónicos Proveedores industriales	SMPS 12V/15A/180W Buscar certificación UL/CE Verificar voltaje 110V AC

Guía de componentes e implementación fisica

Componente	**Función en el Código**	**Implementación en Físico**
ESP32	Cerebro, PID y WiFi.	Se monta en una placa de pruebas o PCB ventilada.
DHT22	Lectura de 37.8 ℃-39℃	Debe ir al centro de la incubadora, lejos de la resistencia de calor.
Relés de estado sólido o Optoacoplador	Switch del calentador y ventilador	Controla una resistencia blindada o bombillas incandescentes y Ventilador

Relé Humedad	Switch del humidificador.	Controla un nebulizador ultrasónico o un ventilador pequeño sobre agua.
Servo MG996R	Volteo automático.	Conectado a una bandeja oscilante que mueve los huevos 45°.
Buzzer	Alarmas sonoras.	Emite pitidos si el panel solar falla o hay calor extremo.

Densidades Presentadas en el Ambiente.

Substancia	Densidad (kg/m3)	Substancia	Densidad (kg/m3)
Aceite	0.92	**Grafito**	1.900-2.200
Aceite de oliva	0.92	**Grafito**	2.510-3.050
Aceros	6.920-9.130	**Helio**	0,18
Acetona	0.79	**Hielo de agua**	0.92
Agua (20° C)	0.998	**Hierro**	7.87
Agua (4° C)	1	**Hormigón**	2.4
Agua de mar	1.027	**Iridio**	22.5
Aire (37.8 ° C)	1.136	**Latón**	8.6
Alabastro	2.300-2.800	**Leche**	1.03
Alcohol etílico	0.78	**Litio**	0.534
Aleaciones de cobre	7.470-8.940	**Luna**	3.34
Aleaciones de titanio	4.430-4.700	**Madera (balsa)**	0.12
Aluminio	2.7	**Madera (cedro)**	0.57
Arena fina seca	1.400-1.650	**Madera (Cerezo)**	0.760-0.840
Basalto	2.700-3.200	**Madera (ébano)**	1.26
Benceno	0.9	**Madera (Encina)**	0.690-1.030
Caliza	2.460-2.840	**Madera (Haya)**	0.660-0.830
Carbono	2.26	**Madera (Manzano)**	0.660-0.840
Caucho	0.95	**Madera (Nogal)**	600-810

Cemento	0.820-1.950	**Madera (Olmo)**	0.560-0.820
Cerámicos	2.300-5.500	**Madera (pino)**	0.310-0.760
Cinc	7.14	**Madera (Roble)**	0.710-1.070
Cloroformo	1.475	**Madera (varias)**	0.400-0.700
Cobalto	8.9	**Magnesio**	1.74
Cobre	8.94	**Mármol ordinario**	2.520-2.850
Cobre	8.9	**Mercurio**	13.58
Cuarzo	2.500-2.800	**Metanol**	0.79
Cuerpo humano	0.95	**Nieve compactada**	0.3
Diamante	3.515	**Níquel**	8.9
Estaño	7.31	**Oro**	19.3
Etanol de 96°	0.81	**Osmio**	2.261
Gasolina	0.680-0.700	**Plasma sanguíneo (37℃)**	1.027
Glicerina	1.261	**Plásticos**	0.900-2.000

9.2. Índice De Funciones Principales

- **`setup()`:** Inicialización del sistema, configuración de pines, WiFi, MQTT y periféricos
- **`loop()`:** Bucle principal del sistema, coordina todas las tareas
- **`PIDController::compute()`:** Cálculo del algoritmo PID para control de temperatura y humedad
- **`isHatchingWindow()`:** Determina si el sistema está en fase de eclosión (días 19-21)
- **`checkAlarms()`:** Verifica condiciones de alarma por desviaciones críticas
- **`updateBuzzer()`:** Actualiza estado del buzzer según alarmas activas
- **`updateLEDs()`:** Actualiza indicadores LED de temperatura y humedad
- **`updateOLED()`:** Refresca pantalla OLED con información del sistema
- **`getTemperatureStatus()`:** Retorna estado de temperatura (ALTA/BAJA/OK)
- **`getHumidityStatus()`:** Retorna estado de humedad (ALTA/BAJA/OK)
- **`getNextServoTime()`:** Calcula tiempo restante para próximo volteo
- **`getCurrentServoAngle()`:** Obtiene ángulo actual del servomotor
- **`imprimirTelemetriaCompleta()`:** Genera reporte detallado de telemetría por puerto serial
- **`reconnectHiveMQ()`:** Gestiona reconexión automática a broker HiveMQ
- **`reconnectAIO()`:** Gestiona reconexión automática a Adafruit IO
- **`callbackHiveMQ()`:** Procesa mensajes MQTT recibidos desde HiveMQ
- **`callbackAIO()`:** Procesa mensajes MQTT recibidos desde Adafruit IO
- **`publishHiveMQ()`:** Publica telemetría a broker HiveMQ cada 10 segundos
- **`publishAIO()`:** Publica telemetría a Adafruit IO cada 15 segundos

9.3. Código Fuente Completo Con Comentarios Técnicos

A continuación, se presenta el código fuente completo del firmware de la incubadora, con comentarios técnicos detallados que explican la lógica de implementación, algoritmos de control y consideraciones de diseño. El código está organizado en secciones claramente identificadas para facilitar su comprensión y mantenimiento.

```
/* PROYECTO DE TESIS ASLINGNEY,LUISALVARADO,JUSTINVALLEJOS.
  INCUBADORA Carry FUSION - DUAL MQTT COMPACT
  Configuracion dual MQTT: HiveMQ + Adafruit IO
  Control PID temperatura y humedad
  Servo volteo con bloqueo en eclosion
  Buzzer, LEDs, OLED, Reles
  Servidor web comentado (descomentar para hardware)
*/

#include <WiFi.h>

/* COMENTARIO TÉCNICO:
  Las librerías incluidas proporcionan las interfaces necesarias para:
  - WiFi.h: Gestión de conectividad inalámbrica con auto-reconexión
  - PubSubClient.h: Implementación del protocolo MQTT para IoT
  - DHTesp.h: Lectura del sensor DHT22 con manejo de errores
  - Wire.h: Protocolo I2C para comunicación con display OLED
  - Adafruit_GFX/SSD1306: Renderizado gráfico en pantalla 128x64
  - ESP32Servo.h: Control PWM del servomotor con API avanzada
*/

#include <PubSubClient.h>
#include <DHTesp.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP32Servo.h>
// #include <WebServer.h>  // Descomentar para hardware

// Pines
#define DHT_PIN 15
#define DHT_TYPE DHT22
#define OLED_SDA 14
#define OLED_SCL 12
#define OLED_WIDTH 128
#define OLED_HEIGHT 64
#define OLED_ADDR 0x3C
#define PIN_RELE_CALOR 26
#define PIN_RELE_HUMEDAD 27
#define PIN_BUZZER 25
#define PIN_LED_TEMP 21
#define PIN_LED_HUM 5
#define PIN_SERVO 19
#define PIN_LED_WIFI 2

// WiFi

/* COMENTARIO TÉCNICO - CONFIGURACIÓN DE PINES:
```

```
  Se utilizan pines GPIO específicos del ESP32 considerando:
  - Pines con capacidad PWM (25, 19) para buzzer y servo
  - Pines I2C dedicados (14, 12) para comunicación con OLED
  - Pines digitales estándar para relés y LEDs
  - GPIO 2 incluye LED integrado en ESP32 para indicador WiFi
  Esta distribución evita conflictos con periféricos internos del ESP32
*/

      const char* ssid = "Wokwi-GUEST";
      const char* password = "";

      // HiveMQ
 40   const char* hivemq_server = "broker.hivemq.com";
      const int hivemq_port = 1883;
      const char* hivemq_client = "ESP32_Incubadora_Carry_001";

      // Adafruit IO
 45   const char* aio_server = "io.adafruit.com";
      const int aio_port = 1883;
      const char* aio_username = "competitivefer50";
      const char* aio_key = "aio_HjvR7438LVUoFRQ6ayMUL1bEh0kX";
      const char* aio_client = "ESP32_Incubadora_AIO";
 50
      // Objetos
      WiFiClient espClient1, espClient2;

/* COMENTARIO TÉCNICO - ARQUITECTURA MQTT DUAL:
  Sistema dual-broker para mayor confiabilidad:
  - HiveMQ: Broker público para desarrollo y pruebas
  - Adafruit IO: Plataforma IoT con dashboards integrados
  Se utilizan dos instancias WiFiClient independientes para evitar
  conflictos de conexión y permitir publicación simultánea
*/

      PubSubClient hivemq(espClient1);
      PubSubClient aio(espClient2);
 55   Adafruit_SSD1306 oled(OLED_WIDTH, OLED_HEIGHT, &Wire, -1);
      Servo servo;
      // WebServer server(80);  // Descomentar para hardware

      // Variables PID
 60   struct PIDController {
        float Kp, Ki, Kd;
        float integral, lastError;
        float outMin, outMax;
        PIDController(float p, float i, float d, float min, float max)
 65       : Kp(p), Ki(i), Kd(d), integral(0), lastError(0), outMin(min),
outMax(max) {}

        float compute(float setpoint, float input, float dt) {
          float error = setpoint - input;

/* COMENTARIO TÉCNICO - ALGORITMO PID:
  Implementación de controlador PID digital:
  - Término Proporcional (Kp): Respuesta inmediata al error actual
  - Término Integral (Ki): Elimina error en estado estacionario
  - Término Derivativo (Kd): Anticipa cambios futuros
  La función compute() calcula: salida = Kp*e + Ki*∫e*dt + Kd*de/dt
  Incluye anti-windup para evitar saturación del término integral
*/
```

```
        integral += error * dt;
        integral = constrain(integral, outMin/Ki, outMax/Ki);
        float derivative = (error - lastError) / dt;
        lastError = error;
        float output = Kp*error + Ki*integral + Kd*derivative;
        return constrain(output, outMin, outMax);
    }
};


pidTemp(0.1791, 0.0181, 0.4419, 0, 100);
PIDController pidHum(0.8152, 0.0516, 3.2195, 0, 100) ;

// Variables globales
DHTesp dht;
float temp = 0, hum = 0;
float setTemp = 37.8, setHum = 55.0;

/* COMENTARIO TÉCNICO - PARÁMETROS PID OPTIMIZADOS:
  - Kp elevado para respuesta rápida a cambios
  - Ki bajo para evitar oscilaciones
  - Kd moderado para suavizar transiciones
  - Parámetros más conservadores debido a inercia del sistema
*/

int dayIncubation = 1;
bool manualControl = false;
bool alarmActive = false;
unsigned long lastPID = 0, lastServo = 0, lastHiveMQ = 0, lastAIO = 0;
unsigned long lastBuzzer = 0, lastOLED = 0, lastTelemetry = 0,
windowStartTime = 0;
bool servoPos = false;
int fanPWM = 0;
float potenciaCalor = 0;
const unsigned long windowSize = 2000;

/* COMENTARIO TÉCNICO - GESTIÓN TEMPORAL:
  Variables unsigned long almacenan timestamps en milisegundos
  Permite programación no bloqueante mediante polling:
  - lastPID: Control de frecuencia de cálculo PID (1 Hz)
  - lastServo: Intervalo de volteo de huevos (30 segundos)
  - lastHiveMQ/lastAIO: Frecuencia de publicación MQTT (10s/15s)
  - windowStartTime: Base para PWM por ventana de tiempo (2000 ms)
*/


// Funciones auxiliares
bool isHatchingWindow() {
  return (dayIncubation >= 19 && dayIncubation <= 21);
}

void checkAlarms() {
  alarmActive = (temp < setTemp - 2 || temp > setTemp + 2 ||
                 hum < setHum - 10 || hum > setHum + 10);
}

/* COMENTARIO TÉCNICO - FUNCIÓN isHatchingWindow():
  Determina fase de eclosión (días 19-21) para:
  - Bloquear servomotor en posición central (90°)
  - Evitar daño a polluelos durante nacimiento
  - Ajustar parámetros de ventilación si es necesario
```

```
  Implementa lógica crítica de seguridad del proceso
*/


 105  void updateBuzzer() {
        unsigned long now = millis();
        if(alarmActive) {
          if(now - lastBuzzer >= 500) {

/* COMENTARIO TÉCNICO - SISTEMA DE ALARMAS:
  Umbrales de alarma basados en literatura avícola:
  - Temperatura: ±2°C respecto a setpoint (37.5°C)
  - Humedad: ±10% respecto a setpoint (60%)
  Desviaciones mayores activan buzzer intermitente (500ms)
  y señalización visual mediante LEDs indicadores
*/

            lastBuzzer = now;
 110        static bool buzzerState = false;
            buzzerState = !buzzerState;
            ledcWrite(PIN_BUZZER, buzzerState ? 128 : 0);
          }
        } else {
 115      ledcWrite(PIN_BUZZER, 0);
        }
      }

      void updateLEDs() {
 120    digitalWrite(PIN_LED_TEMP, (temp < setTemp - 1 || temp > setTemp + 1) ?
HIGH : LOW);
        digitalWrite(PIN_LED_HUM, (hum < setHum - 5 || hum > setHum + 5) ? HIGH :
LOW);
      }

      void updateOLED() {
 125    unsigned long now = millis();
        if(now - lastOLED >= 1000) {
          lastOLED = now;
          oled.clearDisplay();
          oled.setTextSize(1);
 130      oled.setTextColor(SSD1306_WHITE);

          oled.setCursor(0, 0);
          oled.print("INCUBADORA CARRY");

 135      oled.setCursor(0, 8);
          oled.print("Dia:");
          oled.print(dayIncubation);
          oled.print(" ");
          oled.print(isHatchingWindow() ? "ECLOSION" : "INCUBA");
 140
          oled.setCursor(0, 16);
          oled.print("T:");
          oled.print(temp, 1);
          oled.print("(");
 145      oled.print(setTemp, 1);
          oled.print(")[");
          oled.print(getTemperatureStatus());
          oled.print("]");

 150      oled.setCursor(0, 24);
          oled.print("H:");
```

```
        oled.print(hum, 1);
        oled.print("%(");
        oled.print(setHum, 1);
155     oled.print(")[");
        oled.print(getHumidityStatus());
        oled.print("]");

        oled.setCursor(0, 32);
160     float heatPower = potenciaCalor;
        oled.print("H:");
        oled.print((int)heatPower);
        oled.print("% F:");
        oled.print(fanPWM);
165     oled.print(" V:");
        oled.print(digitalRead(PIN_RELE_HUMEDAD) ? "ON" : "OFF");

        oled.setCursor(0, 40);
        oled.print("Servo:");
170     oled.print(getCurrentServoAngle());
        oled.print(" Volteo:");
        oled.print(getNextServoTime());
        oled.print("s");

175     oled.setCursor(0, 48);
        oled.print("HM:");
        oled.print(hivemq.connected() ? "OK" : "ERR");
        oled.print(" AIO:");
        oled.print(aio.connected() ? "OK" : "ERR");
180
        oled.setCursor(0, 56);
        oled.print(alarmActive ? "ALARMA!" : "Sistema OK");

        oled.display();
185   }
    }

    String getTemperatureStatus() {
      float diff = temp - setTemp;
190   if (diff > 1.0) return "ALTA";
      if (diff < -1.0) return "BAJA";
      return "OK";
    }

195 String getHumidityStatus() {
      float diff = hum - setHum;
      if (diff > 3.0) return "ALTA";
      if (diff < -3.0) return "BAJA";
      return "OK";
200 }

    int getNextServoTime() {
      unsigned long elapsed = millis() - lastServo;
```

/ COMENTARIO TÉCNICO - TELEMETRÍA COMPLETA:*
Sistema de reporte estructurado que incluye:
- Timestamp formateado (HH:MM:SS desde inicio)
- Estados de sensores con valores actuales y setpoints
- Estado de actuadores (potencia, PWM, relés)
- Diagnóstico de conexiones MQTT
- Detección y reporte de alarmas activas

```
  Facilita debugging y análisis del comportamiento del sistema
*/

        unsigned long remaining = (elapsed < 30000) ? (30000 - elapsed) : 0;
205     return remaining / 1000;
      }

      int getCurrentServoAngle() {
        if (isHatchingWindow()) return 90;
210     return servoPos ? 135 : 45;
      }

      void imprimirTelemetriaCompleta() {
        unsigned long totalSeconds = millis() / 1000;
215     int hours = (totalSeconds / 3600) % 24;
        int minutes = (totalSeconds / 60) % 60;
        int seconds = totalSeconds % 60;

        Serial.println();
220
Serial.println("================================================================");
        Serial.print("[");
        if (hours < 10) Serial.print("0");
        Serial.print(hours);
        Serial.print(":");
225     if (minutes < 10) Serial.print("0");
        Serial.print(minutes);
        Serial.print(":");
        if (seconds < 10) Serial.print("0");
        Serial.print(seconds);
230     Serial.println("] TELEMETRIA SISTEMA INCUBADORA CARRY");

Serial.println("================================================================");

        Serial.print("DIA: ");
        Serial.print(dayIncubation);
235     Serial.print(" | FASE: ");
        Serial.println(isHatchingWindow() ? "Eclosion (Dias 19-21)" : "Incubacion
(Dias 1-18)");

        Serial.print("TEMPERATURA: ");
        Serial.print(temp, 1);
240     Serial.print("C (SP:");
        Serial.print(setTemp, 1);
        Serial.print("C) [");
        Serial.print(getTemperatureStatus());
        float tempDiff = temp - setTemp;
245     if (tempDiff != 0) {
          Serial.print(" ");
          if (tempDiff > 0) Serial.print("+");
          Serial.print(tempDiff, 1);
          Serial.print("C");
250     }
        Serial.println("]");

        Serial.print("HUMEDAD: ");
        Serial.print(hum, 1);
255     Serial.print("% (SP:");
        Serial.print(setHum, 1);
        Serial.print("%) [");
        Serial.print(getHumidityStatus());
        float humDiff = hum - setHum;
260     if (humDiff != 0) {
```

```
        Serial.print(" ");
        if (humDiff > 0) Serial.print("+");
        Serial.print(humDiff, 1);
        Serial.print("%");
    }
    Serial.println("]");

    unsigned long windowTime = millis() - windowStartTime;
    float heatPower = (potenciaCalor / 100.0) * 100.0;
    bool heatRelayOn = digitalRead(PIN_RELE_CALOR);
    Serial.print("POTENCIA CALOR: ");
    Serial.print((int)heatPower);
    Serial.print("% | RELE: ");
    Serial.println(heatRelayOn ? "ON" : "OFF");

    Serial.print("VENTILADOR PWM: ");
    Serial.print(fanPWM);
    Serial.print(" (");
    Serial.print((fanPWM * 100) / 255);
    Serial.println("%)");

    bool humidityValveOn = digitalRead(PIN_RELE_HUMEDAD);
    Serial.print("VALVULA: ");
    Serial.println(humidityValveOn ? "ON" : "OFF");

    Serial.print("SERVO: ");
    Serial.print(getCurrentServoAngle());
    Serial.print(" grados | Proximo volteo: ");
    int nextTurnSeconds = getNextServoTime();
    Serial.print(nextTurnSeconds);
    Serial.println("s");

    Serial.print("MQTT HiveMQ: ");
    if (hivemq.connected()) {
      unsigned long lastHiveMQSeconds = (millis() - lastHiveMQ) / 1000;
      Serial.print("Conectado (ultimo: hace ");
      Serial.print(lastHiveMQSeconds);
      Serial.println("s)");
    } else {
      Serial.println("Desconectado");
    }

    Serial.print("MQTT Adafruit IO: ");

/* COMENTARIO TÉCNICO - CALLBACKS MQTT:
   Funciones de callback para procesamiento de mensajes:
   - Conversión de payload binario a String
   - Parsing de tópicos mediante endsWith()
   - Actualización de variables globales (setTemp, setHum, dayIncubation)
   Permite control remoto bidireccional del sistema
*/

    if (aio.connected()) {
      unsigned long lastAIOSeconds = (millis() - lastAIO) / 1000;
      Serial.print("Conectado (ultimo: hace ");
      Serial.print(lastAIOSeconds);
      Serial.println("s)");
    } else {
      Serial.println("Desconectado");
    }

    Serial.print("ALARMAS: ");
```

```
        bool hasAlarms = false;
315     if (temp > setTemp + 2.0) {
          Serial.print("Temperatura alta detectada ");
          hasAlarms = true;
        }
        if (temp < setTemp - 2.0) {
320       Serial.print("Temperatura baja detectada ");
          hasAlarms = true;
        }
        if (hum < setHum - 5.0) {
          Serial.print("Humedad baja detectada ");
325       hasAlarms = true;
        }
        if (!hasAlarms) {
          Serial.print("Sistema OK");
        }
330     Serial.println();


Serial.println("================================================================");
        Serial.println();
      }
335
      void reconnectHiveMQ() {
        if(!hivemq.connected()) {
          Serial.print("Conectando HiveMQ...");
          if(hivemq.connect(hivemq_client)) {
340         Serial.println("OK");
            hivemq.subscribe("incubadora/carry/settemp");
            hivemq.subscribe("incubadora/carry/sethum");
            hivemq.subscribe("incubadora/carry/setday");
          } else {
345         Serial.print("Fallo, rc=");
            Serial.println(hivemq.state());
          }
        }
      }
350
      void reconnectAIO() {
        if(!aio.connected()) {
          Serial.print("Conectando Adafruit IO...");
          if(aio.connect(aio_client, aio_username, aio_key)) {
355         Serial.println("OK");
            aio.subscribe("competitivefer50/feeds/incubadora.settemp");
            aio.subscribe("competitivefer50/feeds/incubadora.sethum");
            aio.subscribe("competitivefer50/feeds/incubadora.setday");
          } else {
360         Serial.print("Fallo, rc=");
            Serial.println(aio.state());
          }
        }
      }
365
      void callbackHiveMQ(char* topic, byte* payload, unsigned int length) {
        String msg = "";
        for(unsigned int i = 0; i < length; i++) msg += (char)payload[i];


370     Serial.print("HiveMQ [");

/* COMENTARIO TÉCNICO - PUBLICACIÓN MQTT:
   Estrategia de publicación diferenciada:
   - HiveMQ: cada 10 segundos (datos esenciales)
   - Adafruit IO: cada 15 segundos (incluye potencia calor)
   Uso de String(valor, decimales).c_str() para conversión
```

```
  Reduce carga de red manteniendo actualización frecuente
*/

        Serial.print(topic);
        Serial.print("]: ");
        Serial.println(msg);


    String topicStr = String(topic);
        if(topicStr.endsWith("settemp")) {
          setTemp = msg.toFloat();
          Serial.print("Temp ajustada a: ");
          Serial.println(setTemp);
    } else if(topicStr.endsWith("sethum")) {
          setHum = msg.toFloat();
          Serial.print("Humedad ajustada a: ");
          Serial.println(setHum);
        } else if(topicStr.endsWith("setday")) {
      dayIncubation = msg.toInt();
          Serial.print("Dia ajustado a: ");
          Serial.println(dayIncubation);
        }
      }


/* COMENTARIO TÉCNICO - FUNCIÓN setup():
  Secuencia crítica de inicialización:
  1. Configuración de pines (OUTPUT/INPUT)
  2. Inicialización de periféricos (DHT, OLED, Servo)
  3. Conexión WiFi con timeout (20 intentos)
  4. Configuración de brokers MQTT
  Estados iniciales seguros: relés apagados, servo centrado
*/

      void callbackAIO(char* topic, byte* payload, unsigned int length) {
        String msg = "";
        for(unsigned int i = 0; i < length; i++) msg += (char)payload[i];


    Serial.print("AIO [");
        Serial.print(topic);
        Serial.print("]: ");
        Serial.println(msg);


    String topicStr = String(topic);
        if(topicStr.endsWith("settemp")) {
          setTemp = msg.toFloat();
          Serial.print("Temp ajustada (AIO) a: ");
          Serial.println(setTemp);
    } else if(topicStr.endsWith("sethum")) {
          setHum = msg.toFloat();
          Serial.print("Humedad ajustada (AIO) a: ");
          Serial.println(setHum);
        } else if(topicStr.endsWith("setday")) {
      dayIncubation = msg.toInt();
          Serial.print("Dia ajustado (AIO) a: ");
          Serial.println(dayIncubation);
        }
      }

      void publishHiveMQ() {
        unsigned long now = millis();
        if(now - lastHiveMQ >= 10000) {
          lastHiveMQ = now;
      hivemq.publish("incubadora/carry/temp", String(temp, 2).c_str());
```

```
          hivemq.publish("incubadora/carry/hum", String(hum, 2).c_str());
          hivemq.publish("incubadora/carry/settemp", String(setTemp, 2).c_str());
          hivemq.publish("incubadora/carry/sethum", String(setHum, 2).c_str());
          hivemq.publish("incubadora/carry/day", String(dayIncubation).c_str());
          hivemq.publish("incubadora/carry/alarm", alarmActive ? "1" : "0");
          hivemq.publish("incubadora/carry/status", "online");
          Serial.println("Publicado en HiveMQ");
        }
      }

      void publishAIO() {
        unsigned long now = millis();
        if(now - lastAIO >= 15000) {
          lastAIO = now;
          aio.publish("competitivefer50/feeds/incubadora.temp", String(temp,
2).c_str());
          aio.publish("competitivefer50/feeds/incubadora.hum", String(hum,
2).c_str());
          aio.publish("competitivefer50/feeds/incubadora.day",
String(dayIncubation).c_str());
          aio.publish("competitivefer50/feeds/incubadora.alarm", alarmActive ? "1"
: "0");
          aio.publish("competitivefer50/feeds/incubadora.status", "online");
          aio.publish("competitivefer50/feeds/potencia-calor",
String((int)potenciaCalor).c_str());
          Serial.println("Publicado en Adafruit IO");
        }
      }

      void setup() {
        Serial.begin(115200);
        Serial.println("Iniciando Incubadora Carry - Fusion Dual MQTT");

        pinMode(PIN_LED_WIFI, OUTPUT);
        pinMode(PIN_RELE_CALOR, OUTPUT);
        pinMode(PIN_RELE_HUMEDAD, OUTPUT);
        pinMode(PIN_LED_TEMP, OUTPUT);
        pinMode(PIN_LED_HUM, OUTPUT);
        digitalWrite(PIN_RELE_CALOR, LOW);
        digitalWrite(PIN_RELE_HUMEDAD, LOW);
        digitalWrite(PIN_LED_TEMP, LOW);
        digitalWrite(PIN_LED_HUM, LOW);
        digitalWrite(PIN_LED_WIFI, LOW);

        ledcAttach(PIN_BUZZER, 2000, 8);
        ledcWrite(PIN_BUZZER, 0);

        dht.setup(DHT_PIN, DHTesp::DHT22);

        Wire.begin(OLED_SDA, OLED_SCL);
        if(!oled.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
          Serial.println("Error OLED");
        } else {
          Serial.println("OLED OK");
          oled.clearDisplay();

/* COMENTARIO TÉCNICO - CONTROL PID EN loop():
   Ejecución cada 1000ms para estabilidad:
   - Calcula delta de tiempo (dt) para integración/derivación
   - Obtiene salida PID normalizada (0-100)
   - Implementa PWM por ventana de tiempo (Slow PWM)
   - Controla relés mediante comparación tiempo vs potencia
```

```
  Método efectivo para cargas inductivas (resistencias, válvulas)
*/

          oled.setTextSize(1);
          oled.setTextColor(SSD1306_WHITE);
          oled.setCursor(0, 0);
          oled.print("Iniciando...");
 475      oled.display();
        }

        servo.attach(PIN_SERVO);
        servo.write(90);
 480    Serial.println("Servo OK (Advanced API - MG995)");

        WiFi.begin(ssid, password);
        Serial.print("Conectando WiFi");
        int attempts = 0;
 485
        while(WiFi.status() != WL_CONNECTED && attempts < 20) {
          delay(500);
          Serial.print(".");
          digitalWrite(PIN_LED_WIFI, !digitalRead(PIN_LED_WIFI));
 490      attempts++;
        }

        if(WiFi.status() == WL_CONNECTED) {
          Serial.println("WiFi conectado");

/* COMENTARIO TÉCNICO - CONTROL DE SERVO:
  Volteo automático cada 30 segundos:
  - Alterna entre 45° y 135° (90° de rotación)
  - Bloqueo automático en fase de eclosión (días 19-21)
  - Posición segura 90° durante eclosión
  Previene adherencia de embrión a membrana interna
*/

 495      Serial.print("IP: ");
          Serial.println(WiFi.localIP());
          digitalWrite(PIN_LED_WIFI, HIGH);
        } else {
          Serial.println("WiFi fallido");
 500      digitalWrite(PIN_LED_WIFI, LOW);
        }

        hivemq.setServer(hivemq_server, hivemq_port);
        hivemq.setCallback(callbackHiveMQ);
 505    aio.setServer(aio_server, aio_port);
        aio.setCallback(callbackAIO);

        Serial.println("Setup completo");
        windowStartTime = millis();
 510  }

      void loop() {
        unsigned long now = millis();

 515    if(WiFi.status() == WL_CONNECTED) {
          if(!hivemq.connected()) reconnectHiveMQ();
          if(!aio.connected()) reconnectAIO();
          hivemq.loop();
          aio.loop();
```

```
    }

/* COMENTARIO TÉCNICO - LECTURA DE SENSORES:
   DHT22 proporciona estructura TempAndHumidity:
   - Verificación de estado mediante getStatus()
   - Actualización solo si lectura válida (ERROR_NONE)
   - Previene propagación de valores erróneos
   Sensor tiene precisión ±0.5°C y ±2% HR
*/


        TempAndHumidity data = dht.getTempAndHumidity();
        if(dht.getStatus() == DHTesp::ERROR_NONE) {
          temp = data.temperature;
          hum = data.humidity;
        }

        if(now - lastPID >= 1000) {
          float dt = (now - lastPID) / 1000.0;
          lastPID = now;

          float outTemp = pidTemp.compute(setTemp, temp, dt);
          potenciaCalor = outTemp;

          float tempError = temp - setTemp;
          if(tempError > 1.0) {
              fanPWM = map(constrain(tempError * 50, 0, 100), 0, 100, 128, 255);
          } else {
              fanPWM = 64;
          }

          float outHum = pidHum.compute(setHum, hum, dt);

          unsigned long windowTime = now - windowStartTime;
          if(windowTime >= windowSize) windowStartTime = now;

          digitalWrite(PIN_RELE_CALOR, (outTemp > (windowTime * 100.0 /
windowSize)) ? HIGH : LOW);
          digitalWrite(PIN_RELE_HUMEDAD, (outHum > (windowTime * 100.0 /
windowSize)) ? HIGH : LOW);
        }

        if(now - lastServo >= 30000) {
          lastServo = now;
          if(!isHatchingWindow()) {
            servoPos = !servoPos;
            int angle = servoPos ? 135 : 45;
            servo.write(angle);
            Serial.print("Servo volteo: ");
            Serial.println(angle);
          } else {
            servo.write(90);
            Serial.println("Servo bloqueado (eclosion)");
          }
        }

        checkAlarms();
        updateBuzzer();
        updateLEDs();
        updateOLED();
```

```
    if(now - lastTelemetry >= 5000) {
      lastTelemetry = now;
      imprimirTelemetriaCompleta();
    }

    publishHiveMQ();
    publishAIO();
  }
```

Diagrama de flujo de Fimware

```
INICIO
  ↓
Inicialización de pines, DHT22, OLED, Servo, WiFi y MQTT
  ↓
now = millis()
  ↓
Verificación de WiFi y reconexión MQTT
  ↓
hivemq.loop() + aio.loop()
  ↓
Lectura de temperatura y humedad (DHT22)
  ↓
¿now - lastPID ≥ 1000 ms?
  ├─ Sí → Cálculo PID → control de calefacción + ventilador PWM + humidificación
  └─ No → continuar
  ↓
¿now - lastServo ≥ 30000 ms?
  ├─ Sí → volteo del servo (45° ↔ 135°) / bloqueo a 90° en eclosión
  └─ No → continuar
  ↓
Actualización de alarmas + buzzer + LEDs
  ↓
¿now - lastOLED ≥ 1000 ms?
  ├─ Sí → actualización de pantalla OLED
  └─ No → continuar
  ↓
¿now - lastTelemetry ≥ 5000 ms?
  ├─ Sí → impresión de telemetría serial
  └─ No → continuar
  ↓
¿now - lastHiveMQ ≥ 10000 ms?
  ├─ Sí → publicación en HiveMQ
  └─ No → continuar
  ↓
¿now - lastAIO ≥ 15000 ms?
  ├─ Sí → envío de temperatura, humedad, alarmas y potencia a Adafruit IO
  └─ No → continuar
  ↓
RETORNO AL LOOP PRINCIPAL
```

“

El diagrama de flujo muestra que la lógica del firmware se basa en temporización no bloqueante mediante la función millis(), la cual permite ejecutar tareas periódicas sin detener el lazo principal del programa. Gracias a este enfoque, se coordinan de forma continua la lectura de sensores, el control PID, el volteo, la gestión de alarmas y la publicación de datos en Adafruit IO, evitando el uso de delay() durante la operación normal del sistema y garantizando así una comunicación fluida y estable con la plataforma IoT”

CODIGO SIMULACIÓN MATLAB

```
clear; clc; close all; s = tf('s');

%% ========================================================= % PARAMETROS
INCUBADORA CARRY % ========================================================== %
Setpoints Tsp = 37.8; % Temperatura [°C] Hsp = 55; % Humedad [%HR]

% Condiciones ambientales base Tamb = 29; % Temperatura ambiente [°C] Hamb = 45; %
Humedad ambiente [%HR]

% Referencias en desviación rT = Tsp - Tamb; % 8.8 °C rH = Hsp - Hamb; % 10 %HR

%% ========================================================= % MODELOS FOPDT DE LA
PLANTA % ========================================================== % ----------
TEMPERATURA ---------- KT = 80; % Ganancia térmica normalizada tauT = 150; %
Constante de tiempo [s] LT = 5; % Tiempo muerto [s]

% Aproximación de Padé de primer orden [numLT, denLT] = pade(LT,1); GT =
tf(KT*numLT, conv([tauT 1], denLT));

% ---------- HUMEDAD ---------- KH = 18.3; % Ganancia de humedad normalizada tauH =
250; % Constante de tiempo [s] LH = 8; % Tiempo muerto [s]

[numLH, denLH] = pade(LH,1); GH = tf(KH*numLH, conv([tauH 1], denLH));

%% ========================================================= % CALCULO ANALITICO DE
ZIEGLER-NICHOLS % ========================================================== %
Relación de fase: % atan(tauw) + Lw = pi

% ---------- TEMPERATURA ---------- fT = @(w) atan(tauTw) + LTw - pi; wuT =
fzero(fT, 0.3); % frecuencia última [rad/s] PuT = 2pi/wuT; % periodo último [s] KuT
= sqrt(1 + (tauTwuT)^2)/KT; % ganancia última

% PID clásico ZN KpT = 0.6KuT; TiT = PuT/2; TdT = PuT/8; KiT = KpT/TiT; KdT =
KpTTdT;

% PID conservador (50%) KpT_c = 0.5KpT; KiT_c = 0.5KiT; KdT_c = 0.5*KdT;

% ---------- HUMEDAD ---------- fH = @(w) atan(tauHw) + LHw - pi; wuH = fzero(fH,
0.2); % frecuencia última [rad/s] PuH = 2pi/wuH; % periodo último [s] KuH = sqrt(1
+ (tauHwuH)^2)/KH; % ganancia última

% PID clásico ZN KpH = 0.6KuH; TiH = PuH/2; TdH = PuH/8; KiH = KpH/TiH; KdH =
KpHTdH;

% PID conservador (50%) KpH_c = 0.5KpH; KiH_c = 0.5KiH; KdH_c = 0.5*KdH;

%% ========================================================= % MOSTRAR RESULTADOS
DE CALCULO PARA AMBOS LAZOS %
==========================================================
disp('=================================================='); disp(' RESULTADOS
ANALITICOS DE ZIEGLER-NICHOLS');
disp('==================================================');
```

```
fprintf('\n--- TEMPERATURA ---\n'); fprintf('K = %.4f\n', KT); fprintf('tau = %.4f
s\n', tauT); fprintf('L = %.4f s\n', LT); fprintf('wu = %.4f rad/s\n', wuT);
fprintf('Ku = %.4f\n', KuT); fprintf('Pu = %.4f s\n', PuT);

fprintf('\nPID clásico ZN (Temperatura):\n'); fprintf('Kp = %.4f\n', KpT);
fprintf('Ki = %.4f\n', KiT); fprintf('Kd = %.4f\n', KdT);

fprintf('\nPID conservador (Temperatura):\n'); fprintf('Kp = %.4f\n', KpT_c);
fprintf('Ki = %.4f\n', KiT_c); fprintf('Kd = %.4f\n', KdT_c);

fprintf('\n--- HUMEDAD ---\n'); fprintf('K = %.4f\n', KH); fprintf('tau = %.4f
s\n', tauH); fprintf('L = %.4f s\n', LH); fprintf('wu = %.4f rad/s\n', wuH);
fprintf('Ku = %.4f\n', KuH); fprintf('Pu = %.4f s\n', PuH);

fprintf('\nPID clásico ZN (Humedad):\n'); fprintf('Kp = %.4f\n', KpH); fprintf('Ki
= %.4f\n', KiH); fprintf('Kd = %.4f\n', KdH);

fprintf('\nPID conservador (Humedad):\n'); fprintf('Kp = %.4f\n', KpH_c);
fprintf('Ki = %.4f\n', KiH_c); fprintf('Kd = %.4f\n', KdH_c);

%% ======================================================== % SIMULACION SOLO DEL
LAZO TERMICO % ======================================================== %
Controlador térmico conservador CT = pid(KpT_c, KiT_c, KdT_c);

% Lazo cerrado térmico TT = feedback(CT*GT, 1);

% Tiempo de simulación t = (0:1:2500)';

% Entrada de referencia (escalón en desviación) uT = rT * ones(size(t));

% Respuesta sin perturbación yT = lsim(TT, uT, t); Treal = Tamb + yT;

%% ========================================================= % PERTURBACION TERMICA
% ========================================================== % Perturbación
ambiental: +3 °C desde t = 1000 s dTamb = zeros(size(t)); dTamb(t >= 1000) = 3;

% Canal de perturbación ambiente -> salida GdT = tf(1, [tauT 1]);

% Sensibilidad del lazo ST = feedback(1, CT*GT);

% Respuesta a perturbación Td_cl = minreal(GdT * ST); yTd = lsim(Td_cl, dTamb, t);

% Temperatura total perturbada Treal_pert = Treal + yTd;

%% ========================================================= % METRICAS DE
DESEMPEÑO SOLO PARA TEMPERATURA %
========================================================== eT = Tsp - Treal_pert;

infoT = stepinfo(Treal, t, Tsp);

essT = abs(eT(end)); IAE_T = trapz(t, abs(eT)); ISE_T = trapz(t, eT.^2); ITAE_T =
trapz(t, t.*abs(eT));

disp(' '); disp('=================================================='); disp('
METRICAS DE DESEMPEÑO DEL LAZO TERMICO');
disp('==================================================');
```

```
fprintf('\nRiseTime = %.4f s\n', infoT.RiseTime); fprintf('SettlingTime = %.4f
s\n', infoT.SettlingTime); fprintf('Overshoot = %.4f %%\n', infoT.Overshoot);
fprintf('Error estacionario = %.4f\n', essT); fprintf('IAE = %.4f\n', IAE_T);
fprintf('ISE = %.4f\n', ISE_T); fprintf('ITAE = %.4f\n', ITAE_T);

%% ========================================================= % GRAFICAS FINALES
PARA TESIS % =========================================================
figure('Color','w','Position',[100 100 1100 850]);

% ---------- 1. Respuesta completa ---------- subplot(3,1,1) plot(t, Treal, 'b',
'LineWidth', 2); hold on; plot(t, Treal_pert, 'k', 'LineWidth', 2); yline(Tsp, '--
g', 'Setpoint', 'LineWidth', 1.5, ... 'LabelHorizontalAlignment','left', ...
'LabelVerticalAlignment','bottom'); xline(1000, ':r', 'Perturbación', 'LineWidth',
1.5, ... 'LabelOrientation','aligned', ... 'LabelVerticalAlignment','middle');

grid on; set(gca,'Color','w','XColor','k','YColor','k','FontName','Times New
Roman','FontSize',12); xlabel('Tiempo [s]','Color','k'); ylabel('Temperatura
[°C]','Color','k'); title('Respuesta completa del sistema
térmico','Color','k','FontWeight','bold'); legend('PID conservador', 'PID

 conservador + perturbación', 'Setpoint', ... 'Location', 'northeast'); xlim([0
2500]); ylim([35 43]);

% ---------- 2. Zoom del arranque ---------- subplot(3,1,2) plot(t, Treal, 'b',
'LineWidth', 2); hold on; plot(t, Treal_pert, 'k', 'LineWidth', 2); yline(Tsp, '--
g', 'Setpoint', 'LineWidth', 1.5, ... 'LabelHorizontalAlignment','left', ...
'LabelVerticalAlignment','bottom');

grid on; set(gca,'Color','w','XColor','k','YColor','k','FontName','Times New
Roman','FontSize',12); xlabel('Tiempo [s]','Color','k'); ylabel('Temperatura
[°C]','Color','k'); title('Zoom del transitorio
inicial','Color','k','FontWeight','bold'); legend('PID conservador', 'PID
conservador + perturbación', 'Setpoint', ... 'Location', 'southeast'); xlim([0
120]); ylim([35 43]);

% ---------- 3. Zoom alrededor de la perturbación ---------- subplot(3,1,3) plot(t,
Treal_pert, 'k', 'LineWidth', 2); hold on; yline(Tsp, '--g', 'Setpoint',
'LineWidth', 1.5, ... 'LabelHorizontalAlignment','left', ...
'LabelVerticalAlignment','bottom'); xline(1000, ':r', 'Perturbación', 'LineWidth',
1.5, ... 'LabelOrientation','aligned', ... 'LabelVerticalAlignment','middle');

grid on; set(gca,'Color','w','XColor','k','YColor','k','FontName','Times New
Roman','FontSize',12); xlabel('Tiempo [s]','Color','k'); ylabel('Temperatura
[°C]','Color','k'); title('Respuesta ante perturbación
térmica','Color','k','FontWeight','bold'); xlim([900 1300]); ylim([37.6 38.2]); %
<- zoom vertical más fino

exportgraphics(gcf,'respuesta_termica_tesis_final.png','Resolution',300);
```