ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Sistema de Supervisión y Control con LOGO Web Editor para un Secador Híbrido

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

Ingeniero en Electrónica y Automatización

Presentado por:

Angel Ruben Zumba Toledo

Miguel Gustavo Mayorga Torres

GUAYAQUIL - ECUADOR

Año: 2025

DEDICATORIA

Deseo dedicar el presente trabajo a mi familia, por su apoyo tanto moral como económico y sin el cual no hubiera logrado llegar a este punto de mi vida. También a mis profesores por compartir no solo sus conocimientos, sino también sus valores y ejemplo de dedicación. A todos ustedes dedico con gran gratitud y cariño este logro.

(Angel Zumba)

Quiero dedicar afectuosamente este trabajo a todas aquellas personas que directa e indirectamente contribuyeron culminar con éxito este proyecto, a los amigos que donaron materiales, a los profesores que contribuyeron con ideas y a mis familiares por su apoyo incondicional. (Miguel Mayorga)

AGRADECIMIENTOS

Quiero agradecer a las personas que hicieron posible la culminación de este proyecto. Al Sr. Pazmiño que gentilmente en más de una ocasión nos recogió en la universidad a altas horas de la noche y nos acogió en su casa para pasar la noche, agradezco su invaluable amistad. Quiero agradecer especialmente a los Ing. Xavier Ladines, Ing. Jaun Peralta, Ing. Emerita Delgado y al Ing. José Reinoso, por abrirme las puertas del CDTS de donde me llevo gratas experiencias y conocimiento práctico que apliqué durante el desarrollo de este trabajo. Finalmente quiero agradecer a mi familia, pilar fundamental de toda esta etapa. (Miguel Mayorga)

Quiero agradecer primero a Dios por darme la fuerza necesaria para superarme y culminar con éxito esta fase de mi vida. A mi familia por su constante apoyo que me ha impulsado en incontables ocasiones. A mi amigo Gabriel Pazmiño que a pesar de su cansancio se tomó la molestia en varias ocasiones de ayudarnos con transporte y con posada. Finalmente, a mis profesores, mi tutor de tesis el Ing. Holger Cevallos y mi profesor de la materia integradora el Ing. Efrén Herrera, agradezco su paciencia y colaboración que me ayudaron en gran medida a lograr este gran objetivo en mi vida. A todos gracias por su gran ayuda. (Angel Zumba)

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Angel Ruben Zumba Toledo y Miguel Gustavo Mayorga Torres y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Firmado electrónicamente por:
ANGEL RUBEN ZUMBA TOLEDO
Validar únicamente con FirmaEC

Angel Ruben Zumba Toledo

Firmado electrónicamente por:
MIGUEL GUSTAVO MAYORGA TORRES
Validar únicamente con FirmaEC

Miguel Gustavo Mayorga Torres

EVALUADORES

HOLGER IGNACIO CEVALLOS ULLOA

Firmado digitalmente por HOLGER IGNACIO CEVALLOS ULLOA
Fecha: 2025.09.09 03:27:55 -05'00'

PhD. Holger Ignacio Cevallos Ulloa

PROFESOR TUTOR

EFREN VINICIO HERRERA MUENTES

Digitally signed by EFREN VINICIO HERRERA MUENTES
Date: 2025.09.09 23:23:00 -05'00'

PhD. Efrén Vinicio Herrera Muentes

PROFESOR DE LA MATERIA

ÍNDICE GENERAL

EVALUADORES 5
ABREVIATURAS iii
CAPÍTULO 1 1
1. INTRODUCCIÓN 1
1.1 Descripción del problema 1
1.2 Justificación del problema 1
1.3 Objetivos 2
1.3.1 Objetivo General 2
1.3.2 Objetivos Específicos 2
1.4 Marco Teórico 3
CAPÍTULO 2 6
2. Metodología 6
2.1 Análisis de la estructura de secador 6
2.2 Diagrama de bloques 7
2.3 Diagrama topológico de Red 8
2.4 Entradas y salidas 9
2.5 Tabla de materiales 11
2.6 Diseño de tablero 12
2.7 Programación de la Raspberry Pi 14
2.8 Programación del PLC LOGO! 15
2.9 Consideraciones técnicas y normativas 16
CAPITULO 3 18
3. Análisis y Resultados 18
3.1 Tabla de Precios de componentes. 18
3.2 Análisis y comparación de datos de sensores dht22 y termopares externos: 19
3.3 Evolución de la temperatura del secador hibrido 22
3.4 Comparación Sensores DHT22 vs Termopares. 23
3.5 Comportamiento de la humedad relativa. 23
3.6 Visualización de datos y control de temperatura en la página web. 23
3.7 Discusión de resultados 25

3.8 Conclusiones parciales del capítulo 26
CAPÍTULO 4 28
4. CONCLUSIONES Y RECOMENDACIONES 28
4.1 Conclusiones 28
4.2 Recomendaciones 29
BIBLIOGRAFÍA 30
ANEXOS 1
Anexo 1. Fotos de proceso de construcción del Sistema de Secado Hibrido 1
Apendice A. 1
Diseños en 3D 1
Apéndice B. 3
Pasos para instalación y configuración del sistema Rasbian 3
Apéndice C. 7
Programación de PLC Logo: 7
Apéndice D. 10
Código del script de Python para comunicación modbus entre PLC Logo y servidor Raspberry pi. 10
Apéndice E 30
Diseño de tablero 30
Diagrama Unifilar 30
Apéndice F. 31
Finalización del Proyecto 31

ABREVIATURAS

ESPOL – Escuela Superior Politécnica del Litoral

CDTS – Centro de Desarrollo Tecnológico Sustentable

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

En la ciudad de Guayaquil, a la altura del kilómetro 30.5 de la vía Perimetral, dentro del Centro de Desarrollo Tecnológico Sustentable (CDTS) de la Escuela Superior Politécnica del Litoral (ESPOL), se encuentra un secador híbrido diseñado para el tratamiento de biomasa. Este equipo integra energía solar y resistencias eléctricas, con el propósito de mantener las condiciones térmicas necesarias para el secado. El sistema cuenta con una cámara de secado cubierta con una lámina de policarbonato transparente y un flujo de aire precalentado mediante un colector solar, complementado por resistencias que permiten dar estabilidad al proceso.

Actualmente, el secador presenta limitaciones operativas significativas. El registro de datos se realiza de forma manual y se transfiere posteriormente a hojas de cálculo en Excel, lo que genera pérdida de tiempo y aumenta la probabilidad de errores. Adicionalmente, el controlador de temperatura encargado de regular las resistencias se encuentra averiado, lo que dificulta su activación y ocasiona discontinuidades en el proceso. Estas deficiencias afectan la eficiencia del sistema, comprometen la calidad de los datos obtenidos y han provocado que el equipo se encuentre fuera de servicio.

La ausencia de un sistema automatizado de supervisión y control limita el uso experimental del secador, impidiendo validaciones técnicas y científicas. En consecuencia, se desaprovecha el potencial del equipo para apoyar proyectos de investigación orientados a la eficiencia energética y a la sostenibilidad de procesos agrícolas.

1.2 Justificación del problema

La automatización del secador híbrido surge como una respuesta a las limitaciones que presenta su operación manual. El registro no automatizado de variables ambientales, así como la activación directa de los actuadores, incrementa el riesgo de errores humanos y la exposición a condiciones eléctricas inseguras.

La implementación de un sistema automatizado permite garantizar la continuidad operativa y la obtención de datos confiables en tiempo real, mejorando la eficiencia energética y la calidad de los resultados experimentales. Al integrar tecnologías de control y monitoreo, se reduce la dependencia de la intervención humana, se optimiza el uso de la energía solar y se respalda el proceso con resistencias eléctricas únicamente cuando es necesario.

El secador híbrido, además, constituye una alternativa sostenible para el procesamiento de biomasa y productos agrícolas, al aprovechar fuentes renovables y minimizar la huella ambiental. La incorporación de un sistema automatizado fortalece su viabilidad como herramienta de investigación aplicada, permitiendo validar parámetros técnicos y científicos con mayor precisión.

Finalmente, este proyecto tiene un impacto formativo relevante, pues al desarrollarse en un entorno académico, contribuye al fortalecimiento de competencias en áreas como la automatización, el control de procesos, las energías renovables y el análisis de datos. De esta manera, no solo se optimiza un equipo de laboratorio, sino que también se fomenta la generación de conocimiento y el desarrollo de soluciones tecnológicas aplicables al sector agroindustrial.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema automatizado de supervisión y control para un secador híbrido, que permita registrar en tiempo real las variables de humedad y temperatura, gestionar el funcionamiento de la resistencia eléctrica de manera eficiente y disponer de un registro confiable de datos para su análisis posterior.

1.3.2 Objetivos Específicos

- Automatizar la captura de datos de humedad y temperatura mediante sensores conectados a un sistema de adquisición, con comunicación bidireccional hacia el controlador lógico programable, para visualizar la información en tiempo real en una interfaz web.

- Implementar una interfaz gráfica accesible dentro de la red local que permita al usuario supervisar el sistema y gestionar el encendido y apagado de las resistencias de manera segura y práctica.
- Diseñar un tablero de control que aloje los componentes eléctricos y electrónicos del sistema, garantizando la correcta organización, seguridad y facilidad de mantenimiento.
- Desarrollar un sistema de supervisión y control que permita el monitoreo en tiempo real de las variables ambientales y del estado del secador híbrido, incorporando además la generación de archivos en formato CSV para el registro histórico y análisis de datos.

1.4 Marco Teórico

El secado de productos agrícolas constituye una etapa crítica en la cadena de postcosecha, ya que influye directamente en la calidad, vida útil y valor comercial del producto final. En regiones rurales o con infraestructura limitada, los métodos tradicionales de secado suelen presentar desventajas significativas, entre ellas tiempos prolongados, dependencia de las condiciones climáticas y riesgo de contaminación por agentes externos como insectos, hongos o polvo. En este contexto, los sistemas híbridos de secado solar-eléctrico se presentan como una alternativa eficiente y sostenible, al combinar fuentes de energía renovable con el respaldo de resistencias eléctricas que aseguran la continuidad del proceso.

La utilización de energía solar en procesos de secado ha sido ampliamente investigada debido a su carácter renovable, bajo costo operativo y sostenibilidad ambiental. Sin embargo, su intermitencia por factores climáticos adversos ha impulsado el desarrollo de configuraciones híbridas que integran apoyos eléctricos o mecánicos, como resistencias y blowers, garantizando así estabilidad térmica y reducción en los tiempos de secado (Paes et al., 2022).

En los últimos años, la automatización ha desempeñado un rol fundamental en la optimización de sistemas de secado. Diversos autores han desarrollado soluciones que integran sensores, controladores lógicos programables (PLC), microcontroladores y plataformas de visualización para supervisar variables críticas como temperatura, humedad relativa, velocidad del aire y masa del producto. Por

ejemplo, Dharmender et al. (2024) propusieron un secador solar automatizado con asistencia de desecantes, incorporando control de temperatura mediante sensores y arquitectura basada en microcontroladores, lo que permitió mejorar la eficiencia energética y la calidad de las semillas tratadas.

De igual manera, Paes et al. (2022) desarrollaron un sistema de adquisición automática de datos en un secador híbrido, empleando sensores, comunicación inalámbrica y aplicaciones móviles para facilitar el monitoreo remoto. Sus resultados evidenciaron mayor precisión en la medición y validación frente a métodos convencionales, lo que resalta la relevancia de incorporar tecnologías de bajo costo y accesibles para el sector agrícola.

En el ámbito industrial, los PLC constituyen la solución más robusta y confiable para la automatización de procesos. Rosa et al. (2020) implementaron un sistema embebido para la supervisión de deshumidificadores industriales, integrando sensores, interfaces gráficas y comunicación mediante protocolos estándar, como Modbus RTU, lo que evidenció la importancia de la interoperabilidad en entornos distribuidos. Asimismo, investigaciones recientes exploran la incorporación de algoritmos evolutivos y técnicas de autooptimización en entornos PLC, generando código de control adaptativo y reduciendo los tiempos de desarrollo (Löppenberg & Schwung, 2023).

El uso de plataformas de visualización es otro pilar en la automatización de procesos de secado. Herramientas como el Logo Web Editor permiten crear interfaces gráficas accesibles desde una red local, facilitando tanto la supervisión en tiempo real como el control directo de los actuadores del sistema. Si bien estos entornos no alcanzan la complejidad de un sistema SCADA industrial, su implementación constituye una arquitectura de supervisión simplificada que satisface las necesidades de registro, visualización y control a escala de laboratorio. En este tipo de soluciones, la generación de archivos en formato CSV cumple la función de registro histórico, permitiendo analizar los datos en hojas de cálculo o bases de datos sin necesidad de servidores dedicados.

Finalmente, la integración de sensores de bajo costo, sistemas embebidos y controladores programables en sistemas híbridos de secado agrícola representa un

enfoque viable, económico y escalable hacia la digitalización de procesos rurales. El proyecto actual se enmarca en esta línea, proponiendo la implementación de un sistema automatizado que combina adquisición de datos, supervisión mediante interfaz web y control de resistencias eléctricas, con potencial de evolución hacia arquitecturas más avanzadas de monitoreo remoto y análisis predictivo.

CAPÍTULO 2

El presente capitulo contiene la descripción de todo el proceso seguido para la realización del presente proyecto, así como los diferentes diagramas que explican las conexiones y forma de funcionamiento de este mismo.

2. Metodología

2.1 Análisis de la estructura de secador

El secador híbrido utilizado en este proyecto se encuentra instalado en el Centro de Desarrollo Tecnológico Sustentable (CDTS) y fue previamente diseñado y construido como parte de iniciativas anteriores. Su configuración incluye una cámara de secado cubierta por policarbonato transparente, un colector solar encargado de precalentar el aire y resistencias eléctricas para complementar el aporte térmico cuando las condiciones solares son insuficientes.

Durante la etapa inicial del proyecto se evaluó la posibilidad de alimentar la resistencia eléctrica con la planta fotovoltaica existente en el sitio. Sin embargo, tras la inspección técnica se determinó que dicha planta no contaba con la capacidad suficiente para cubrir la demanda energética del sistema. En consecuencia, se decidió conectar las resistencias a la red eléctrica, garantizando su operación confiable y continua.

Con el fin de automatizar su funcionamiento y mejorar la precisión del registro de datos, se identificaron los siguientes subsistemas de trabajo:

- **Subsistema de alimentación:** El aire de secado se precalentó en el colector solar y posteriormente atravesó las resistencias eléctricas, alimentadas desde la red pública, que permitieron mantener la temperatura de operación establecida.
- **Subsistema de control:** Se empleó un controlador lógico programable (PLC) en conjunto con una unidad de procesamiento auxiliar para recibir la información de los sensores de humedad y temperatura. Con base en estos datos, se ejecutó la lógica de encendido y apagado de la resistencia de acuerdo con el valor de consigna establecido.

- **Subsistema de red y monitoreo:** La comunicación entre dispositivos se estableció mediante el protocolo Modbus TCP/IP, permitiendo la transmisión bidireccional de datos entre el PLC, la unidad de procesamiento y el router del CDTS. Gracias a esta infraestructura se desplegó una interfaz gráfica en Logo Web Editor, desde la cual fue posible supervisar en tiempo real el estado del sistema y registrar las variables en archivos de tipo CSV para su posterior análisis.

La integración de estos subsistemas permitió disponer de un equipo operativo y automatizado, reduciendo la intervención manual y garantizando un monitoreo confiable de las variables críticas del proceso de secado.

2.2 Diagrama de bloques

Con el fin de representar la organización funcional del secador híbrido y su sistema de automatización, se elaboró un diagrama de bloques que muestra la interacción entre los diferentes subsistemas y dispositivos empleados. Dicho diagrama se presenta en el **Gráfico *2.1***

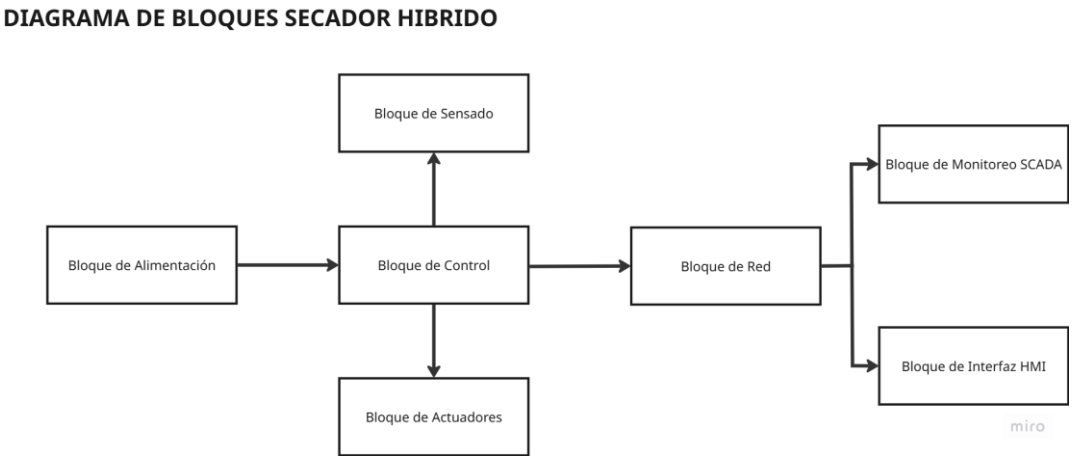


Gráfico 2.1 Diagrama de bloques del proyecto.

El sistema se estructuró en tres bloques principales:

Bloque de alimentación: El aire de secado fue precalentado mediante un colector solar y posteriormente pasó a través de resistencias eléctricas de 2 kW, alimentadas desde la red eléctrica pública. Este esquema permitió garantizar la estabilidad de la temperatura requerida, complementando el aporte solar en momentos de baja radiación.

Bloque de control: Se utilizó un controlador lógico programable (PLC) Siemens LOGO! 8.2, en conjunto con una Raspberry Pi 3 Modelo B, que actuó como unidad de procesamiento auxiliar. Los sensores de humedad y temperatura DHT22 se conectaron a la Raspberry Pi, la cual transmitió los datos al PLC mediante el protocolo Modbus TCP/IP. El PLC ejecutó la lógica de control, determinando el encendido y apagado de las resistencias en función del valor de consigna establecido.

Bloque de red y monitoreo: Para la comunicación entre los dispositivos se utilizó un switch Ethernet, que interconectó el PLC, la Raspberry Pi y el router del CDTS. Gracias a esta infraestructura, se desplegó una interfaz gráfica desarrollada en Logo Web Editor, desde la cual fue posible supervisar en tiempo real el estado del sistema y registrar automáticamente los datos obtenidos en archivos de formato CSV, accesibles para su análisis en hojas de cálculo.

La integración de estos bloques permitió implementar una solución práctica y confiable para la supervisión y control del secador híbrido, reduciendo la intervención manual y garantizando un acceso confiable a la información del proceso.

2.3 Diagrama topológico de Red

Con el fin de garantizar la comunicación entre el PLC y el servidor Raspberry, se implementó una topología de red en estrella, representada en el **Gráfico *2.2***

En esta configuración, el switch Ethernet actuó como nodo central de conexión. A través de él se interconectaron el PLC Siemens LOGO! 8.2, la Raspberry y el router del CDTS, lo que permitió establecer un canal de comunicación estable mediante el protocolo Modbus TCP/IP.

La Raspberry Pi recibió la información de los sensores DHT22 y la transmitió al PLC, mientras que este último ejecutó la lógica de control y envió los datos hacia la red. Gracias a la integración con el router, la interfaz desarrollada en Logo Web Editor estuvo disponible dentro de la red local, posibilitando el acceso al dashboard para la visualización en tiempo real del estado del sistema.

El empleo de esta topología permitió centralizar la gestión de las comunicaciones y reducir la complejidad de las conexiones entre los dispositivos, asegurando un flujo de información confiable para la supervisión y control del secador híbrido.

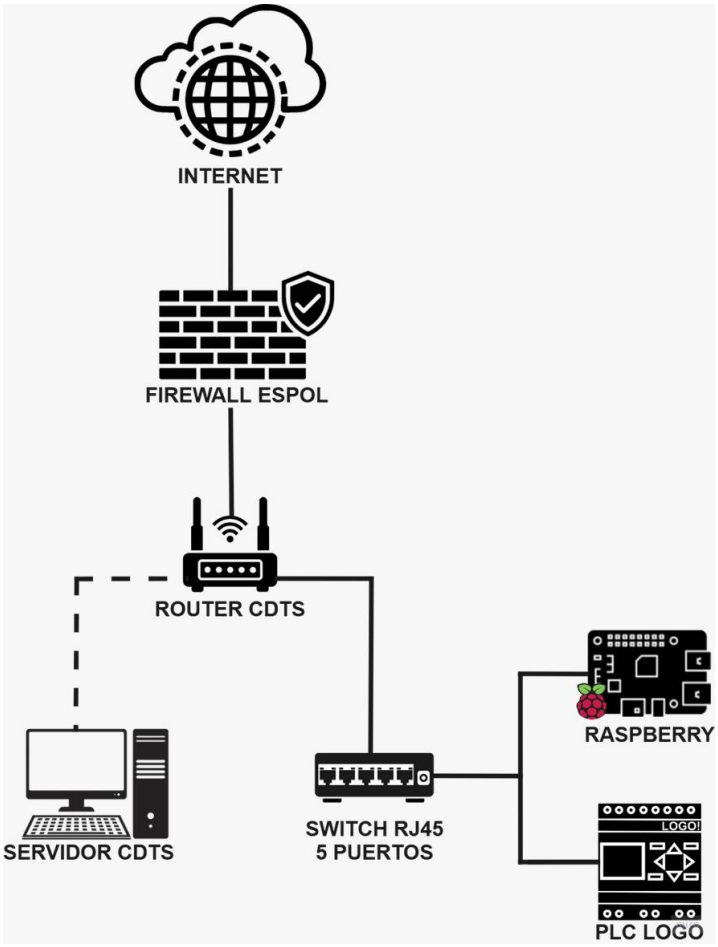


Gráfico 2.2 Topología de Red.

2.4 Entradas y salidas

Tabla 2.1Lista de entradas y salidas del sistema.

TABLA DE ENTRADAS Y SALIDAS		
Señal	Tipo	Descripción
TH1	Entrada	Sensor DHT22 ubicado en la cámara de secado (posición bandeja inferior) para medir humedad y temperatura.

TH2	Entrada	Sensores DHT22 ubicados en posiciones estratégicas dentro de la cámara del secador para medir las variables de temperatura y de humedad.
TH3	Entrada	Sensor DHT22 ubicado en la cámara de secado (posición punto muerto) para medir humedad y temperatura.
TH4	Entrada	Sensor DHT22 ubicado en la cámara de secado (posición bandeja intermedia inferior) para medir humedad y temperatura.
TH5	Entrada	Sensor DHT22 ubicado en la cámara de secado (posición bandeja intermedia superior) para medir humedad y temperatura.
TH6	Entrada	Sensor DHT22 ubicado en la cámara de secado (posición bandeja superior) para medir humedad y temperatura.
PARO DE EMERGENCIA	Entrada	Pulsador de paro de emergencia para detener el sistema en condiciones críticas.
START_RES	Entrada	Pulsador de arranque de la resistencia.
STOP_RES	Entrada	Pulsador de apagado de la resistencia.
START_BLOWER	Entrada	Pulsador de arranque del blower.
STOP_BLOWER	Entrada	Pulsador de apagado del blower.
START_AUTO	Entrada	Pulsador de arranque del modo Automático
STOP_AUTO	Entrada	Pulsador de apagado del modo Automático
AUTO_ON	Salida	Luz piloto verde que indica encendido del modo Automático.
AUTO_OFF	Salida	Luz piloto roja que indica apagado del modo Automático.
BLOWER_ON	Salida	Activación del blower

BLW_ON	Salida	Luz piloto verde que indica encendido del blower.
BLW_OFF	Salida	Luz piloto roja que indica apagado del blower.
RES_ON	Salida	Activación de la resistencia.
PLT_ RES_ON	Salida	Luz piloto verde que indica encendido de la resistencia.
PLT_ RES_OFF	Salida	Luz piloto roja que indica apagado de la resistencia.

Fuente: Elaboración propia a partir de la implementación del sistema.

2.5 Tabla de materiales

Tabla 2.2 Lista de materiales.

ÍTEM	DESCRIPCIÓN	CANTIDAD	FUNCIÓN PRINCIPAL
1	Controlador lógico programable (PLC) Siemens LOGO! 8.2	1	Ejecución de la lógica de control del sistema.
2	Raspberry Pi 3 Modelo B	1	Procesamiento de datos de sensores y comunicación con el PLC mediante Modbus TCP/IP.
3	Switch Ethernet de 5 puertos	1	Interconexión de la red local (PLC, Raspberry Pi y router CDTS).
4	Sensor DHT22	6	Medición de humedad relativa y temperatura en diferentes puntos de la cámara del secador.
5	Pulsador de arranque/parada	3	Control manual de la resistencia, blower y modo automático.
6	Pulsador de paro de emergencia	1	Detención del sistema en condiciones críticas.
7	Luz piloto verde 220 V	3	Indicación visual de encendido (resistencia, blower y modo automático).

8	Luz piloto roja 220 V	3	Indicación visual de apagado (resistencia, blower y modo automático).
9	Tablero eléctrico ABS con plafón metálico (50×40×18 cm)	1	Montaje y protección de los componentes del sistema.
10	Breaker 2P – 25 A	1	Breaker principal.
11	Seccionador 2P – 16 A	1	Seccionamiento de la resistencia.
12	Seccionador 1P – 4 A	1	Seccionamiento del blower.
13	Seccionador 1P – 4A	1	Seccionamiento del circuito de control.
14	Resistencia eléctrica 220 V / 2 kW	1	Calentamiento del aire que ingresa a la cámara de secado.
15	Blower 282 W, salida 2.5”	1	Inyección de flujo de aire hacia el colector solar y la cámara de secado.
16	Borneras	34	Conexión (alimentación, sensores, etc)
17	Riel DIN	1	Montaje estructurado de los componentes eléctricos.
18	Canaleta PVC 33×25 mm	1	Organización del cableado dentro del tablero.
19	Cable eléctrico 18 AWG	40 m	Cableado y conexiones de control.
20	Cable eléctrico 12 AWG	15 m	Alimentación del sistema de potencia.
21	Cable eléctrico 22 AWG	20 m	Conexión de señales de sensores
22	Jumper para ensamblar	1	Ensamblaje y conexiones

Fuente: Elaboración propia a partir de la implementación del sistema.

2.6 Diseño de tablero

Para la elaboración del sistema, se construyó un tablero eléctrico en el que se integraron todos los componentes principales del proyecto. El diseño se realizó considerando criterios de seguridad eléctrica, organización interna y facilidad de

mantenimiento, de manera que se asegurara un montaje confiable y accesible para futuras intervenciones.

En el tablero se dispusieron el controlador lógico programable Siemens LOGO! 8.2, la Raspberry Pi 3 Modelo B, los seccionadores, el breaker principal, las borneras, el relé para la resistencia y los elementos de señalización. La distribución de los equipos se efectuó sobre un riel DIN, lo que permitió un montaje estructurado y ordenado, mientras que el cableado fue guiado mediante canaletas internas que facilitaron la organización y redujeron riesgos de interferencias o fallos eléctricos.

El diseño incluyó también un conjunto de pulsadores de arranque/parada para la resistencia, el blower y el modo automático, así como un pulsador de paro de emergencia para asegurar la seguridad del operador. Los pilotos luminosos de color verde y rojo fueron instalados en el frente del tablero para indicar el estado de encendido y apagado de cada elemento, lo que proporcionó una referencia visual inmediata al operador.

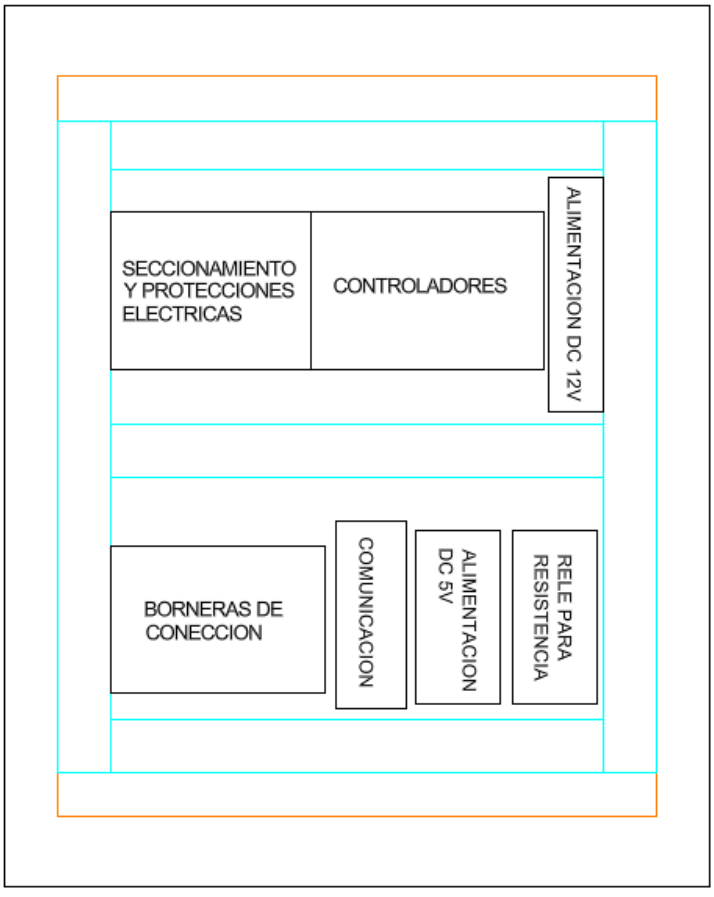


Gráfico 2.3 Boceto del tablero de control.

2.7 Programación de la Raspberry Pi

Se empleó una Raspberry Pi 3 junto con el sistema operativo Raspberry Pi OS, la distribución oficial basada en Linux Debian, para llevar a cabo la implementación del sistema de adquisición de datos y comunicación, ya que ofreció estabilidad y soporte para las librerías necesarias en el desarrollo.

El programa fue desarrollado en Python 3, utilizando librerías específicas para la adquisición de datos, comunicación y manejo de archivos:

- **pymodbus 2.5:** empleada para implementar el servidor Modbus TCP/IP y establecer la comunicación bidireccional entre la Raspberry Pi y el PLC Siemens LOGO! 8.2. Se seleccionó la versión 2.5.x debido a que la versión 3.x presentaba cambios en la API que ocasionaban incompatibilidades con métodos usados en el proyecto.
- **adafruit_dht:** utilizada para la adquisición de datos de los sensores DHT22 de humedad relativa y temperatura.
- **RPi.GPIO:** empleada en el control de salidas digitales asociadas a pilotos y relevadores.
- **Librerías estándar de Python:** time, threading, datetime, csv y os, necesarias para el manejo de procesos concurrentes, registro de datos y administración de archivos.

El código ejecutado en la Raspberry Pi cumplió cuatro funciones principales:

1. Lectura concurrente de sensores DHT22:

   Se conectaron seis sensores en diferentes posiciones de la cámara de secado. El programa realizó lecturas en paralelo con manejo de timeouts para evitar bloqueos. En caso de fallas consecutivas, los valores se reemplazaban por 0.0 tanto en el registro Modbus como en el archivo CSV, de acuerdo con el umbral configurado.

2. Comunicación con el PLC mediante Modbus TCP/IP:

   La Raspberry Pi operó como servidor Modbus TCP, publicando en tiempo real los datos de los sensores hacia el PLC Siemens LOGO! 8.2. Dichos datos fueron empleados por el PLC para ejecutar la lógica de control del proceso de secado.

3. Recepción de señales desde el PLC:

   Además de enviar información, la Raspberry Pi recibió instrucciones del PLC a través de registros Modbus. Estas señales se utilizaron para accionar salidas GPIO que controlaban pilotos y relevadores, complementando las salidas físicas limitadas del PLC.

4. Almacenamiento de datos en archivos CSV:

   Los datos adquiridos se almacenaron en la propia Raspberry Pi en archivos de formato CSV, configurados con rotación diaria. El intervalo de muestreo se estableció en 5 minutos, dado que la temperatura es una variable de cambio lento y no se requerían frecuencias de adquisición más altas para representar adecuadamente la tendencia del proceso.

La programación de la Raspberry Pi fue un elemento clave en la arquitectura del sistema, ya que permitió la adquisición confiable de las variables ambientales, la comunicación bidireccional con el PLC y la generación de registros históricos en un formato accesible para su análisis posterior.

2.8 Programación del PLC LOGO!

La programación del controlador lógico programable se realizó en el software LOGO! Soft Comfort v8.4, empleando diagramas en bloques de funciones (FBD). La lógica de control se estructuró en dos modos de operación: manual y automático, garantizando flexibilidad en la operación y seguridad en el uso del equipo.

En el modo manual, tanto la resistencia eléctrica como el blower pudieron ser accionados desde las botoneras físicas ubicadas en el tablero de control o desde las botoneras virtuales disponibles en la interfaz web. Este esquema permitió que el operador contara con redundancia en los mecanismos de control, manteniendo la supervisión del estado del sistema en todo momento.

En el modo automático, se implementó un control ON/OFF con histéresis, configurado con una ventana de operación de ±2 °C respecto al valor de consigna. De esta manera, la resistencia se encendía y apagaba automáticamente para mantener la temperatura dentro del rango establecido, mientras que el blower permanecía encendido durante todo el ciclo de secado. En este modo, se bloquearon

los accionamientos manuales tanto de la resistencia como del blower, a fin de prevenir activaciones accidentales mediante pulsos en las botoneras físicas. La desactivación del modo automático se efectuó exclusivamente desde la botonera de apagado del modo automático.

La recepción de las señales de humedad y temperatura se realizó mediante el protocolo Modbus TCP/IP, configurando al PLC en modo cliente. A través de entradas analógicas de red, el LOGO recibió los datos transmitidos por la Raspberry Pi. Para la lógica de control, se seleccionaron cuatro señales de temperatura correspondientes a los sensores DHT22 TH6, TH5, TH4 y TH1, con las cuales se calculó un valor promedio que fue comparado contra el setpoint configurado en la interfaz web.

Adicionalmente, se programó el envío de cuatro datos tipo coil hacia la Raspberry Pi, empleando salidas digitales de red. Estas señales se utilizaron para accionar las luces piloto correspondientes al estado del blower y del modo automático, integrando de esta manera la supervisión visual del proceso.

La interfaz de usuario se desarrolló en LOGO! Web Editor, en la cual se diseñaron dos pantallas principales. En la primera se incluyeron los controles para el accionamiento manual y automático, junto con indicadores visuales animados que facilitaron la supervisión:

- Una baliza azul en formato GIF para señalar el estado activo del modo automático.
- Un ventilador animado en GIF para indicar el encendido del blower.
- Una flama animada en GIF para mostrar el estado de encendido de la resistencia.

Esta programación integró tanto los mecanismos de control como la interfaz de usuario, proporcionando una solución intuitiva y segura para la operación del secador híbrido.

2.9 Consideraciones técnicas y normativas

A lo largo del desarrollo del sistema se consideraron técnicas y normativas que garantizaron la seguridad, confiabilidad y replicabilidad del proyecto:

- **Seguridad eléctrica:** La construcción del tablero de control se realizó siguiendo criterios básicos de seguridad establecidos en la normativa IEC 60364 – Instalaciones eléctricas de baja tensión, así como en buenas prácticas de cableado industrial. Se incorporaron protecciones mediante breaker principal y seccionadores independientes para los circuitos de potencia, blower y control, a fin de aislar adecuadamente cada subsistema y proteger a los usuarios ante sobrecorrientes o fallas.
- **Organización de componentes:** El montaje de los equipos en el tablero se efectuó sobre riel DIN, lo que permitió un orden estructurado y facilitó el mantenimiento. Asimismo, se utilizaron canaletas plásticas para guiar el cableado y borneras para las conexiones de potencia y señales, asegurando un sistema accesible y seguro.
- **Comunicación industrial:** Para la transmisión de datos se implementó el protocolo Modbus TCP/IP, ampliamente aceptado en entornos industriales por su estandarización, compatibilidad y simplicidad. Esto garantizó la interoperabilidad entre el PLC y la Raspberry Pi, además de permitir la escalabilidad futura hacia sistemas de mayor complejidad.
- **Gestión de datos:** El almacenamiento de registros en archivos CSV permitió disponer de información histórica de forma sencilla y accesible, sin requerir bases de datos externas. Esta estrategia se alineó con principios de eficiencia y facilidad de análisis, al ser compatible con herramientas comunes como Excel y plataformas de análisis de datos.
- **Interfaz de usuario:** El desarrollo de la interfaz en LOGO! Web Editor se enfocó en la seguridad y en la usabilidad. Al estar restringida a la red local del CDTS, se evitó la exposición del sistema a accesos externos no autorizados. Además, se incorporaron elementos visuales intuitivos que facilitaron la supervisión de las variables y el control de los actuadores.

Estas consideraciones aseguraron que el sistema cumpliera no solo con la funcionalidad prevista, sino también con criterios de seguridad, confiabilidad y buenas prácticas de ingeniería, lo que garantiza su utilidad como prototipo académico y su potencial de adaptación a entornos de mayor exigencia.

CAPITULO 3

Con el sistema construido se procedió a tomar muestras de datos por periodos de una hora para comparar resultados y obtener una gráfica de histórico. Además, se utilizaron una serie de sensores termopares externos con el fin de tener una referencia externa con la cual comparar y validar resultados.

También se hicieron pruebas de comunicación entre el servidor raspberry con los demás equipos para validar el acceso a los archivos con los datos tal como lo solicito el cliente.

3. Análisis y Resultados

3.1 Tabla de Precios de componentes.

Tabla 3.1 Tabla de precios.

Material	Cantidad	Precio por Unidad
PLC Logo V8.2	1	$320
Raspberry pi 3 modelo B	1	$125
Luz Piloto 22[mm] Led verde de 220V	3	$5
Luz Piloto 22[mm] Led Rojo de 220V	3	$5
Cable concéntrico de 4 hilos 12awg	15[m]	$4
Switch de 8 puertos RJ45	1	$35
Fuente conmutada 12V 10A	1	$15
Pulsador doble metálico sin Luz	3	$10
Pulsador tipo hongo de paro de emergencia 22[mm]	1	$9
Sensor dht22	9	$6
Cable 18awg de un hilo café	5[m]	$0.80
Cable 18awg de un hilo azul	5[m]	$0.80
Cable 18awg de un hilo negro	5[m]	$0.80
Tomacorriente sobrepuesto de 220V	2	$7.0
Canaleta pvc 2[m]x20[mm]x12[mm] de ancho	3	$3.0

Tablero plástico con plafón metálico 50[cm]x40[cm]x18[cm]	1	$70
Total		$770

Fuente: Elaboración propia a partir de la implementación del sistema

3.2 Análisis y comparación de datos de sensores dht22 y termopares externos:

Para la obtención de datos se consideró cubrir en lo posible todas las secciones de la cámara de secado, colocando 4 de los 6 sensores en las 4 bandejas a diferentes alturas, el 5to sensor en una esquina de la cámara el cual se considera el punto muerto donde hay menos circulación de aire caliente. Finalmente, el 6to sensor se lo coloco en el exterior de la cámara para poder tener un censado de la temperatura del ambiente. En este análisis y comparación se utilizarán únicamente los 4 primeros sensores y como referencia 4 sensores termopares colocados en la misma posición. Además, se debe tomar en cuenta que el tiempo del guardado de los datos del sistema coincida con el tiempo en que se tomen los datos de los termopares para evitar variaciones en lo posible.

Tabla 3.1.232 Comparativa de ubicación de sensores DHT22 vs termopares.

Sensores DHT22	Sensores Termopar
T6	T1
T5	T2
T4	T3
T1	T4

Fuente: Elaboración propia a partir de la implementación del sistema

El cuadro anterior muestra la posición de cada uno de los 4 sensores DHT22 ubicados a diferentes alturas en comparación a las ubicaciones de los sensores termopares.

Tabla 3.3 Valores de temperatura de termopares de referencia.

Hora	T1	T2	T3	T4
16:45	32,8	33,4	33,4	33,3
16:50	83,1	81	89,3	73,5
16:55				
17:00				
17:05				
17:10	84,3	82,1	85,3	78,7
17:15	88,7	87,5	92	85
17:20	83,2	85,8	91,6	82,5
17:25	90,3	89,4	93,9	88,1
17:30	91,8	90	95,4	89,2
17:35	94,7	90,5	96,2	90,2
17:40	92,7	91,6	97,1	94,3
17:45				
18:35	30,5	30,9	31,2	35,6
18:40	64,5	60,7	58,8	53
18:45	63	65	64,3	58,2
18:50	63,3	65	64,3	53,2
18:55				

Fuente: Elaboración propia a partir de la implementación del sistema

Tabla 3.4 Valores de temperatura de sensores dht22.

Fuente: Elaboración propia a partir de la implementación del sistema

Hora	T1	T2 (T Ambiental)	T3 (Punto Muerto)	T4	T5	T6
16:43:32	35	31	32,1	34,9	35,2	35,1
16:48:38	36,1	30	43,5	36,6	36,3	36,4
16:53:38	52,7	31,9	61,2	52,6	53,1	53,2
16:58:44	63,9	33,3	68	63,9	64,3	63,6
17:03:50	69,7	34,1	71	70,1	70	69
17:08:50	70,9	33,6	67,8	71,7	71,4	69,2
17:13:50	72,6	33,7	77,2	72,8	72,5	70,8
17:18:56	75,4	34,4	0	75,4	75,3	73,2
17:24:02	75,2	33,4	79,1	75,5	75,2	72,7
17:29:02	76,6	34,9	0	76,6	76,5	74,1
17:34:08	77,4	33	0	77,5	77,3	74,7
17:39:14	78	34,5	0	78,1	77,8	75,1
17:45:00						
17:50:00						
18:34:56	32,3	27,3	31,7	32,1	32	31,6
18:39:56	37	27,5	36,8	38,3	37,5	38,5
18:45:02	50,9	28,6	49,7	51,8	53,3	53,8
18:50:02	53,4	29,3	55	55,6	58,5	57,8
18:55:08	51,9	29,6	51,3	53,3	54,9	53,4

El sistema de secado hibrido se configuro para guardar datos cada 5 minutos, debido a que la temperatura es un dato de baja variabilidad. Tomando en cuenta esto, se tomaron las medidas obtenidas de los termopares en intervalos de tiempo aproximados, aunque con cierto margen de diferencia en tiempo debido a algunos errores de comunicación que se corrigieron posteriormente.

Se colocaron los márgenes de tiempo donde hubo errores de comunicación y no se pudieron tomar datos (rojo), con el fin de no cortar la continuidad en el intervalo de tiempo. De la misma forma se colocaron las franjas de tiempo donde si se obtuvieron datos, pero no se los utilizo en el análisis porque no había con que compararlos (gris).

Tabla 3.1.5. Valores de Humedad respecto al tiempo

Fecha	Hum1	Hum2 (Hum del Ambiente)	Hum3	Hum4	Hum Punto Muerto
16:43:32	40,7	42,1	43	42,1	47,7
16:48:38	39	41	40,8	40,5	30
16:53:38	21,7	23,4	22,2	21,1	15,9
16:58:44	15,8	16,5	14,8	14	12,9
17:03:50	13,8	14,3	12	12	12
17:08:50	13,7	13,7	11,4	11,2	13
17:13:50	13,2	13,4	11,1	10,9	10,4
17:18:56	12,6	12,6	10,3	10	0
17:24:02	12,6	12,7	10,5	10,2	9,9
17:29:02	12,4	12,3	10,1	10	0
17:34:08	12,3	12,2	10	9,6	0
17:39:14	12,2	12	9,8	9,4	0
17:44:20	12,1	11,9	9,6	9,3	0

Fuente: Elaboración propia a partir de la implementación del sistema

3.3 Evolución de la temperatura del secador hibrido

Los sensores DHT22 mostraron un incremento progresivo de la temperatura en el interior de la cámara de secado, alcanzando valores aproximados entre 75 °C y 78 °C. Cabe aclarar que la temperatura optima del secador es aproximadamente de 50 °C a 55 °C ya que a temperaturas más altas cualquier muestra de biomasa pierde sus propiedades órgano-eléctricas, pero para consideraciones del análisis no se controló la temperatura límite del secador.

- **Etapa inicial (16:45–16:55)**: temperaturas entre 30 y 40 °C, con alta humedad relativa (40–50%).
- **Etapa de estabilización (17:00–17:20)**: ascenso sostenido de la temperatura por encima de 60 °C y descenso de la humedad a valores entre 10–15%.
- **Etapa de máxima operación (17:25–17:45)**: temperaturas superiores a 75 °C, con humedades cercanas al 10%.

Se presenta un comportamiento coherente con lo esperado en el sistema de secado, donde el aire caliente reduce el nivel de humedad interna del material a secar.

3.4 Comparación Sensores DHT22 vs Termopares.

Al comparar las lecturas de ambos sistemas de medición se observó lo siguiente:

- Los termopares registraron temperaturas de referencia estables y coherentes con la dinámica de calentamiento del sistema.
- Los DHT22 siguieron la misma tendencia, pero en algunos momentos presentaron desviaciones superiores al ±5 °C.
- Se identificaron valores atípicos (0.0 °C y 0.0 % HR en el sensor DHT22 #3), que corresponden a fallas de comunicación y fueron tratados como datos faltantes en el análisis.

En promedio, los errores relativos de los DHT22 respecto a los termopares oscilaron entre 2% y 8%, con picos mayores en los momentos de rápida variación térmica. Estos resultados concuerdan con las especificaciones técnicas de los DHT22, que tienen un margen de error de ±0.5 °C, pero son sensibles a condiciones extremas de calor y baja humedad.

3.5 Comportamiento de la humedad relativa.

Los DHT22 registraron una disminución drástica de la humedad relativa conforme aumentaba la temperatura. En la etapa inicial, los valores se mantuvieron entre 40–50%, mientras que al llegar a 70–75 °C descendieron hasta 10–12%, lo que confirma que el aire dentro de la cámara se volvió más seco y por tanto más efectivo para la extracción de humedad de los productos.

3.6 Visualización de datos y control de temperatura en la página web.

La solución desarrollada incluyó una interfaz web que permitió tanto la visualización en tiempo real de las variables medidas como el control de los parámetros operativos del secador híbrido.

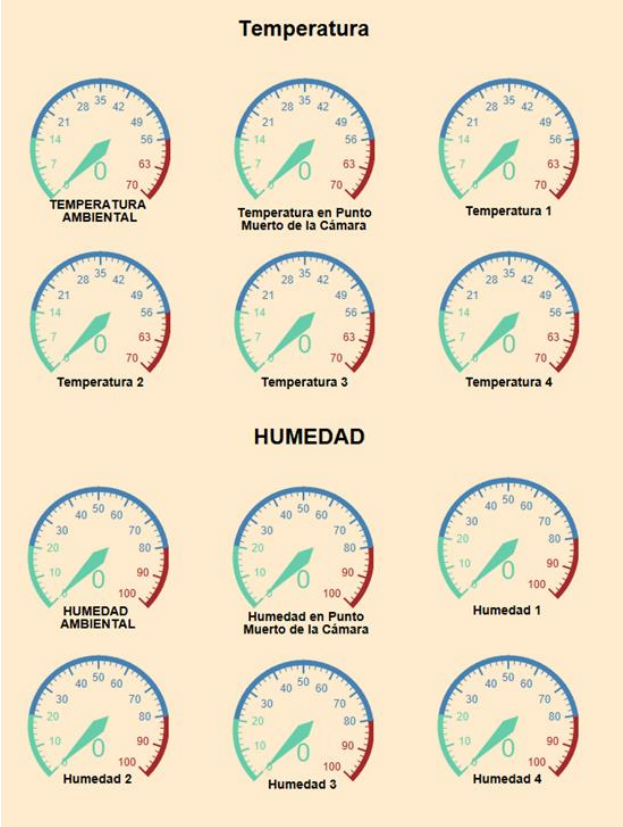


Gráfico 3.1Pestaña de interfaz web con valores de temperatura y humedad.

En dicha página se mostraron de manera dinámica los valores de temperatura y humedad relativa obtenidos a partir de los seis sensores DHT22 distribuidos dentro de la cámara de secado. Estos datos son actualizados automáticamente, lo que facilita al usuario el seguimiento del comportamiento térmico del sistema durante el proceso de secado.

Además de la visualización, la interfaz web permitió establecer el valor de consigna (setpoint) de temperatura. Para ello se incorporó un control tipo “slider”, con el cual el usuario seleccionaba la temperatura objetivo deseada. Este valor era transmitido al PLC Siemens LOGO! 8.4, que ajustaba la operación de las resistencias eléctricas y el blower para mantener la cámara en condiciones óptimas de secado.

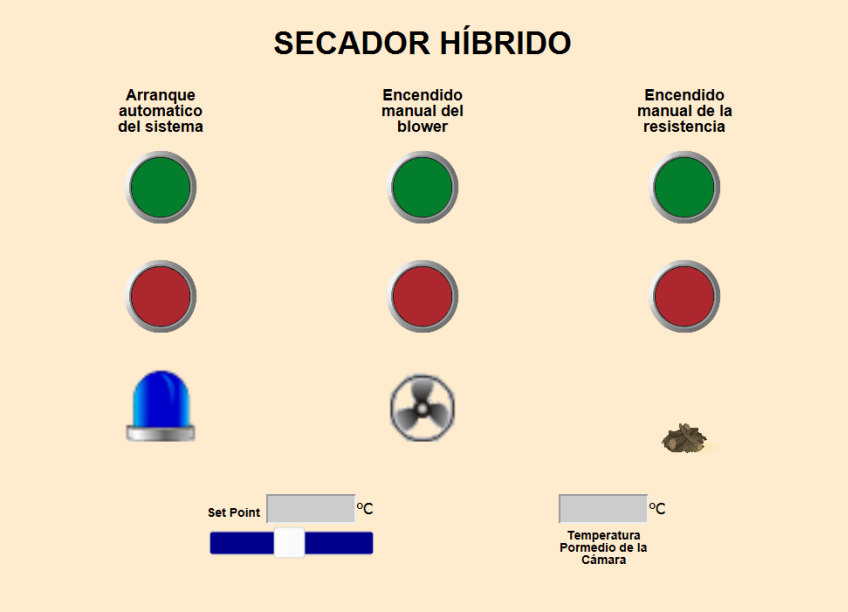


Gráfico 3.2 Pestaña de interfaz web para control del sistema.

El uso de esta plataforma web simplificó la interacción con el sistema, eliminando la necesidad de ajustar parámetros de manera manual en el PLC. Asimismo, garantizó que cualquier operador, incluso con conocimientos básicos, pudiera supervisar y modificar la temperatura de trabajo de forma intuitiva y segura.

Finalmente se agregó una red virtual gracias al uso del software Zero Tier, utilizando la Raspberry como Gateway para otorgarle una ip virtual al PLC Logo ya que este no es capaz de instalar el programa por su cuenta. De esta forma cualquier equipo con el permiso de la red virtual es capaz de ingresar a la página web y controlar el sistema desde cualquier parte del mundo siempre y cuando este esté encendido y conectado a la red física del CDTS.

3.7 Discusión de resultados

- El sistema de control y monitoreo permitió seguir en tiempo real la evolución de la temperatura y la humedad en el secador híbrido, lo que facilitó la supervisión del proceso y evitó la necesidad de registros manuales.
- A pesar de las limitaciones de precisión de los sensores DHT22, la tendencia general de sus mediciones coincidió con la de los termopares de referencia, validando su uso en aplicaciones de bajo costo donde se prioriza la supervisión general y no la exactitud científica.

- Las discrepancias detectadas se debieron principalmente a fallas de comunicación en el sensor DHT22 #3 y a las limitaciones de este tipo de sensor en condiciones extremas de alta temperatura y baja humedad.
- La interfaz web desarrollada representó un avance significativo, al mostrar dinámicamente las variables medidas (temperatura y humedad relativa) y permitir el control directo del setpoint de temperatura mediante un control tipo “slider”. Esta solución mejoró la interacción con el sistema, haciéndolo accesible incluso para operadores con conocimientos básicos.
- La integración de la red virtual mediante ZeroTier permitió el acceso remoto desde cualquier dispositivo autorizado, garantizando que la supervisión y el control del sistema pudieran realizarse desde cualquier parte del mundo. Esta característica aumentó notablemente la versatilidad del sistema y su aplicabilidad en entornos rurales o industriales distribuidos.
- En general, el sistema demostró ser tecnológicamente factible y económicamente viable, con la ventaja adicional de escalabilidad gracias a su conectividad y facilidad de operación.

3.8 Conclusiones parciales del capítulo

- El secador híbrido alcanzó condiciones adecuadas para el secado de biomasa, registrando temperaturas superiores a 70 °C y reducciones de humedad relativa por debajo del 15%, lo que confirma su eficiencia térmica.
- Los sensores DHT22, aunque menos precisos que los termopares, demostraron ser útiles para el control general del sistema, validando su empleo en proyectos de bajo costo y fácil implementación.
- La comparación con los termopares permitió identificar márgenes de error en un rango entre 2% y 8%, reforzando la validez de los datos y mostrando la necesidad de combinar sensores económicos con dispositivos de referencia para análisis más precisos.
- La plataforma web implementada permitió tanto la visualización en tiempo real de los parámetros como la modificación del ‘setpoint’ de temperatura, simplificando la operación y eliminando la necesidad de interactuar directamente con el PLC.

- La integración de ZeroTier como red virtual otorgó acceso remoto seguro y global al sistema, lo que convierte a la solución en una herramienta flexible y escalable para distintos entornos de aplicación.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El secador híbrido logró mantener temperaturas superiores a 70 °C y niveles de humedad relativa inferiores al 15%, condiciones que resultaron adecuadas para el secado de biomasa, validando así la efectividad de la propuesta.
- La incorporación del PLC Siemens LOGO! 8.4 permitió la automatización del proceso, reduciendo la dependencia de la intervención manual, mejorando la continuidad del secado y aumentando la confiabilidad operativa.
- La comparación de lecturas entre los sensores DHT22 y los termopares de referencia mostró coherencia en la tendencia, a pesar de las limitaciones de precisión de los DHT22. Esto confirmó que son apropiados para aplicaciones de monitoreo en campo y de bajo costo, siempre que se utilicen con criterios de validación.
- La plataforma web desarrollada facilitó la visualización en tiempo real de los parámetros de operación, así como la modificación del setpoint de temperatura. Esta herramienta simplificó la interacción con el sistema, eliminando la necesidad de ajustes directos en el PLC y garantizando un uso más accesible incluso para operadores con conocimientos básicos.
- La integración de la red virtual ZeroTier permitió el acceso remoto seguro y global al sistema de secado, ampliando su versatilidad y ofreciendo la posibilidad de supervisar y controlar el proceso desde cualquier lugar del mundo.
- El análisis de costos evidenció que el sistema diseñado resulta económicamente accesible, lo que refuerza la viabilidad de su implementación en contextos rurales o pequeñas agroindustrias.
- En conjunto, el trabajo cumplió con los objetivos planteados, al demostrar que es posible implementar un sistema de secado híbrido solar–eléctrico automatizado, remoto, viable técnica y económicamente.

4.2 Recomendaciones

- Evaluar el reemplazo o complementación de los sensores DHT22 por dispositivos de mayor precisión (como SHT31, SHT85 o PT100 con módulos de adquisición) en proyectos que requieran datos de carácter científico o de investigación.
- Ampliar las pruebas a diferentes productos agrícolas con el fin de determinar la adaptabilidad del sistema a distintos procesos de secado y establecer curvas específicas de comportamiento.
- Integrar un módulo de registro en la nube para permitir monitoreo remoto avanzado, almacenamiento histórico y análisis de datos a través de plataformas IoT o SCADA distribuidas.
- Desarrollar un sistema de control inteligente que, mediante algoritmos de predicción o control adaptativo, optimice el uso de la resistencia eléctrica en función de la radiación solar disponible, reduciendo así el consumo energético.
- Considerar la incorporación de un sistema de seguridad adicional en la interfaz web y en la red virtual para reforzar la protección contra accesos no autorizados.
- Realizar pruebas de eficiencia energética comparativa entre el uso solar y eléctrico, de manera que se puedan cuantificar los ahorros obtenidos y la contribución a la sostenibilidad del proceso.

BIBLIOGRAFÍA

- Cid, D., & Correa, C. (2019). Automatización de un prototipo secador de cacao con control y monitoreo de variables físicas mediante una aplicación móvil. Universidad Técnica de Manabí.

- Dharmender, D., Khura, T. K., Mani, I., Parray, R. A., Dubey, A., Kumari, S., Malkani, P., & Mandal, S. (2024). Improving vegetable seed quality and dryer performance with an automated desiccant-assisted hybrid solar dryer system. AATCC Review, 12(02), 284-289. https://doi.org/10.21276/AATCCReview.2024.12.02.284

- Löppenberg, M., & Schwung, A. (2023). Self optimisation and automatic code generation by evolutionary algorithms in PLC based controlling processes. arXiv preprint arXiv:2304.05638.

- Malkani, P., Dubey, A., & Mandal, S. (2024). Design and Performance of an Automated Desiccant-based Hybrid Solar Dryer for Onion Seed Drying. International Journal of Agricultural Science and Technology.

- Oliveira, C. M. de, Pacheco, J. R., & Almeida, G. T. (2021). Diseño e implementación de un sistema automático de control para regulación y monitoreo de las condiciones internas de un secador de cacao. Universidad Técnica Estatal de Quevedo.

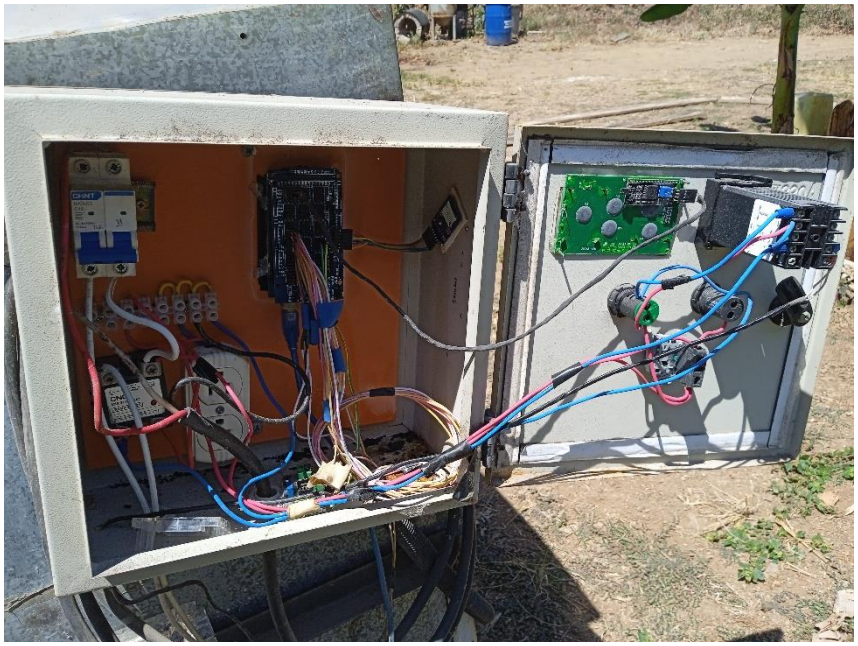
- Paes, J. L., Ramos, V. A., de Oliveira, M. V. M., Pinto, M. F., Lovisi, T. A. P., & de Souza, W. D. (2022). Automation of monitoring of drying parameters in hybrid solar electric dryer for agricultural product. Revista Brasileira de Engenharia Agrícola e Ambiental, 26(4), 283-291. https://doi.org/10.1590/1807-1929/agriambi.v26n4p283-291

- Rosa, E. de O., Grabarski, L., Fragoso, M. F., Ferrari, A. C. K., Schuertz, J. R., & da Silva, C. A. G. (2020). An embedded system for monitoring industrial air dehumidifiers using a mobile Android application for IEEE 802.11 networks. arXiv preprint arXiv:2008.11123}

- Solowjow, E., Ugalde, I., Shahapurkar, Y., Aparicio, J., Mahler, J., Satish, V., Goldberg, K., & Claussen, H. (2020). Industrial Robot Grasping with Deep Learning using a Programmable Logic Controller (PLC). arXiv preprint arXiv:2004.10251.

ANEXOS

Anexo 1. Fotos de proceso de construcción del Sistema de Secado Hibrido



Apendice A.

Diseños en 3D

Diseño 3d de bases para riel Din:

Se realizaron diseños de base para riel Din para imprimir en 3D debido para tener una mayor organización en el tablero, así como una mejor presentación.

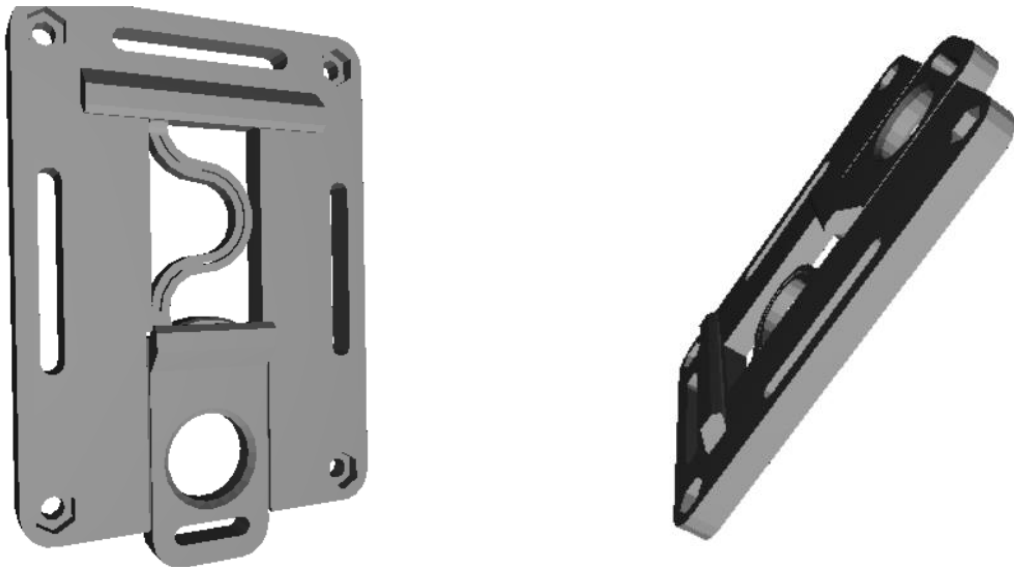


Imagen 1. Modelo de base para sujeción de módulo de 4 reles y raspberry pi

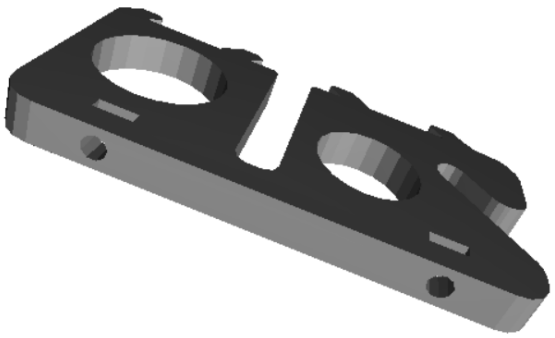


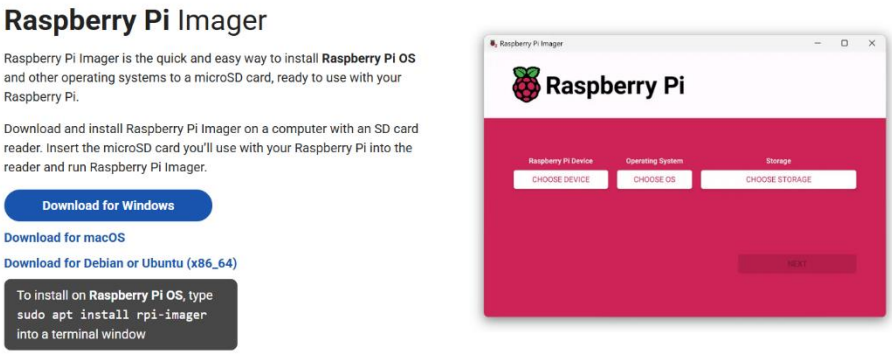
Imagen 2. Modelo de base para sujeción de switch de 8 puertos rj45

Imagen 3. Modelo de base para sujeción de switch de 8 puertos rj45, vista frontal

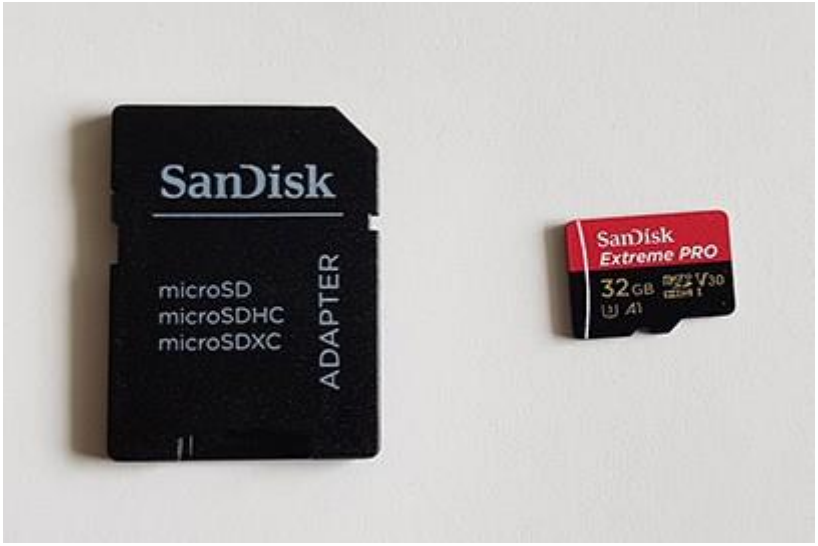
Apéndice B.

Pasos para instalación y configuración del sistema Rasbian

Para la programación de la raspberry se procede a instalar el sistema basado en Linux, Raspbian. Se descarga de la página oficial donde se encuentra el programa de instalación para Windows.

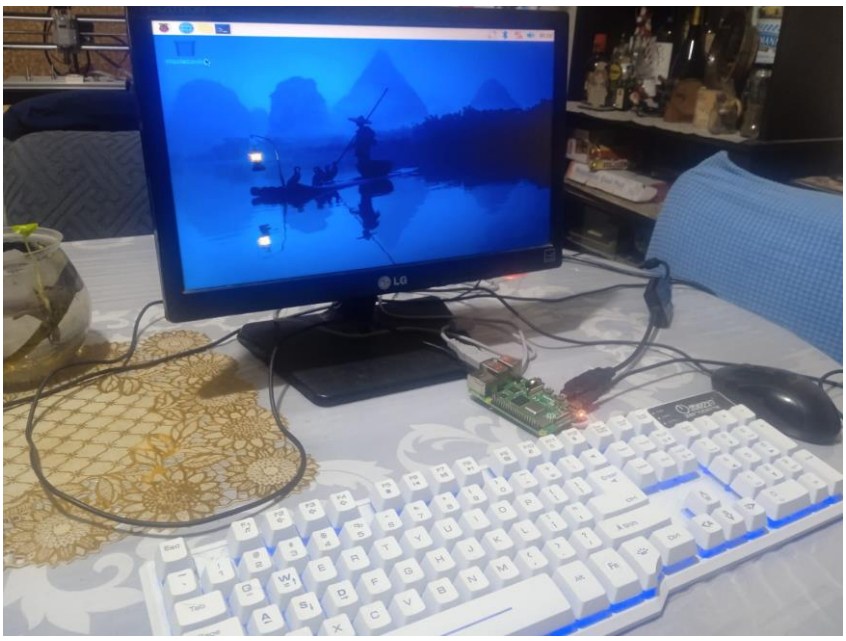


Normalmente se utiliza una SD de 32Gb con el fin no alcanzar el máximo de capacidad y evitar que se alenté.

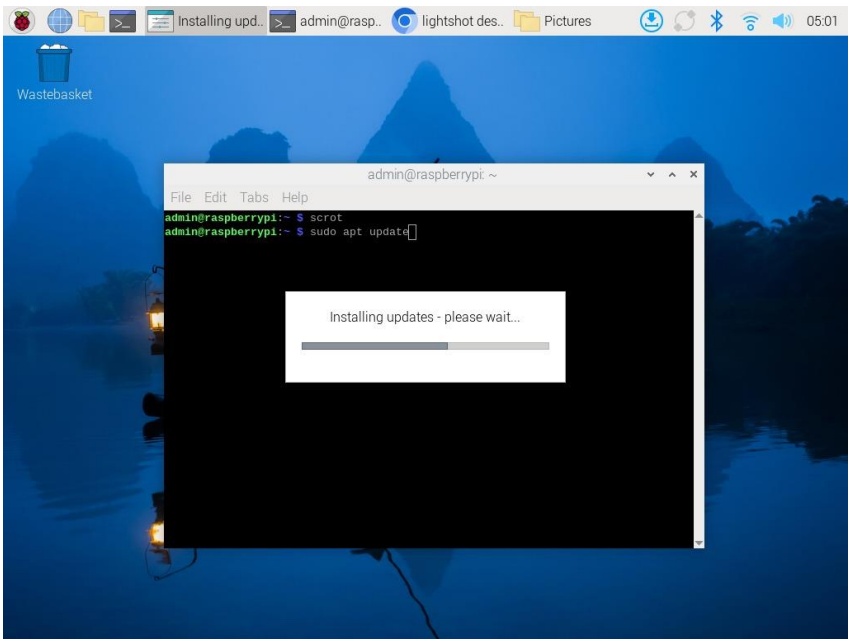


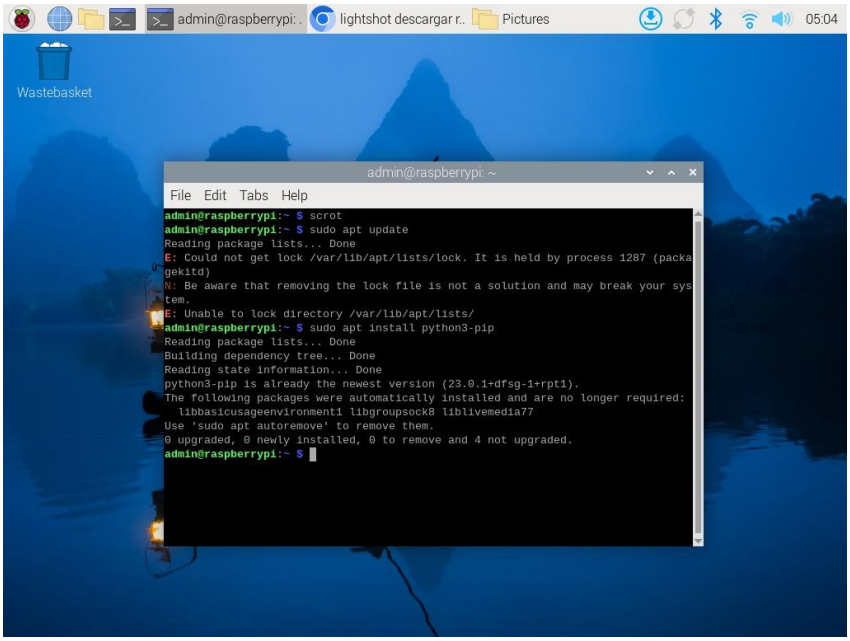
Posterior a la instalación del programa en Windows Raspberry Pi Imager, se lo ejecuta y con la SD colocada en la ranura de la laptop se procede a hacer la instalación del

sistema. Hay que considerar que ls SD será formateada y cualquier información previa se perderá.

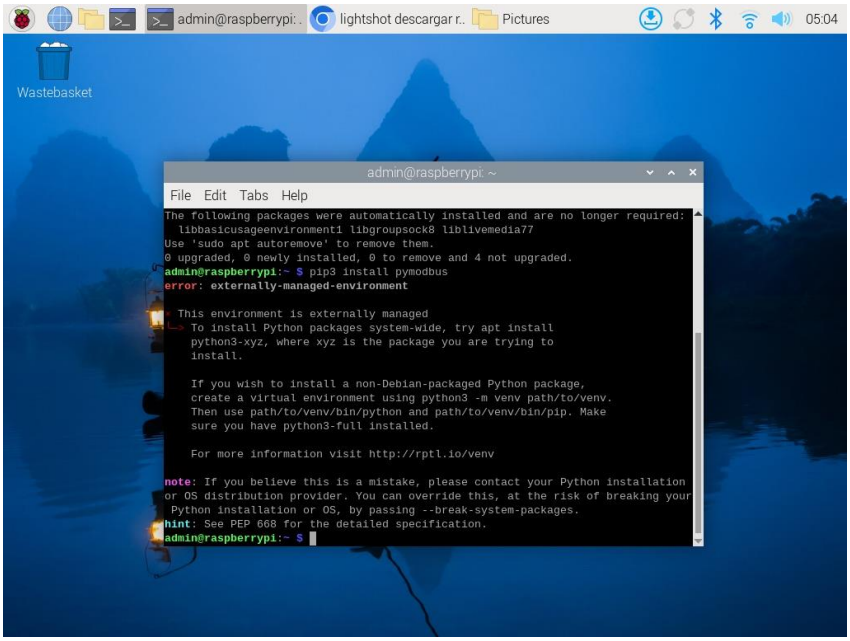


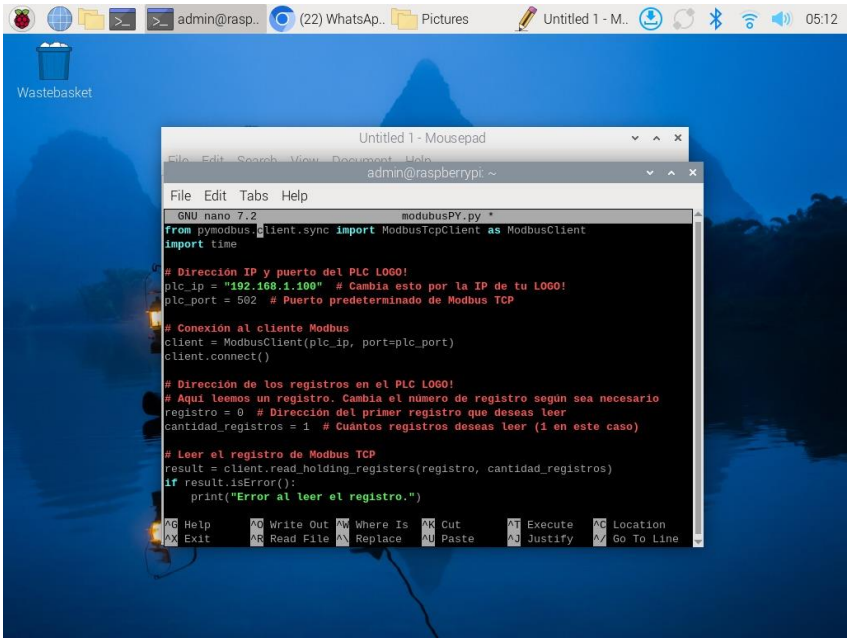
Una vez instalada el sistema se procede a encender la raspberry para verificar que el sistema se haya instalado bien. Considerar la alimentación de 5V para la Raspberry y la necesidad de un mouse y teclado para su manipulación.





Una vez instalado el sistema se realiza la actualización con el código sudo apt update y se espera, luego se realizan las instalaciones de Python y librerías para la comunicación con el PLC Logo

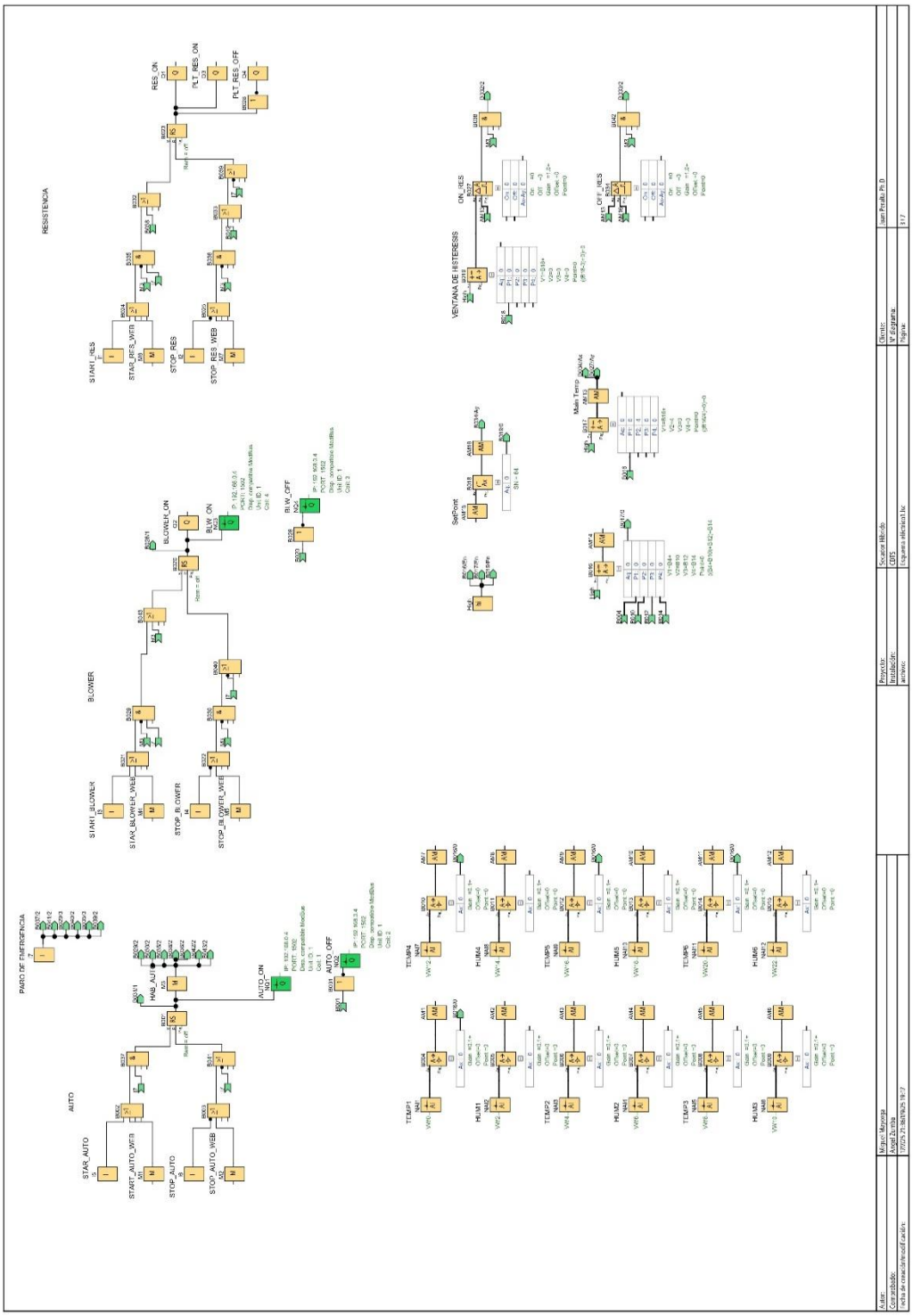




Finalmente se procede con la escritura y posterior ejecución del código. Se deben leer los datos de 6 sensores DHT22 para mandarlo por comunicación Modbus al PLC Logo.

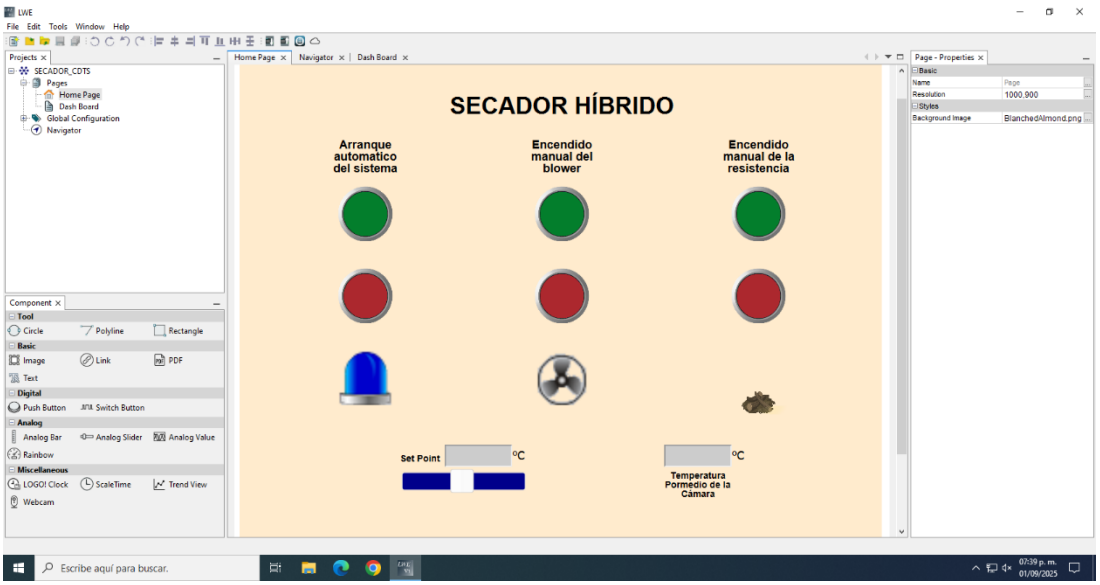
Apéndice C.

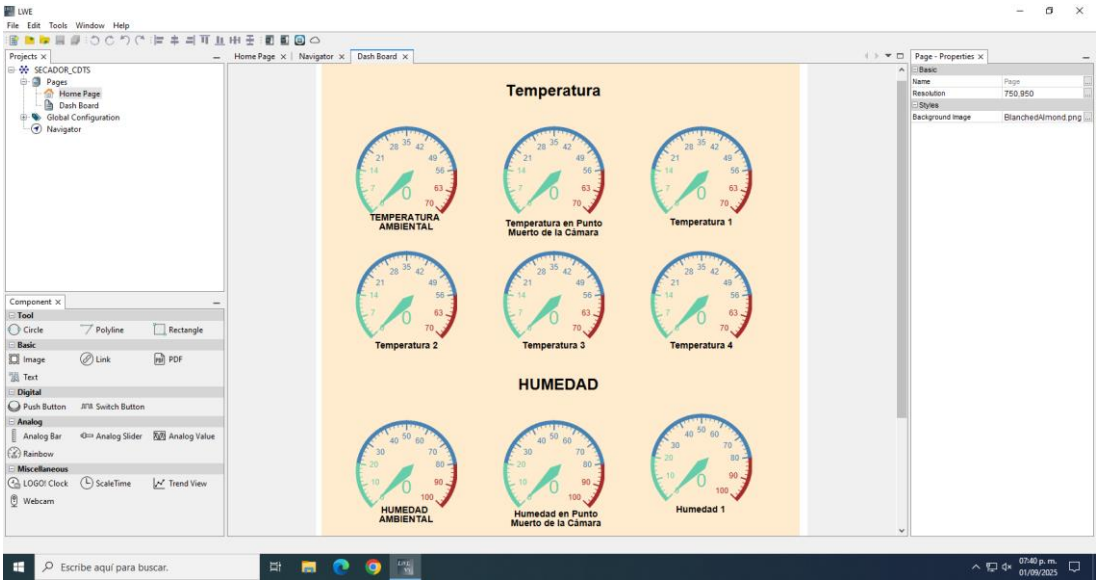
Programación de PLC Logo:

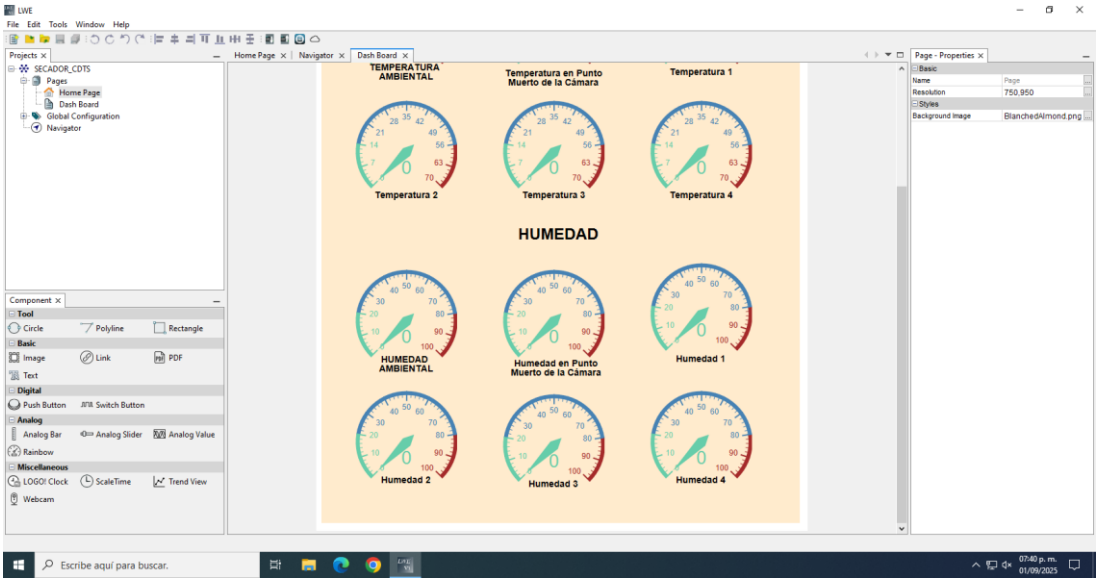


Usando diagrama de bloques se procedió con la programación del PLC Logo sin considerar las variables de los sensores DHT22 ya que estos vienen por comunicación Modbus y posteriormente se reflejan en el scada de Logo Web Editor.

Diseños de HMI para la visualización de variables:






LWE
File Edit Tools Window Help
Projects
SECADOR_CDTS
Pages
Home Page
Dash Board
Global Configuration
Navigator
Component
Tool
Circle
Polyline
Rectangle
Basic
Image
Link
PDF
Text
Digital
Push Button
Switch Button
Analog
Analog Bar
Analog Slider
Analog Value
Rainbow
Miscellaneous
LOGO! Clock
ScaleTime
Trend View
Webcam
Home Page
Navigator
Dash Board
TEMPERATURA AMBIENTAL
Temperatura en Punto Muerto de la Cámara
Temperatura 1
Temperatura 2
Temperatura 3
Temperatura 4
HUMEDAD
HUMEDAD AMBIENTAL
Humedad en Punto Muerto de la Cámara
Humedad 1
Humedad 2
Humedad 3
Humedad 4
Page - Properties
Basic
Name
Page
Resolution
750,950
Styles
Background Image
BlanchedAlmond.png
Escribe aquí para buscar.
07:40 p. m.
01/09/2025

Apéndice D.

Código del script de Python para comunicación modbus entre PLC Logo y servidor Raspberry pi.

```
#!/home/admin/mi_entorno/bin/python
# -*- coding: utf-8 -*-
"""
Proyecto Sistema de Supervision y Control de un Secador Hibrido
Integrantes: Angel Zumba - Miguel Mayorga
Codigo - Servidor Raspberry
Raspberry Pi: Modbus TCP (pymodbus) + 6 DHT22 + 4 salidas (luces)

Comportamiento:
- CMD (print): si un sensor no entrega lectura válida EN EL CICLO -> 0.0 / 0.0 inmediato.
- Modbus/CSV: enceran (0/0) SOLO si las fallas consecutivas >= FALLAS_CONSEC_PARA_ENCERAR.
  Antes de ese umbral, se mantiene el ÚLTIMO VALOR VÁLIDO.
- Lectura en paralelo por sensor con timeout (no bloquea el bucle).
- Ciclo con sleep adaptativo para durar ~PERIODO_LECTURA_S.
- Puerto Modbus TCP: 502
"""

import time
import threading
import adafruit_dht
import board
import RPi.GPIO as GPIO
import csv, os
from datetime import datetime, date, timedelta
```

```
from pymodbus.datastore import (
    ModbusSlaveContext,
    ModbusServerContext,
    ModbusSequentialDataBlock,
)
from pymodbus.device import ModbusDeviceIdentification

# Compatibilidad pymodbus 3.x / 2.5.x
try:
    from pymodbus.server import StartTcpServer, ModbusConnectedRequestHandler  # >=3.x
except Exception:
    from pymodbus.server.sync import StartTcpServer, ModbusConnectedRequestHandler  # 2.5.x

# ---------------- CONFIG ----------------
SENSORES = [
    ("DHT1", board.D4),
    ("DHT2", board.D17),
    ("DHT3", board.D27),
    ("DHT4", board.D22),
    ("DHT5", board.D5),
    ("DHT6", board.D6),
]

SALIDAS_GPIO = {0: 23, 1: 24, 2: 25, 3: 26}
RELES_ACTIVOS_EN_BAJO = False

PERIODO_LECTURA_S = 6.0     # duración objetivo del ciclo
```

```
PERIODO_LOG_S     = 300.0     # CSV
INTERVALO_PRINT_S = 6.0     # impresión agrupada
_ultimo_print = 0.0

LECTURA_TIMEOUT_S = 5.0     # timeout por sensor (por ciclo)

# <<< UMBRAL CONFIGURABLE >>>
FALLAS_CONSEC_PARA_ENCERAR = 7  # <-- cambia aquí: # de fallas seguidas para poner 0 en Modbus/CSV

SERVER_BIND_IP = "0.0.0.0"
SERVER_PORT    = 1502
UNIT_ID        = 1

CSV_DIR = "/home/admin/datos_secador"
os.makedirs(CSV_DIR, exist_ok=True)
ROTACION_DIAS = 1

# ========= Inicialización =========
GPIO.setmode(GPIO.BCM)
for _, bcm in SALIDAS_GPIO.items():
    inicial = GPIO.HIGH if RELES_ACTIVOS_EN_BAJO else GPIO.LOW
    GPIO.setup(bcm, GPIO.OUT, initial=inicial)

def x10(v):
    return int(round(v*10)) if v is not None else 0

# HRs/coils base
hr_block = ModbusSequentialDataBlock(0, [0]*20)
```

```
co_block = ModbusSequentialDataBlock(0, [0]*16)
store   = ModbusSlaveContext(hr=hr_block, co=co_block, zero_mode=False)
context = ModbusServerContext(slaves={UNIT_ID: store}, single=False)

_conectado = False
ds_lock = threading.Lock()

# Estado por sensor
fallas_consecutivas = {nombre: 0 for nombre, _ in SENSORES}
last_good_x10 = {nombre: (0, 0) for nombre, _ in SENSORES}   # último valor válido para Modbus (x10)
last_good_flt = {nombre: (0.0, 0.0) for nombre, _ in SENSORES}# último valor válido para CSV/uso humano

class ConnLoggerHandler(ModbusConnectedRequestHandler):
    def connectionMade(self):
        global _conectado
        super().connectionMade()
        _conectado = True
        try:
            host, port = self.request.getpeername()
            print(f"[CONN] Cliente conectado: {host}:{port}", flush=True)
        except Exception:
            print("[CONN] Cliente conectado (peername no disponible)", flush=True)

    def connectionLost(self, reason):
        global _conectado
        print(f"[DISC] Cliente desconectado: {reason}", flush=True)
        _conectado = False
        super().connectionLost(reason)
```

```
identity = ModbusDeviceIdentification()
identity.VendorName = 'Raspberry Pi Server'
identity.ProductCode = 'PM'
identity.ProductName = 'DHT22+GPIO Server'
identity.ModelName = 'Secador Híbrido'
identity.MajorMinorRevision = '1.0'

def ventana_n_dias(hoy: date):
    idx = hoy.toordinal() // ROTACION_DIAS
    inicio_ord = idx * ROTACION_DIAS
    inicio = date.fromordinal(inicio_ord)
    fin = inicio + timedelta(days=ROTACION_DIAS - 1)
    return inicio, fin

def ruta_csv_actual(dt: datetime):
    d0, _ = ventana_n_dias(dt.date())
    return os.path.join(CSV_DIR, f"secador_{d0:%d_%m_%Y}.csv")

def preparar_csv(path):
    nuevo = not os.path.exists(path)
    if nuevo:
        with open(path, "a", newline="") as f:
            w = csv.writer(f)
            header = ["timestamp"]
            for i in range(1, 7):
                header += [f"T{i}", f"Hum{i}"]
            w.writerow(header)
```

```
def lectura_dht_valida(t, h):
    if t is None or h is None: return False
    if not (-40.0 <= t <= 80.0): return False
    if not (0.0 <= h <= 100.0):  return False
    return True

# --- Lectura concurrente con timeout y SIN CACHE (nuevo objeto por lectura) ---
def _worker_leer_dht(pin, idx, result_dict, name):
    """
    Hilo: crea un DHT22 nuevo, intenta hasta 2 lecturas rápidas.
    Si no hay sensor/pin desconectado, levantará excepción y devolvemos falla.
    """
    t = h = None
    ok = False
    for _ in range(2):
        dht = None
        try:
            dht = adafruit_dht.DHT22(pin, use_pulseio=False)
            t = dht.temperature
            h = dht.humidity
            if lectura_dht_valida(t, h):
                ok = True
                break
        except Exception:
            pass
        finally:
            try:
                if dht is not None:
                    dht.exit()
```

```
            except Exception:
                pass
        time.sleep(0.2)
    result_dict[idx] = (ok, t if ok else None, h if ok else None, name)


def leer_todos_los_sensores_con_timeout():
    threads = []
    results = {}
    for i, (nombre_pin) in enumerate(SENSORES):
        nombre, pin = nombre_pin
        th = threading.Thread(target=_worker_leer_dht, args=(pin, i, results, nombre), daemon=True)
        th.start()
        threads.append(th)

    deadline = time.time() + LECTURA_TIMEOUT_S
    for th in threads:
        remaining = deadline - time.time()
        if remaining > 0:
            th.join(remaining)

    for i, (nombre, _) in enumerate(SENSORES):
        if i not in results:
            results[i] = (False, None, None, nombre)
    return results  # idx -> (ok, t, h, nombre)

# ========= Tareas =========
def tarea_lecturas_y_hr():
    global _ultimo_print
```

```
ultimo_log = 0.0
csv_path = ruta_csv_actual(datetime.now())
preparar_csv(csv_path)

while True:
    ciclo_inicio = time.time()
    ahora = datetime.now()

    # rotación CSV
    nuevo_path = ruta_csv_actual(ahora)
    if nuevo_path != csv_path:
        csv_path = nuevo_path
        preparar_csv(csv_path)

    # 1) Leer sensores en paralelo con timeout y sin cache
    resultados = leer_todos_los_sensores_con_timeout()

    # 2) Preparar HR, CSV y datos para imprimir
    regs = [0]*12
    fila_csv = [ahora.isoformat(sep=" ", timespec="seconds")]
    lecturas_print = []  # (nombre, t_ok|None, h_ok|None)

    for i in range(len(SENSORES)):
        ok, t, h, nombre = resultados[i]
        idxT = 2*i
        idxH = idxT + 1

        if ok:
            # reset fallas y actualiza último válido
```

```
            fallas_consecutivas[nombre] = 0
            last_good_x10[nombre] = (x10(t), x10(h))
            last_good_flt[nombre] = (float(f"{t:.1f}"), float(f"{h:.1f}"))

            # Modbus/CSV usan el valor actual (válido)
            regs[idxT], regs[idxH] = last_good_x10[nombre]
            fila_csv.extend([f"T{i+1}: {t:.1f}", f"Hum{i+1}: {h:.1f}"])

            # Print muestra el valor real
            lecturas_print.append((nombre, t, h))

        else:
            # falla este ciclo
            fallas_consecutivas[nombre] += 1

            # CMD: SIEMPRE 0/0 cuando falla el ciclo
            lecturas_print.append((nombre, None, None))

            # Modbus/CSV: solo 0/0 si superó umbral; si no, mantener último válido
            if fallas_consecutivas[nombre] >= FALLAS_CONSEC_PARA_ENCERAR:
                regs[idxT], regs[idxH] = (0, 0)
                fila_csv.extend([f"T{i+1}: 0.0", f"Hum{i+1}: 0.0"])
            else:
                # Mantener último válido (si no hay histórico, será 0/0 por defecto)
                lt_x10, lh_x10 = last_good_x10[nombre]
                lt_f, lh_f = last_good_flt[nombre]
                regs[idxT], regs[idxH] = (lt_x10, lh_x10)
                fila_csv.extend([f"T{i+1}: {lt_f:.1f}", f"Hum{i+1}: {lh_f:.1f}"])
```

```
        # 3) Escribir HRs
        with ds_lock:
            context[UNIT_ID].setValues(3, 0, regs)

        # 4) Print agrupado (CMD: 0/0 si falla en el ciclo)
        if time.time() - _ultimo_print >= INTERVALO_PRINT_S:
            ok_count = sum(1 for _, t_ok, h_ok in lecturas_print if t_ok is not None and h_ok is not None)
            fail = len(lecturas_print) - ok_count
            print(f"[{ahora:%Y-%m-%d %H:%M:%S}] Lecturas DHT (ok={ok_count}, fail={fail})", flush=True)
            for j, (nombre, t_ok, h_ok) in enumerate(lecturas_print, start=1):
                t_str = f"{t_ok:.1f}°C" if t_ok is not None else "0.0°C"
                h_str = f"{h_ok:.1f}%"  if h_ok is not None else "0.0%"
                fc = fallas_consecutivas[nombre]
                extra = f" (fallas={fc})" if (t_ok is None or h_ok is None) else ""
                print(f"  HR{2*(j-1):02d}/HR{2*(j-1)+1:02d} <- {nombre}: T={t_str} | H={h_str}{extra}", flush=True)
            print("[HR] ->", regs, flush=True)

            with ds_lock:
                coils = context[UNIT_ID].getValues(1, 0, count=4)
            coils_str = " | ".join([f"C{i}={int(coils[i])}" for i in range(4)])
            print(f"[COILS] <- {coils_str}", flush=True)
            print("[MODBUS] Cliente conectado" if _conectado else "[MODBUS] Esperando conexión de cliente...", flush=True)
            print("-"*60, flush=True)
            _ultimo_print = time.time()

        # 5) CSV
```

```
        if time.time() - ultimo_log >= PERIODO_LOG_S:
            with open(csv_path, "a", newline="") as f:
                csv.writer(f).writerow(fila_csv)
            ultimo_log = time.time()

        # 6) Sleep adaptativo
        elapsed = time.time() - ciclo_inicio
        rest = max(0.0, PERIODO_LECTURA_S - elapsed)
        time.sleep(rest)

def tarea_coils_a_gpio():
    while True:
        with ds_lock:
            coils = context[UNIT_ID].getValues(1, 0, count=max(8, len(SALIDAS_GPIO)))
        for coil_idx, bcm in SALIDAS_GPIO.items():
            activo = bool(coils[coil_idx]) if coil_idx < len(coils) else False
            if RELES_ACTIVOS_EN_BAJO:
                GPIO.output(bcm, GPIO.LOW if activo else GPIO.HIGH)
            else:
                GPIO.output(bcm, GPIO.HIGH if activo else GPIO.LOW)
        time.sleep(0.1)

# ========= Main =========
def main():
    try:
        th1 = threading.Thread(target=tarea_lecturas_y_hr, daemon=True)
        th2 = threading.Thread(target=tarea_coils_a_gpio, daemon=True)
        th1.start()
        th2.start()
```

```
        print(f"[SRV] Modbus TCP escuchando en {SERVER_BIND_IP}:{SERVER_PORT} (UnitID={UNIT_ID})", flush=True)
        StartTcpServer(context, identity=identity, address=(SERVER_BIND_IP, SERVER_PORT), handler=ConnLoggerHandler)
    finally:
        GPIO.cleanup()


if __name__ == "__main__":
    main()
```

Integrantes: Angel Zumba - Miguel Mayorga

Antes de ese umbral, se mantiene el ÚLTIMO VALOR VÁLIDO.

```
    ModbusSlaveContext,
    ModbusServerContext,
    ModbusSequentialDataBlock,
    from pymodbus.server import StartTcpServer, ModbusConnectedRequestHandler  # >=3.x
    from pymodbus.server.sync import StartTcpServer, ModbusConnectedRequestHandler  # 2.5.x
    ("DHT1", board.D4),
    ("DHT2", board.D17),
    ("DHT3", board.D27),
    ("DHT4", board.D22),
    ("DHT5", board.D5),
    ("DHT6", board.D6),
PERIODO_LECTURA_S = 6.0     # duración objetivo del ciclo
PERIODO_LOG_S     = 300.0     # CSV
INTERVALO_PRINT_S = 6.0     # impresión agrupada
LECTURA_TIMEOUT_S = 5.0     # timeout por sensor (por ciclo)
FALLAS_CONSEC_PARA_ENCERAR = 7  # <-- cambia aquí: # de fallas seguidas para poner 0 en Modbus/CSV
SERVER_PORT    = 1502
```

```
UNIT_ID       = 1
    inicial = GPIO.HIGH if RELES_ACTIVOS_EN_BAJO else GPIO.LOW
    GPIO.setup(bcm, GPIO.OUT, initial=inicial)
    return int(round(v*10)) if v is not None else 0
store   = ModbusSlaveContext(hr=hr_block, co=co_block, zero_mode=False)
last_good_x10 = {nombre: (0, 0) for nombre, _ in SENSORES}   # último valor válido para Modbus (x10)
    def connectionMade(self):
        global _conectado
        super().connectionMade()
        _conectado = True
        try:
            host, port = self.request.getpeername()
            print(f"[CONN] Cliente conectado: {host}:{port}", flush=True)
        except Exception:
            print("[CONN] Cliente conectado (peername no disponible)", flush=True)

    def connectionLost(self, reason):
        global _conectado
        print(f"[DISC] Cliente desconectado: {reason}", flush=True)
        _conectado = False
        super().connectionLost(reason)
    idx = hoy.toordinal() // ROTACION_DIAS
    inicio_ord = idx * ROTACION_DIAS
    inicio = date.fromordinal(inicio_ord)
    fin = inicio + timedelta(days=ROTACION_DIAS - 1)
    return inicio, fin
    d0, _ = ventana_n_dias(dt.date())
    return os.path.join(CSV_DIR, f"secador_{d0:%d_%m_%Y}.csv")
```

```
nuevo = not os.path.exists(path)
if nuevo:
    with open(path, "a", newline="") as f:
        w = csv.writer(f)
        header = ["timestamp"]
        for i in range(1, 7):
            header += [f"T{i}", f"Hum{i}"]
        w.writerow(header)
if t is None or h is None: return False
if not (-40.0 <= t <= 80.0): return False
if not (0.0 <= h <= 100.0):  return False
return True
"""
Hilo: crea un DHT22 nuevo, intenta hasta 2 lecturas rápidas.
Si no hay sensor/pin desconectado, levantará excepción y devolvemos falla.
"""
t = h = None
ok = False
for _ in range(2):
    dht = None
    try:
        dht = adafruit_dht.DHT22(pin, use_pulseio=False)
        t = dht.temperature
        h = dht.humidity
        if lectura_dht_valida(t, h):
            ok = True
            break
    except Exception:
        pass
```

```
        finally:
            try:
                if dht is not None:
                    dht.exit()
            except Exception:
                pass
        time.sleep(0.2)
    result_dict[idx] = (ok, t if ok else None, h if ok else None, name)
    threads = []
    results = {}
    for i, (nombre_pin) in enumerate(SENSORES):
        nombre, pin = nombre_pin
        th = threading.Thread(target=_worker_leer_dht, args=(pin, i, results, nombre), daemon=True)
        th.start()
        threads.append(th)

    deadline = time.time() + LECTURA_TIMEOUT_S
    for th in threads:
        remaining = deadline - time.time()
        if remaining > 0:
            th.join(remaining)

    for i, (nombre, _) in enumerate(SENSORES):
        if i not in results:
            results[i] = (False, None, None, nombre)
    return results  # idx -> (ok, t, h, nombre)
    global _ultimo_print
    ultimo_log = 0.0
```

```
csv_path = ruta_csv_actual(datetime.now())
preparar_csv(csv_path)

while True:
    ciclo_inicio = time.time()
    ahora = datetime.now()

    # rotación CSV
    nuevo_path = ruta_csv_actual(ahora)
    if nuevo_path != csv_path:
        csv_path = nuevo_path
        preparar_csv(csv_path)

    # 1) Leer sensores en paralelo con timeout y sin cache
    resultados = leer_todos_los_sensores_con_timeout()

    # 2) Preparar HR, CSV y datos para imprimir
    regs = [0]*12
    fila_csv = [ahora.isoformat(sep=" ", timespec="seconds")]
    lecturas_print = []  # (nombre, t_ok|None, h_ok|None)

    for i in range(len(SENSORES)):
        ok, t, h, nombre = resultados[i]
        idxT = 2*i
        idxH = idxT + 1

        if ok:
            # reset fallas y actualiza último válido
            fallas_consecutivas[nombre] = 0
```

```
            last_good_x10[nombre] = (x10(t), x10(h))
            last_good_flt[nombre] = (float(f"{t:.1f}"), float(f"{h:.1f}"))

            # Modbus/CSV usan el valor actual (válido)
            regs[idxT], regs[idxH] = last_good_x10[nombre]
            fila_csv.extend([f"T{i+1}: {t:.1f}", f"Hum{i+1}: {h:.1f}"])

            # Print muestra el valor real
            lecturas_print.append((nombre, t, h))

        else:
            # falla este ciclo
            fallas_consecutivas[nombre] += 1

            # CMD: SIEMPRE 0/0 cuando falla el ciclo
            lecturas_print.append((nombre, None, None))

            # Modbus/CSV: solo 0/0 si superó umbral; si no, mantener último válido
            if fallas_consecutivas[nombre] >= FALLAS_CONSEC_PARA_ENCERAR:
                regs[idxT], regs[idxH] = (0, 0)
                fila_csv.extend([f"T{i+1}: 0.0", f"Hum{i+1}: 0.0"])
            else:
                # Mantener último válido (si no hay histórico, será 0/0 por defecto)
                lt_x10, lh_x10 = last_good_x10[nombre]
                lt_f, lh_f = last_good_flt[nombre]
                regs[idxT], regs[idxH] = (lt_x10, lh_x10)
                fila_csv.extend([f"T{i+1}: {lt_f:.1f}", f"Hum{i+1}: {lh_f:.1f}"])

    # 3) Escribir HRs
```

```
        with ds_lock:
            context[UNIT_ID].setValues(3, 0, regs)

        # 4) Print agrupado (CMD: 0/0 si falla en el ciclo)
        if time.time() - _ultimo_print >= INTERVALO_PRINT_S:
            ok_count = sum(1 for _, t_ok, h_ok in lecturas_print if t_ok is not None and h_ok is not None)
            fail = len(lecturas_print) - ok_count
            print(f"[{ahora:%Y-%m-%d %H:%M:%S}] Lecturas DHT (ok={ok_count}, fail={fail})", flush=True)
            for j, (nombre, t_ok, h_ok) in enumerate(lecturas_print, start=1):
                t_str = f"{t_ok:.1f}°C" if t_ok is not None else "0.0°C"
                h_str = f"{h_ok:.1f}%"  if h_ok is not None else "0.0%"
                fc = fallas_consecutivas[nombre]
                extra = f" (fallas={fc})" if (t_ok is None or h_ok is None) else ""
                print(f"  HR{2*(j-1):02d}/HR{2*(j-1)+1:02d} <- {nombre}: T={t_str} | H={h_str}{extra}", flush=True)
            print("[HR] ->", regs, flush=True)

            with ds_lock:
                coils = context[UNIT_ID].getValues(1, 0, count=4)
            coils_str = " | ".join([f"C{i}={int(coils[i])}" for i in range(4)])
            print(f"[COILS] <- {coils_str}", flush=True)
            print("[MODBUS] Cliente conectado" if _conectado else "[MODBUS] Esperando conexión de cliente...", flush=True)
            print("-"*60, flush=True)
            _ultimo_print = time.time()

        # 5) CSV
        if time.time() - ultimo_log >= PERIODO_LOG_S:
```

```
            with open(csv_path, "a", newline="") as f:
                csv.writer(f).writerow(fila_csv)
            ultimo_log = time.time()


        # 6) Sleep adaptativo
        elapsed = time.time() - ciclo_inicio
        rest = max(0.0, PERIODO_LECTURA_S - elapsed)
        time.sleep(rest)
    while True:
        with ds_lock:
            coils = context[UNIT_ID].getValues(1, 0, count=max(8, len(SALIDAS_GPIO)))
        for coil_idx, bcm in SALIDAS_GPIO.items():
            activo = bool(coils[coil_idx]) if coil_idx < len(coils) else False
            if RELES_ACTIVOS_EN_BAJO:
                GPIO.output(bcm, GPIO.LOW if activo else GPIO.HIGH)
            else:
                GPIO.output(bcm, GPIO.HIGH if activo else GPIO.LOW)
        time.sleep(0.1)
    try:
        th1 = threading.Thread(target=tarea_lecturas_y_hr, daemon=True)
        th2 = threading.Thread(target=tarea_coils_a_gpio, daemon=True)
        th1.start()
        th2.start()
        print(f"[SRV] Modbus TCP escuchando en {SERVER_BIND_IP}:{SERVER_PORT}
(UnitID={UNIT_ID})", flush=True)
        StartTcpServer(context, identity=identity, address=(SERVER_BIND_IP,
SERVER_PORT), handler=ConnLoggerHandler)
    finally:
        GPIO.cleanup()
    main()
```

Apéndice E

Diseño de tablero

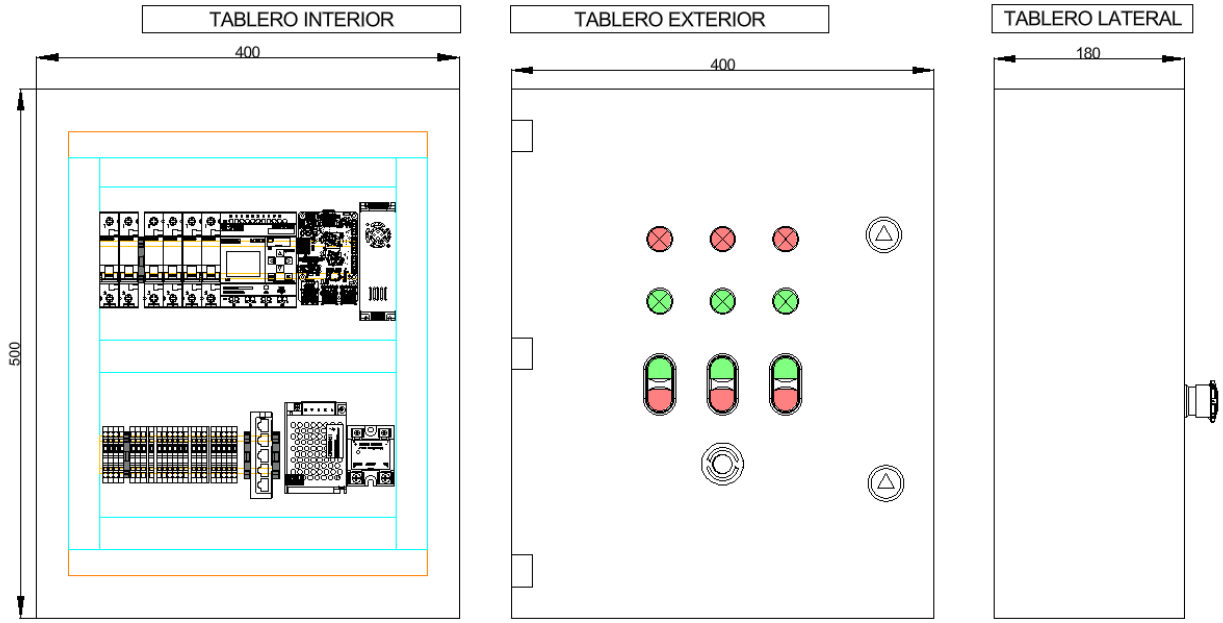
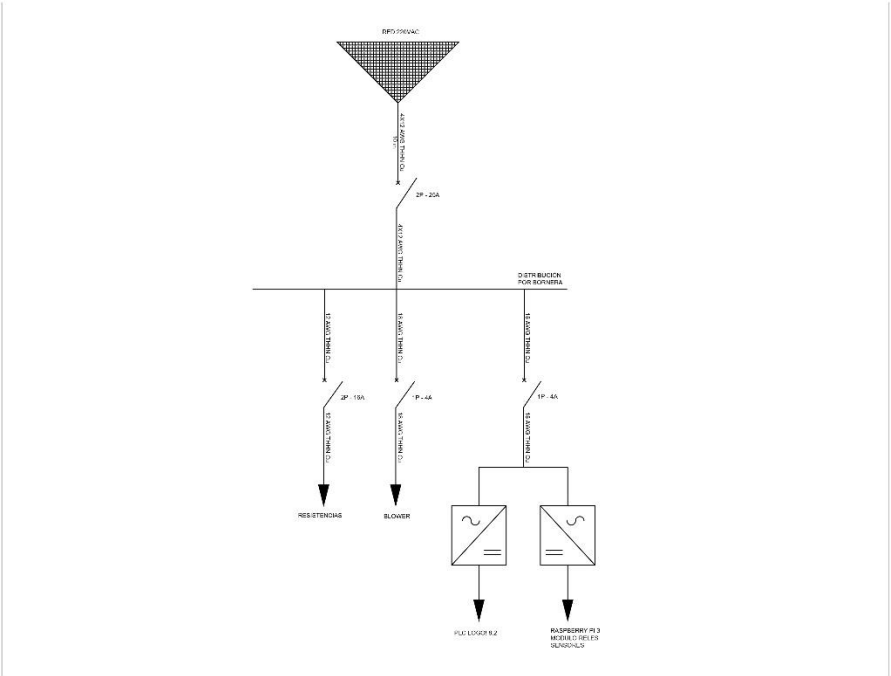


Diagrama Unifilar



Apéndice F.

Finalización del Proyecto