

Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

Diseño de un sistema mecatrónico para el monitoreo de temperatura y humedad
en una planta empacadora de camarón

Proyecto Integrador

Previo la obtención del Título de:

Ingeniera en Mecatrónica

Presentado por:

Melissa Lissette Banchón Valdiviezo

Guayaquil - Ecuador

Año: 2023

Dedicatoria

El presente proyecto lo dedico a mi familia, especialmente a mis dos ángeles de cuatro patas, a quienes les prometí graduarme, trabajar duro y regalarles el mundo entero, pero ahora su mundo es el cielo. Los amé, amo y seguiré amando hasta el final. Gracias por todo.

Agradecimientos

Mi más sincero agradecimiento a mis padres por apoyarme a seguir este largo camino. A mi abuela por haberme ayudado en innumerables ocasiones y seguirme ayudando a pesar de los años.

A mi familia en general por estar siempre a mi lado.

A mis amigos por sus palabras de aliento.

A mi jefe por haberme dado la oportunidad de desarrollar este proyecto.

A mis compañeros de trabajo por compartir su sabiduría conmigo.

Declaración Expresa

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; Melissa Lissette Banchón Valdiviezo doy mi 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”.



Autor

Evaluadores

Efraín Terán Calle, M.Sc.

Profesor de la materia

Marcelo Fajardo Pruna, Ph.D.

Profesor tutor

Resumen

El camarón es el segundo producto de mayor exportación del Ecuador. La medición y control de temperatura y humedad es un punto clave para obtener de un producto inocuo de alta calidad. Los métodos actuales de toma de datos son poco eficaces, dando paso a errores. El presente proyecto contempla el diseño de un sistema mecatrónico para el monitoreo y análisis de temperatura y humedad para una planta camaronera, diseñando los circuitos, protecciones físicas, bases de datos e interfaz para la consulta de datos. Se analizó termodinámicamente el área para determinar la cantidad y ubicación de los equipos. Una Raspberry Pi 4 y ESP32 con sus respectivos UPS y sensores de temperatura y humedad DHT22 fueron incluidos. Se consideraron diversos componentes electrónicos para la fabricación de la PCB del UPS emisor. La programación de los circuitos se desarrolló de manera exitosa permitiendo una transmisión eficaz de datos. Se logró la visualización de datos en tiempo real de manera sencilla por medio de la interfaz, así como la consulta por filtros. El sistema propuesto permite realizar el monitoreo de temperatura y humedad con su respectiva visualización de datos, listo para entrar a fase de pruebas en una planta procesadora de camarón.

Palabras clave: IIOT (Internet Industrial de las Cosas), condiciones ambientales, simulación, programación, diseño.

Abstract

Shrimp is Ecuador's second largest export product. The measurement and control of temperature and humidity is a key point to obtain a safe and high-quality product. Current data collection methods are inefficient and lead to errors. This project includes the design of a mechatronic system for monitoring and analyzing temperature and humidity for a shrimp plant designing circuits, physical protections, databases and interfaces for data consultation. The area was analyzed thermodynamically to determine the quantity and location of the equipment. A Raspberry Pi 4 and ESP32 were included with their respective UPS and DHT22 temperature and humidity sensors. Various electronic components were considered for the manufacture of the UPS emitter PCB. The programming of the circuits was successfully developed allowing efficient data transmission. The visualization of data in real time was achieved easily through the interface, as well as querying by filters. The proposed system allows temperature and humidity monitoring with its respective data visualization, ready to enter the testing phase in a shrimp processing plant.

Keywords: IIOT (Industrial Internet of Things), environmental conditions, simulation, programming, design.

Índice general

Resumen.....	I
Abstract.....	II
Índice general.....	III
Abreviaturas.....	V
Simbología.....	VI
Índice de tablas	IX
1. Capítulo 1.....	1
1.1 Introducción	2
1.2 Descripción del problema	7
1.3 Justificación del problema	9
1.4 Objetivos.....	1
1.4.1 Objetivo general.....	1
1.4.2 Objetivos específicos	1
1.5 Marco teórico	1
1.6 Estado del arte.....	5
2. Capítulo 2.....	8
2.1 Requerimientos de diseño	9
2.2 Selección de la alternativa de solución	10
2.2.1 Propuestas de alternativas de solución.....	10
2.2.2 Análisis de alternativas	12
2.2.3 Selección de alternativa	15
2.3 Proceso de diseño.....	17

2.4	Diseño conceptual.....	18
2.5	Proceso de diseño.....	18
2.5.1	Análisis termodinámico	18
2.5.2	Determinación del número y ubicación de equipos.....	19
2.5.3	Diseño de circuitos eléctricos y electrónicos.....	20
2.5.4	Estudio energético de los circuitos	21
2.5.5	Elección de fuentes energéticas de respaldo.....	21
2.5.6	Diseño de las cajas protectoras de circuitos.....	23
2.5.7	Programación de microcontroladores para envío y recepción de datos.....	24
2.5.8	Diseño de la base de datos	25
2.5.9	Creación de interfaz de usuario	27
3.	Capítulo 3.....	28
3.1	Resultados y análisis	29
3.1.1	Simulaciones de análisis termodinámico	29
3.1.2	Establecimiento de circuitos.....	32
3.1.3	Consumos y baterías	34
3.1.4	Diseño de protecciones físicas.....	34
3.1.5	Programación	36
3.1.6	Análisis de costos.....	39
4.	Capítulo 4.....	42
4.1	Conclusiones y recomendaciones	43
4.1.1	Conclusiones.....	43
4.1.2	Recomendaciones	44
	Referencias.....	45
	Apéndices.....	48

Abreviaturas

ESPOL	Escuela Superior Politécnica del Litoral
OEC	Observatory of Economic Complexity
BAP	Best Aquaculture Practices
GSA	Global Seafood Alliance
FODA	Fortalezas, Oportunidades, Debilidades, Amenazas
CPS	Cyber-physical system
IIoT	Industrial Internet of Things
IDE	Integrated Development Environment
RH	Relative Humidity
USB	Universal Serial Bus
QR	Quick Response
MQTT	Message Queuing Telemetry Transport
SQL	Structured Query Language
UPS	Uninterruptable Power Supply
PCB	Printed Circuit Board
RTD	Resistance Temperature Detector
PLA	Polylactic Acid

Simbología

°C	Grados Centígrados
m	Metro
mm	Milímetro
\$	Dólar
h	Hora
A	Amperios
Ah	Amperios Hora
mA	Miliamperios
mAh	Miliamperios Hora
V	Voltios
Ohm	Ohmios
W	Watts
k	Kilo
u	Micro

Índice de figuras

Figura 1.1 Comparación entre los valores de exportación de los mercados petrolero, bananero y camaronero en Ecuador.....	2
Figura 1.2 Países destino de la exportación de camarón ecuatoriano.....	3
Figura 1.3 Proceso de producción en una planta de camarón.....	4
Figura 1.4 Proceso de recepción del camarón	5
Figura 1.5 Bandas de empacado con camarón clasificado por tallas.....	5
Figura 1.6 Proceso de salida de producto congelado y despacho en camiones	6
Figura 1.7 Avances tecnológicos en los sistemas de monitoreo de temperatura	7
Figura 1.8 Aspectos que abarca la Industria 4.0 (Internet de las cosas)	2
Figura 1.9 Placa Arduino UNO R3.....	3
Figura 1.10 Placa Raspberry Pi 4 Modelo B.....	3
Figura 1.11 Placas ESP32 y módulo Xbee	4
Figura 1.12 Sensor de humedad BME280 de la marca BOSCH.	5
Figura 1.13 Funcionamiento del registrador de temperatura y humedad Elitech RCW-800 Data Logger.....	6
Figura 2.1 Metodología de diseño para el sistema de monitoreo de temperatura y humedad	17
Figura 2.2 Diseño conceptual del sistema de monitoreo	18
Figura 2.3 Interfaz de herramienta CFD	19
Figura 2.4 Modelo de la planta de proceso	20
Figura 2.5 Esquema en Proteus de Raspberry Pi 4	20
Figura 2.6 Esquema en Proteus del circuito emisor de datos (ESP32 + DHT22)	21
Figura 2.7 Circuito de carga de batería con sistema de corte automático.....	22
Figura 2.8 Circuito convertidor y estabilizador de voltaje	23
Figura 2.9 Interfaz de herramienta Inventor 2024	24

Figura 2.10 Creación y configuración de la base de datos local.....	25
Figura 2.11 Creación de la tabla de almacenamiento de datos	26
Figura 2.12 Creación de base de datos remota y tabla correspondiente	26
Figura 2.13 Diseño de la interfaz gráfica para la visualización de datos.....	27
Figura 3.1 Simulación de temperatura en ambiente de planta procesadora.....	29
Figura 3.2 Ubicación de los sensores de temperatura y humedad	30
Figura 3.3 Ubicación recomendada de los dispositivos en la planta de proceso	31
Figura 3.4 Vista frontal de la ubicación sugerida de los dispositivos.....	31
Figura 3.5 Circuito físico de Raspberry Pi 4 con su respectivo UPS y cargador	32
Figura 3.6 Diseño de PCB (ESP32 + DHT22 + UPS).....	33
Figura 3.7 Modelo 3D de PCB del circuito emisor	33
Figura 3.8 Caja protectora para circuito receptor	35
Figura 3.9 Caja protectora para circuito emisor.....	35
Figura 3.10 Pruebas de recepción de datos en la Raspberry Pi 4	36
Figura 3.11 Pruebas de envío de información hacia la base de datos local y remota.....	37
Figura 3.12 Recepción de información en la base de datos local	37
Figura 3.13 Recepción de información en la base de datos remota.....	38
Figura 3.14 Conexión de la interfaz con la base de datos.....	38

Índice de tablas

Tabla 1.1 Comparación entre métodos manuales y automáticos de monitoreo.....	1
Tabla 2.1 Principales requerimientos de diseño a cumplir	9
Tabla 2.2 Distribución de requerimientos en secciones mecatrónicas	10
Tabla 2.3 Establecimiento de criterios de selección	12
Tabla 2.4 Placas disponibles en el mercado.....	13
Tabla 2.5 Sensores disponibles en el mercado.....	14
Tabla 2.6 Matriz de decisión.....	15
Tabla 3.1 Componentes para la construcción de la PCB	39
Tabla 3.2 Costos de placas de desarrollo, sensores y accesorios.....	40
Tabla 3.3 Costos de materiales para instalación	40
Tabla 3.4 Costos de diseño y fabricación	41

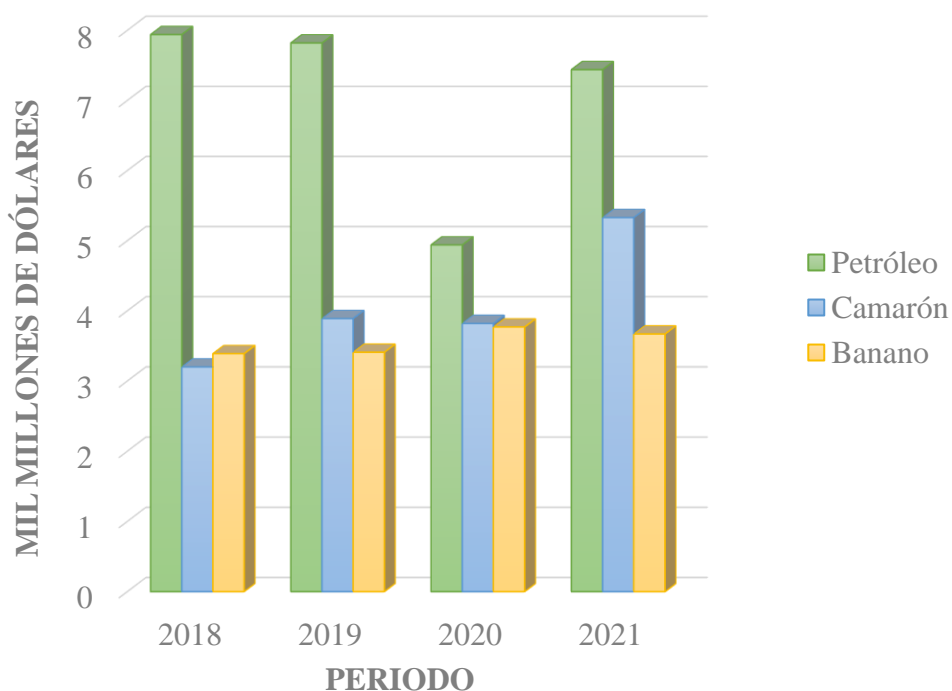
Capítulo 1

1.1 Introducción

Ecuador es un país petrolero, tradicionalmente conocido por sus exportaciones de banano, cacao, flores, entre otros. A este grupo se le suma el camarón, producto cuyo auge ha sido notable en los últimos años. Según datos recogidos por el Observatorio de Complejidad Económica, OEC por sus siglas en inglés, existen fluctuaciones visibles en el nivel de exportaciones realizadas de cada producto [1]. El crecimiento de las exportaciones de camarón en comparación a otros productos expresados en miles de millones de dólares anuales en un periodo de 4 años puede ser visualizado en la Figura 1.1.

Figura 1.1

Comparación entre los valores de exportación de los mercados petrolero, bananero y camarero en Ecuador



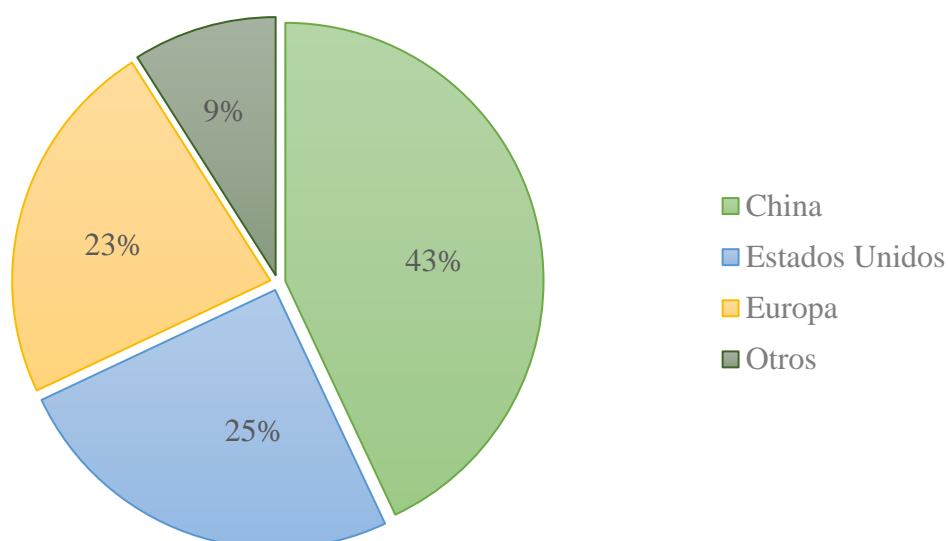
Se puede observar que el mercado del petróleo es predominante, a pesar de ello, el camarón ha tenido un crecimiento gradual en el periodo del 2018 a 2021, convirtiéndose en el primer producto no petrolero de exportación de Ecuador. Existe un decrecimiento de \$70 millones en el año 2020, que responde a las limitaciones presentadas durante el inicio de la

pandemia del COVID-19 [2]. A pesar de ello, en el 2021 logró incrementar las cifras en \$1.51 mil millones de dólares.

En su mayoría, las exportaciones se dirigen a China, Estados Unidos y Europa, cuyas regulaciones en cuanto al manejo de alimentos son sumamente estrictas. Los porcentajes mostrados en la Figura 1.2 corresponden al análisis realizado por la OEC respecto al año 2021 [1].

Figura 1.2

Países destino de la exportación de camarón ecuatoriano



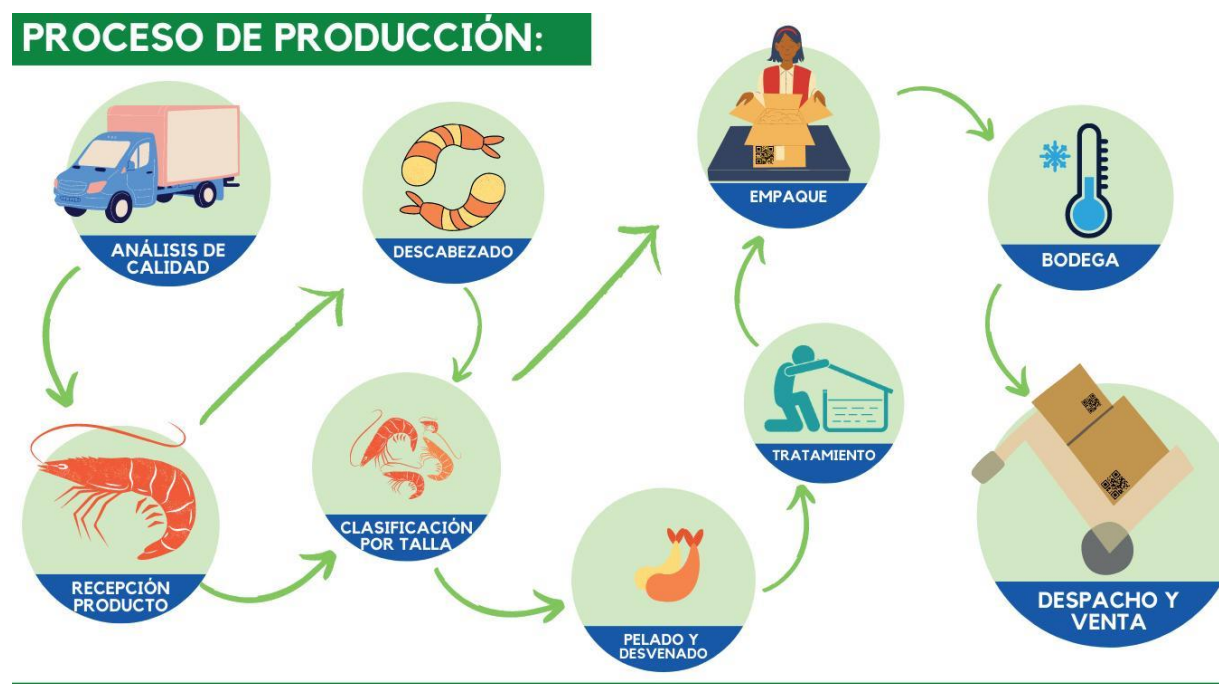
Existe una gran variedad de instituciones internacionales que certifican la calidad e inocuidad del producto, los cuales se conocen como estándares. Uno de los estándares más conocidos es el BAP (Best Aquaculture Practices), donde en el punto 3.17.8 indica “... La instalación deberá mantener temperaturas ambientales de refrigeración y/o congelación que inhiban el crecimiento bacteriano, crecimiento de patógenos y/o desarrollo de toxinas” [3].

Según la Alianza Global de Mariscos (Global Seafood Alliance), regulador del estándar BAP, la temperatura del camarón, desde la recepción hasta su llegada al sitio de almacenamiento de congelados, debe permanecer debajo de los 3°C. Para mantener dicha

temperatura la planta de procesamiento debe tener una temperatura menor a 12°C. Además, los productos se deben almacenar en salas de temperatura negativa, siendo estos de menos de -18°C [4].

Figura 1.3

Proceso de producción en una planta de camarón



La Figura 1.3 [5], presenta el proceso estándar en las planta de camarón. Comienza con un análisis de calidad y se pasa a la recepción del producto, como se observa en la Figura 1.4. En esta etapa se limpia el camarón y se elimina cualquier elemento que no sea camarón, tales como pedazos de madera, piedras y otros animales [4].

Figura 1.4

Proceso de recepción del camarón

**Figura 1.5**

Bandas de empaqueo con camarón clasificado por tallas



Dependiendo del tipo de producto pasa al proceso de descabezado o directamente a la clasificación por talla. Luego, se puede realizar el pelado y desvenado del camarón y un tratamiento adicional o pasar directamente al empaque, como se observa en la Figura 1.5 [4].

Finalmente, es almacenado en una bodega con temperaturas de congelamiento hasta el momento del despacho en los respectivos camiones, como se puede ver en la Figura 1.6 [4].

Figura 1.6

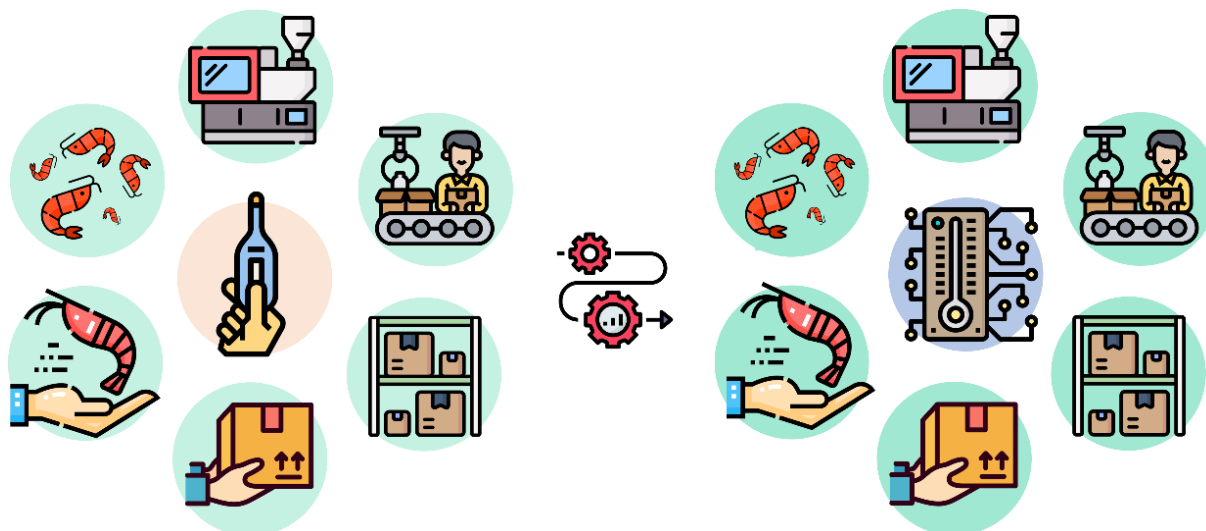
Proceso de salida de producto congelado y despacho en camiones



Mediante el control de temperatura se busca evitar enfermedades como el botulismo, producido por la bacteria *Clostridium botulinum*, la cual se presenta como una enfermedad poco común, pero con consecuencias que pueden llegar a ser mortales [6]. La aparición de este tipo de bacterias en el producto perjudica a todos los involucrados en el proceso, desde la planta procesadora, los puntos de compra, y por supuesto, el consumidor final. En caso de una eventualidad de este tipo, se retira dicho lote del mercado, conocido internacionalmente como “recall”. Luego, comienza un proceso de investigación por parte de agentes internacionales, que, dependiendo de la gravedad del evento, pueden llegar al punto de prohibiciones de venta futuras [7].

Figura 1.7

Avances tecnológicos en los sistemas de monitoreo de temperatura



De manera general, el proceso de producción del camarón se podría resumir en los puntos mostrados en la Figura 1.7: recepción, clasificación, procesamiento, empaque, almacenamiento y despacho. Se muestra que existe un monitoreo manual de las temperaturas y se espera una migración a métodos tecnológicos sin modificar las etapas del proceso actual.

1.2 Descripción del problema

Las empresas necesitan actualizar sus métodos de medición y control de temperatura y humedad para crear las condiciones óptimas que permitirán un proceso fluido y continuo en cuanto a recepción, procesamiento, empaque y almacenamiento de productos. En las plantas de procesamiento de mariscos, la temperatura juega un papel importante en cuanto a calidad del producto. Factores como la proliferación de bacterias, descomposición del marisco y pérdida de peso se controlan por medio de la temperatura en proceso [4].

Por otro lado, la formación de óxido, pérdida de grasas lubricantes y acumulación de humedad en dispositivos eléctricos son situaciones que ocurren en sitios con gran humedad ambiental. Este tipo de ocurrencias generan daños en máquinas y sus componentes provocando atrasos en la producción y gastos elevados en reparaciones [8]. Adicionalmente, el exceso de

humedad puede comprometer el bienestar de los trabajadores al provocar pisos resbalosos y, por ende, potencialmente peligrosos.

Por lo tanto, es necesario implementar un sistema capaz de mostrar los datos de temperatura y humedad de manera interactiva en tiempo real. Entre las características más importantes se encuentran: que permita analizar el histórico de los datos, visualizar fluctuaciones recientes en gráficas de variable vs tiempo y la generación de reportes.

Un sistema de este tipo facilita la información solicitada por los agentes reguladores que avalan las certificaciones nacionales e internacionales, como las mencionadas anteriormente: BAP, FDA, etc. Así mismo, permite constatar el grado de calidad de procesamiento de los productos, afianzando sus ventas dentro y fuera del país. Adicionalmente, permite un mejor control de las temperaturas mediante el manejo de los equipos de refrigeración respectivos.

Tomando en cuenta los aspectos mencionados con anterioridad, la seguridad en el almacenamiento de datos es sumamente importante para evitar filtraciones o corrupción de información. La base de datos a utilizar debe permitir relacionar la información de los sensores con su ubicación, para la identificación de zonas problemáticas o de mayor atención en el proceso. La colocación de dispositivos debe tomar en cuenta la extensión de la planta de proceso, las zonas o secciones identificadas y las propias condiciones ambientales para asegurar la mayor vida útil posible y fiabilidad de los datos.

El presente proyecto está orientado a implementar un sistema de recolección de información que permita monitorizar y analizar la temperatura y humedad durante el procesamiento del camarón, desde su recepción hasta su salida. Este monitoreo incrementa notablemente la calidad, confiabilidad y trazabilidad del producto, asegurando el cumplimiento de los requerimientos del cliente de destino.

1.3 Justificación del problema

Los métodos actuales de toma y análisis de datos se realizan de manera manual dando paso a errores humanos en cuanto a medición y generación de informes para la interpretación de la información. El tiempo invertido en anotaciones y traspaso manual de datos podría ser reducido de gran manera, permitiendo tomar acciones rápidas en momentos de fluctuaciones significativas que podrían comprometer la calidad del producto e inocuidad del proceso. Un análisis FODA (Fortalezas, Oportunidades, Debilidades, Amenazas) de los sistemas de monitoreo manuales y automáticos se presenta en la Tabla 1.1.

Se denota que los métodos manuales presentan más debilidades y amenazas que los métodos automáticos. Además, las debilidades y amenazas de los métodos automáticos pueden ser minimizados creando características como utilización de baterías y sistemas de seguridad adecuados para controlar el acceso del personal involucrado. Los sistemas manuales presentan más dificultades en mitigar los problemas y dependen de gran manera del criterio y formación humana.

En consecuencia, el presente proyecto debe realizar un análisis dimensional de la planta de proceso identificando y limitando las zonas del proceso para determinar la cantidad de dispositivos de toma de datos a utilizar y su disposición en el espacio. El sistema eléctrico adecuado, protecciones y estructuras de acuerdo con las condiciones ambientales. Un software adecuado para la visualización y análisis rápido de los datos necesarios. De esta manera se solucionan los problemas por tomas de datos erróneos, comunicación poco eficiente entre personal el personal involucrado en el monitoreo y control de temperatura y tiempo de ingreso de datos en plataformas como Excel.

Tabla 1.1*Comparación entre métodos manuales y automáticos de monitoreo*

	Manual	Automático
Fortalezas	<ul style="list-style-type: none">• Análisis in-situ de la condición actual.	<ul style="list-style-type: none">• Análisis de varios puntos simultáneos.• Datos accesibles en tiempo real en distintas locaciones.• Respaldo de los datos de manera digital.
Oportunidades	<ul style="list-style-type: none">• Bajo costo	<ul style="list-style-type: none">• Mejora de la calidad del producto.• Personal dedicado a tareas críticas.
Debilidades	<ul style="list-style-type: none">• Poca eficiencia en la toma de datos.• Necesita de una persona y equipo adecuado.• Datos no accesibles en tiempo real y limitados en alcance.• Los equipos más precisos necesitan calibración.	<ul style="list-style-type: none">• Dependiente de una fuente de alimentación.• Necesita mantenimiento para toma de datos óptima.
Amenazas	<ul style="list-style-type: none">• Error humano en la anotación de datos.• Documentos susceptibles a daños irre recuperables.• Dificultad en el reconocimiento de equipos no calibrados.• Corrupción de datos.	<ul style="list-style-type: none">• Corrupción de la base de datos.• Filtración de información.

1.4 Objetivos

1.4.1 *Objetivo general*

Diseñar un sistema mecatrónico de monitoreo de temperatura y humedad en una planta camaronera para la conservación de la calidad e inocuidad del producto mediante el uso de sensores, bases de datos, interfaces de usuario intuitivas y protecciones adecuadas para los circuitos.

1.4.2 *Objetivos específicos*

1. Establecer la configuración del sistema para el monitoreo de puntos críticos por medio de un análisis de dimensiones e identificación de zonas temperatura elevada.
2. Diseñar el circuito electrónico y protecciones físicas adecuadas para la recolección fiable de datos de temperatura y humedad considerando los factores ambientales propios de la planta de procesamiento.
3. Generar una base de datos capaz de recolectar los valores de los puntos de control para permitir la consulta e ingreso de datos evitando la saturación del servidor.
4. Desarrollar una interfaz de usuario que facilite el proceso de consulta y análisis de datos de manera fácil e interactiva permitiendo mayor agilidad en la resolución de problemas.

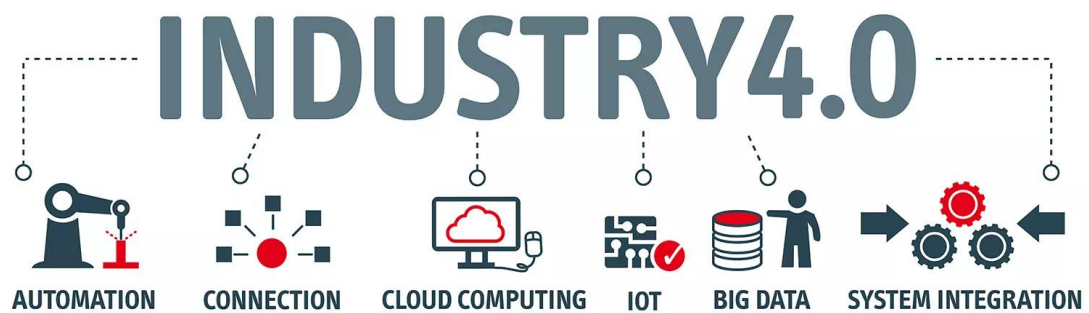
1.5 Marco teórico

En la historia han existido tres revoluciones industriales, caracterizadas principalmente por la transición en cuanto a herramientas utilizadas y la automatización de los procesos realizados. A partir del año 2011 se incluyó una cuarta revolución industrial, a partir del cual surge el término Industria 4.0, donde se incluye la utilización de sistemas ciberfísicos (Cyber-physical systems , CPSs, por sus siglas en inglés), para el incremento de características como la eficiencia, productividad, seguridad y transparencia [9]. Los sistemas ciberfísicos se

caracterizan por el monitoreo y manipulación de elementos y procesos reales, es decir, una interacción que genera resultados físicos [10]. Los elementos de la industria 4.0 se pueden visualizar de mejor manera en la Figura 1.8 [11].

Figura 1.8

Aspectos que abarca la Industria 4.0 (Internet de las cosas)



El Internet Industrial de las cosas, en inglés IIoT (*Industrial Internet of Things*), según Cisco, “es un ecosistema de dispositivos, sensores, aplicaciones y equipo de *networking* que trabaja en conjunto para recolectar, monitorear y analizar información de operaciones industriales” [12]. Este tipo de tratamiento de información automatizado permite que las empresas puedan observar con detalle el comportamiento de un determinado sistema, mejorando el proceso de solución de problemas y el desarrollo de actividades de mantenimiento de los equipos involucrados.

Existen distintos dispositivos que permiten la adquisición, análisis y muestra de datos. Una de las tecnologías más conocidas es Arduino, cuya placa se muestra en la Figura 1.9 [13], es un hardware *opensource* que consiste en una placa con su propio IDE capaz de recolectar y procesar datos; además, envía señales en respuesta a parámetros predefinidos mediante programación a través del entorno de desarrollo integrado [14].

Figura 1.9

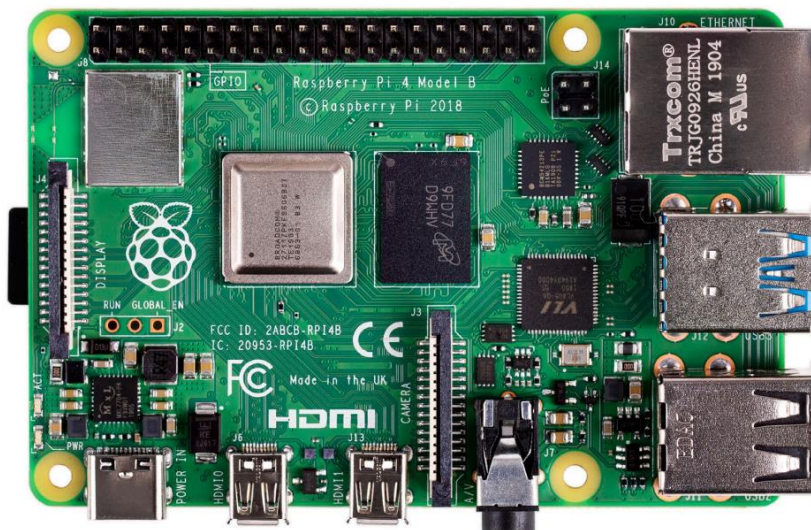
Placa Arduino UNO R3



Se pueden mencionar otras tecnologías tales como Raspberry Pi, mostrada en la Figura 1.10 [15], que es una placa más completa, capaz de albergar un sistema operativo para la creación de bases de datos o programas de mayor capacidad de procesamiento.

Figura 1.10

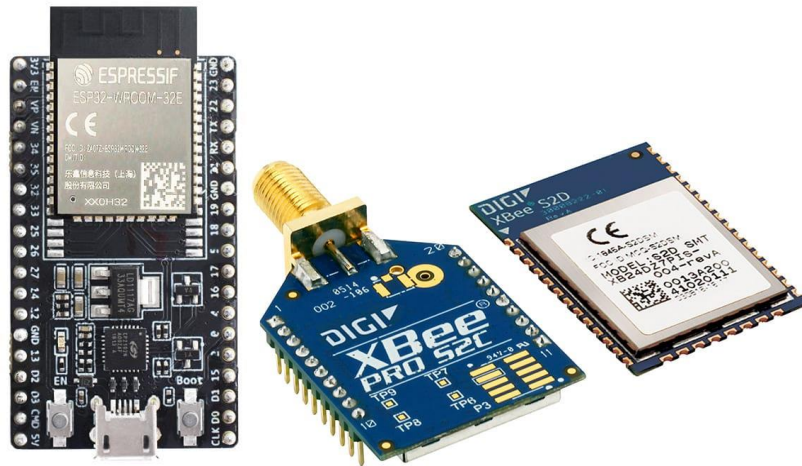
Placa Raspberry Pi 4 Modelo B



Otros dispositivos para la recolección y transmisión de datos son, ESP32 [16] y ZigBee, caracterizados por sus capacidades inalámbricas [17]. Dichos elementos se visualizan en la Figura 1.11.

Figura 1.11

Placas ESP32 y módulo Xbee



En el análisis de las condiciones ambientales de un ecosistema, se utilizan variables como la temperatura, humedad, presión atmosférica, entre otros.

Un sensor de temperatura permite obtener el nivel de calor o frío presente en un ambiente o cuerpo. Estos pueden ser de contacto o sin contacto. Dentro de la primera subdivisión se puede diferenciar entre accionamiento mecánico y eléctrico. Centrándose en el aspecto eléctrico, existen tres tipos de dispositivos para recolección de información de temperatura: termistores, termocuplas y detectores de resistencia-temperatura (RTDs). Se diferencian por el material del que están hechos y la manera en que se obtienen los datos de la variable. Algunos se basan en la medición del cambio de resistencia en la corriente eléctrica como es el caso de los termistores, mientras que los detectores de resistencia-temperatura registran la temperatura midiendo la resistencia del elemento RTD [18].

Para la medición de humedad existen sensores capacitivos, resistivos y piezoresistivos, incluso se incluyen variedades de sensores ópticos, gravimétricos, capacitivos, entre otros. Por un lado, se diferencian por su método de obtención de datos, los rangos de medición y precisión

de la lectura. Por otro lado, también se debe tener en cuenta el ambiente donde serán implementados para su elección [19]. Un ejemplo de estos sensores es el BME280, mostrado en la Figura 1.12 [20].

Figura 1.12

Sensor de humedad BME280 de la marca BOSCH.



1.6 Estado del arte

La toma y análisis de temperaturas y humedad ambiental es un tema existente en el país. Existen dispositivos inalámbricos importados que ofrecen servicios de registro de temperatura y descarga de datos, denominados Data logger. Un ejemplo de este tipo de dispositivos es la gama Elitech RC-61, RC-4HC, RCW-800, entre otros, proveniente de Estados Unidos. Se diferencian entre sí por la resolución de los datos, su interfaz de conexión, rangos de temperatura y humedad detectables.

El primero detecta un rango de -30°C a 70°C y 10% a 99% RH, el segundo un rango de -30°C a 60°C con opción a ampliación mediante un sensor externo siendo de -40°C a 85°C y un rango de humedad del 10% al 99%. Ambos se conectan con el software mediante USB y no poseen opción a protección por contraseñas. Utilizan baterías que duran 2 y 1 año respectivamente.

Figura 1.13

Funcionamiento del registrador de temperatura y humedad Elitech RCW-800 Data Logger



El último tiene un rango de temperatura de -40°C a $80^{\circ}\text{C} \pm 5^{\circ}\text{C}$ y 10% a $95\% \pm 5\%$. Como se observa en la Figura 1.13, cuenta con una conexión Wifi a la plataforma de la marca; el acceso a la plataforma móvil se realiza por medio del código QR del Data Logger adquirido. El periodo acceso y almacenamiento de datos se define por medio de planes estándar, avanzado y profesional con una duración de 30 días, 3 y 5 años respectivamente. La batería tiene una duración de 6 horas [21].

Un servicio más avanzado lo ofrece SensMax en Lituania para compañías de almacenamiento. Los parámetros de temperatura y humedad que maneja son del -20°C a $55^{\circ}\text{C} \pm 5^{\circ}\text{C}$ y 0% a $100\% \pm 3.5\%$ respectivamente. Proveen de sensores a prueba de agua y polvo con una batería de 5 años de duración. El software se conecta con internet para ser consultado de manera remota, con este tipo de conexión se permite la inclusión de hasta 250 sensores en rangos de entre 150 a 500 m usando repetidores de señal. Permite la visualización en tiempo real de los datos por medio de gráficas e indicadores, así como la inclusión de alertas [22].

En cuanto a proyectos relacionados realizados en el país, se encuentra el proyecto desarrollado por los ingenieros Willy Sandoya y Marco Mena en su trabajo de titulación “Diseño e implementación de un prototipo de monitoreo de temperatura, humedad y presión con comunicación vía radio e internet para mejorar la producción agrícola” [23]. En donde se utilizan microcontroladores ESP32, el protocolo MQTT, una base de datos en Heidi SQL y la creación de una página web para la consulta de los datos por medio de la plataforma Flatkit. Para la obtención de parámetros se utiliza el sensor BME280, que permite adquirir los datos de temperatura, humedad y presión atmosférica, con rangos de -40°C a 85°C en temperatura y una tolerancia de $\pm 3\%$ en la medición de humedad relativa.

Capítulo 2

En este capítulo se detalla el proceso de diseño en cuanto a la solución del problema planteado. Incluye la elección de elementos para obtención de temperatura y humedad, el análisis del entorno para la distribución de sensores, diseño de protecciones físicas para los circuitos eléctricos, selección de servidor y base de datos.

2.1 Requerimientos de diseño

Los requerimientos discutidos con el cliente y se resumen en la Tabla 2.1.

Tabla 2.1

Principales requerimientos de diseño a cumplir

Requerimiento	Detalle
Rango de temperatura	-30°C a 30°C
Rango de humedad	10% a 90% RH
Precisión	Alta
Tiempo de operación	24/7
Tipo de ambiente de operación	Altamente húmedo
Espacio de operación	54x12x2.5
Administración de información	Servidor con base de datos
Visualización	Interfaz gráfica accesible remotamente de manera segura.
Información	Gráficas en tiempo real, exportación a Excel con datos de un rango de tiempo determinado.

Los puntos centrales del proyecto estuvieron en el rango y capacidad de operación de los dispositivos en el espacio confinado establecido. Así mismo, se realizó cierto énfasis en la accesibilidad y facilidad de visualización de los datos estando fuera de la zona de estudio.

Estas características se organizaron por tipo de requerimiento. Se lograron identificar 3 secciones siendo estas: mecánico, electrónico e informático, como se muestra en la Tabla 2.2.

Tabla 2.2

Distribución de requerimientos en secciones mecatrónicas

Sección	Requerimiento
Mecánicos	<ul style="list-style-type: none"> • Diseño de caja protectora para los circuitos necesarios. • Análisis termodinámico del espacio donde se ubicarán los sensores.
Electrónicos	<ul style="list-style-type: none"> • Diseño de circuitos para obtención y transmisión de datos. • Elección de las fuentes y/o baterías correspondientes. • Transmisión de datos entre dispositivos.
Informáticos	<ul style="list-style-type: none"> • Elección y diseño de base de datos. • Creación de interfaz gráfica.

2.2 Selección de la alternativa de solución

Para la propuesta de opciones se exploraron las variedades de dispositivos disponibles en el mercado, considerando distintas características que podrían considerarse importantes para el desarrollo del proyecto dado el contexto presentado en el Capítulo 1 y los requerimientos expresados en el punto 2.1.

2.2.1 Propuestas de alternativas de solución

Los principales puntos que considerar fueron las placas y sensores a utilizar, así como el tipo de transmisión de datos. Se plantearon cuatro alternativas.

Alternativa 1: Utilización de PLC, sensores de temperatura y humedad por separado. Conexión totalmente cableada entre PLC y sensores. Almacenamiento de datos de respaldo en el propio PLC y conexión con el servidor remoto.

Alternativa 2: Utilización de PLC, un solo sensor para los datos de temperatura y humedad. Conexión totalmente cableada entre PLC y sensores. Almacenamiento de datos de respaldo en el propio PLC y conexión con el servidor remoto.

Alternativa 3: Utilización de ESP32 como punto recolector de datos y Raspberry Pi como central de recepción de datos, almacenamiento temporal y envío de datos al servidor remoto. Utilización un solo sensor para la temperatura y humedad. Conexión inalámbrica entre ESP32 y Raspberry Pi.

Alternativa 4: Utilización de Arduino como punto recolector de datos y Raspberry Pi como central de recepción de datos, almacenamiento temporal y envío de datos al servidor remoto. Utilización un solo sensor para la temperatura y humedad. Conexión cableada entre Arduino y Raspberry Pi.

Los criterios que se utilizaron para la selección de alternativas son los siguientes:

Costo inicial es la inversión que se debe realizar para la adquisición de hardware y software necesarios para la implementación.

Precisión es el grado de desviación que pueden tener los sensores respecto a las lecturas que realizan. Se busca minimizar dicho valor para una mejor toma de datos.

Durabilidad es el tiempo que se estima dure el equipo bajo las condiciones descritas en el problema. Se busca una gran resistencia a ambientes severos.

Costo de mantenimiento en cuanto a la disponibilidad de repuestos y la facilidad al poder reemplazarlos y que el sistema siga funcionando correctamente.

Estética para la presentación del equipo dentro de la planta de proceso. Los dispositivos deben poder mantenerse limpios y verse como parte del ambiente sin destacar negativamente.

Requerimiento energético para determinar el tipo de conexiones energéticas que se necesitan, así como las baterías a utilizar para asegurar una larga duración.

Se realizó la Tabla 2.3 donde se asignaron los pesos en relación con el grado de importancia en el contexto del sistema de monitoreo planteado.

Tabla 2.3

Establecimiento de criterios de selección

Criterio	Ranking	Peso relativo	Porcentaje
Precisión	1	10	36%
Durabilidad	2	7	25%
Costo de mantenimiento	3	5	17%
Costo inicial	4	3	11%
Estética	5	2	7%
Requerimiento de energía	6	1	4%
Total		28	100%

2.2.2 Análisis de alternativas

Para realizar una mejor elección de la alternativa a implementar, se recabó información de ciertas placas y sensores disponibles. Para la comparación de las placas se consideraron cuatro opciones, las cuales se muestran en la Tabla 2.4. Para sensores que permitan obtener datos de temperatura y humedad simultáneamente, las opciones se muestran en la Tabla 2.5.

En la Tabla 2.4, la cantidad de pines de entrada da una idea de la cantidad de sensores que pueden ser conectados a dicha placa en caso de necesitar adaptarse. La capacidad de almacenamiento da mayor seguridad en momentos de fallos energéticos generales para que el equipo siga recolectando información. El tipo de comunicación indica la flexibilidad en cuanto a implementaciones alámbricas o inalámbricas. La temperatura de operación debe permitir al equipo funcionar en bajas temperaturas. El consumo ayuda en la elección de fuentes de energía o baterías más adelante.

Tabla 2.4

Placas disponibles en el mercado


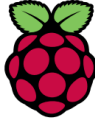




				
	LOGO!			
	PLC [24]	Raspberry Pi [15]	Arduino [13]	ESP32 [16]
Pines de entrada	8 entradas	28x pines GPIO	14x pines GPIO	34x pines GPIO
Almacenamiento	32 GB	64 GB	32 GB	32 GB
máximo	(SD Card)	(SD Card)	(SD Card)	(MicroSD Mod)
Comunicación	<ul style="list-style-type: none"> • Puerto Ethernet • Servidor Web 	<ul style="list-style-type: none"> • Bluetooth 5.0 • USB • Ethernet/WiFi 	<ul style="list-style-type: none"> • USB Integrables: <ul style="list-style-type: none"> • Ethernet/WiFi 	<ul style="list-style-type: none"> • Bluetooth 4.2 • USB • WiFi
Temp. de operación	0°C a 55°C	0 a 50°C	-40°C a 85°C	-40°C a 125°C
Consumo	0.15 A	3 A	1 A	0.2 A
Dimensiones [mm]	71.5 x 60 x 90	56.5 x 85.6 x 11	68.6 x 53.4	18 x 25.5 x 2
Precio	\$155	\$125	\$18	\$13

Tabla 2.5*Sensores disponibles en el mercado*

	 SHT30	 DHT22
Rango	T: -40°C a 125°C H: 0% a 100%	T: -40°C a 80°C H: 0% a 100%
Precisión	T: ±0.2 °C H: ±2%	T: ±0.5 °C H: ±2%
Ambiente	No indica	Sensibilidad afectada por vapores químicos
Consumo	6V 1A	3.3-5.5 VDC 2.5 mA
Distancia de transmisión	2m	100 m
Dimensiones	80 x 10 x 10 mm	58.8 x 26.7 x 13.8 mm
Compatibilidad	Raspberry Arduino PLC	Raspberry Arduino ESP32
Precio	\$45	\$10

En la Tabla 2.5 el rango y precisión son importantes para una toma de datos confiable. La sección ambiental da una idea de la durabilidad que tendría en determinadas condiciones ambientales. El consumo y distancia de transmisión permiten determinar características del circuito eléctrico. Las dimensiones permiten la creación de las protecciones físicas necesarias para salvaguardar la integridad del sistema. La compatibilidad permite conocer la flexibilidad del sensor en cuanto a conexiones con variedad de dispositivos.

Se plantearon dos alternativas para el almacenamiento y accesibilidad de la información respecto a la base de datos: local o remoto. Las principales diferencias entre estos servidores son precio, usabilidad de espacio, seguridad y costo de mantenimiento.

2.2.3 Selección de alternativa

Con los criterios seleccionados y una pequeña contextualización de las alternativas propuestas, se realizó la matriz de decisión expresada en la Tabla 2.6.

Tabla 2.6

Matriz de decisión

Criterios	Precisión	Durabilidad	Costo de mantenimiento	Costo inicial	Estética	Requerimiento de energía	Total
Peso	10	7	5	3	2	1	28
	36%	25%	17%	11%	7%	4%	100%
Alternativa 1	6	6	7	4	5	7	5,9
Alternativa 2	6	6	6	5	5	7	5,9
Alternativa 3	7	7	6	7	8	5	6,8
Alternativa 4	7	5	4	7	7	6	6,0

Como se puede observar, la mejor opción es la alternativa número tres al haber recibido un mayor puntaje. Las alternativas 1 y 2 bajan de nivel en cuanto a estética por la robustez del PLC, provocando que la caja protectora a realizar deba ser más grande y vistosa. El costo de un PLC es elevado, mientras que las placas de desarrollo tienen precios más bajos, con cualidades similares. Las opciones que consideran una comunicación cableada a lo largo de un espacio de gran magnitud provocan problemas en cuanto a estética y extensión del circuito eléctrico, así como posibles pérdidas de datos debido al recorrido del cableado. Las opciones inalámbricas tienen mayores requerimientos energéticos por la constante comunicación y factores a considerar como las interferencias a causa de otros dispositivos circundantes.

2.3 Proceso de diseño

Figura 2.1

Metodología de diseño para el sistema de monitoreo de temperatura y humedad

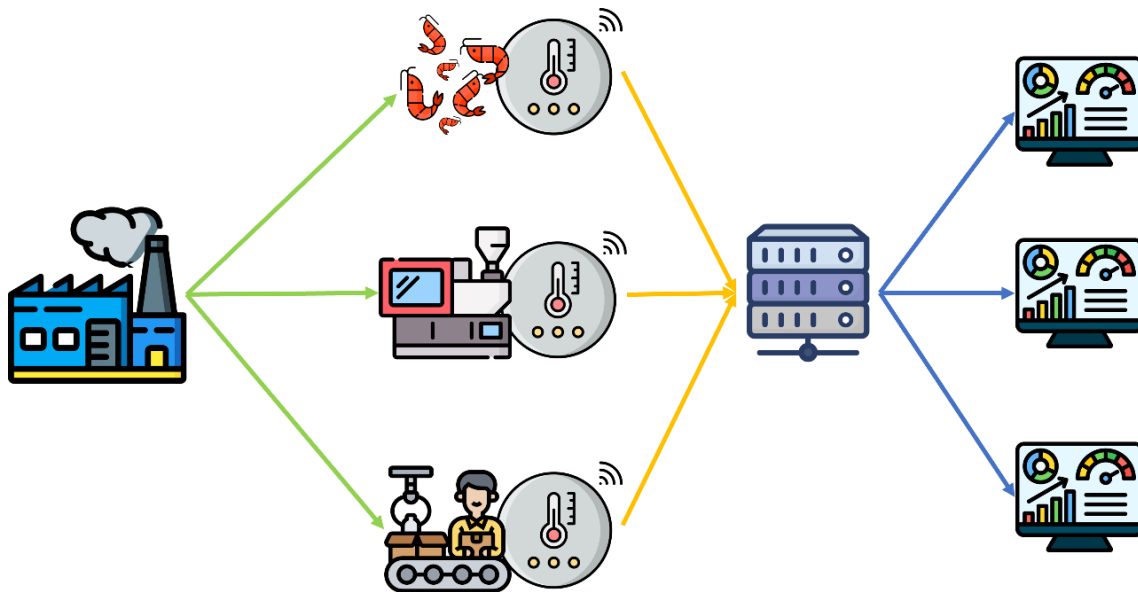


El proceso de diseño que se llevó a cabo durante el proyecto está descrito en la Figura 2.1, contemplando los tres aspectos principales del proyecto: mecánica, electrónica e informática.

2.4 Diseño conceptual

Figura 2.2

Diseño conceptual del sistema de monitoreo



Como se puede observar en la Figura 2.2, se pretende ubicar equipos de monitoreo en cada una de las etapas más críticas de la sección de proceso planteada, que va desde la recepción/clasificación, tratado y empaque. Estos datos deberán guardarse en un servidor y poder ser visualizados en distintos ordenadores.

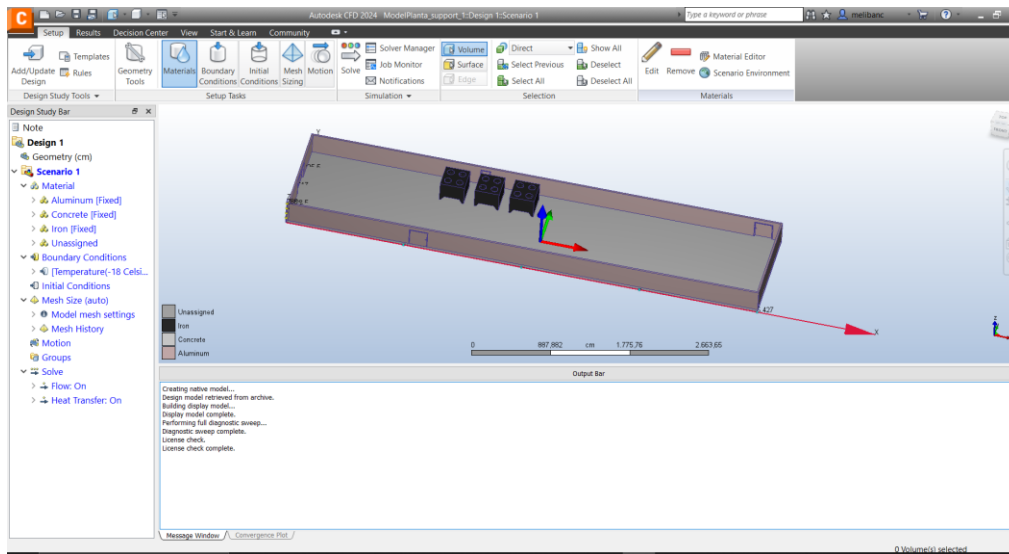
2.5 Proceso de diseño

2.5.1 Análisis termodinámico

Para el análisis termodinámico, se utilizó la herramienta Revit de Autodesk; se construyó el modelo con los datos de la Tabla 2.1. Entre algunas características que no figuran en la tabla mencionada, está el material de las paredes que están conformados por planchas aislantes de temperatura y la ubicación de los equipos generadores de frío. El modelo se puede ver en la Figura 2.3. Los procesos de recepción, clasificación, tratamiento y empaque se realizan en este espacio, tomando la figura de derecha a izquierda.

Figura 2.3

Interfaz de herramienta CFD



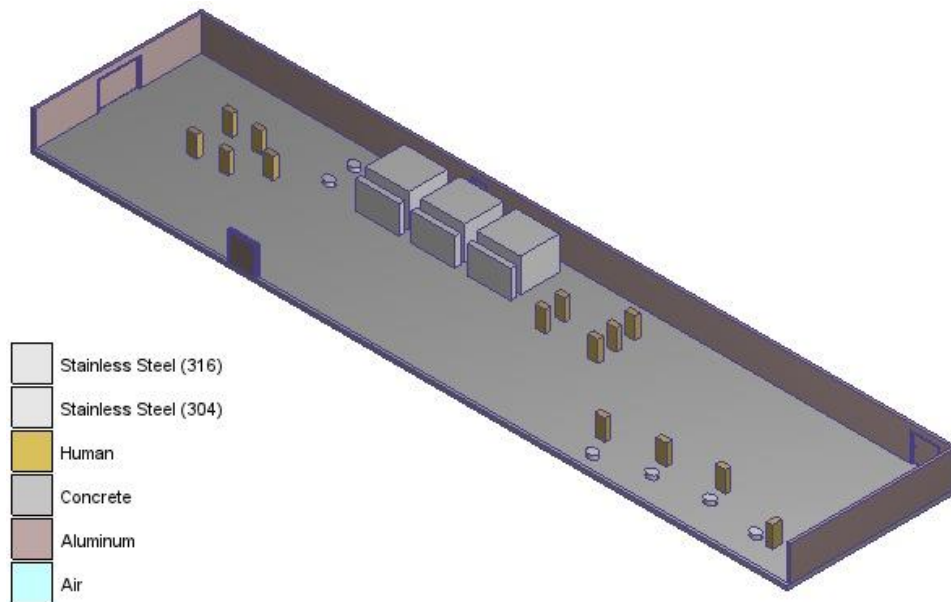
2.5.2 Determinación del número y ubicación de equipos

Los puntos de control se determinaron a partir de las secciones identificadas con anterioridad. Para establecer la ubicación de los equipos fue importante identificar los gradientes de temperatura, correspondientes a las variaciones de color en la simulación. Se consideraron varios elementos y sus respectivas temperaturas para incluir su aporte de calor dentro del proceso, como se muestra en la Figura 2.4. Los elementos de acero inoxidable corresponden a maquinaria, también se incluye el personal presente, el material de las paredes y piso siendo este aluminio y concreto respectivamente.

Adicionalmente, se consideró que el aire en las secciones cercanas al techo presenta una temperatura mayor debido a factores físicos, pues, el aire caliente al tener menor densidad tiende a subir. Por lo tanto, si estas temperaturas permanecen en el rango establecido, se tiene la seguridad de que las temperaturas al nivel del proceso son adecuadas. A partir de esta información se conoce que la altura a la que se recomendó ubicar los equipos no comprometerá significativamente la información mostrada.

Figura 2.4

Modelo de la planta de proceso

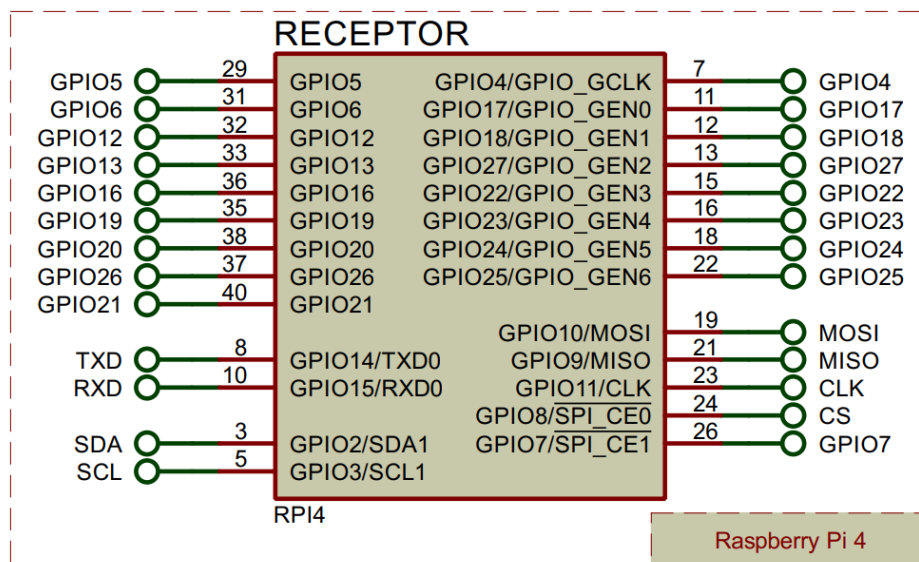


2.5.3 Diseño de circuitos eléctricos y electrónicos

Se consideraron dos tipos de circuitos. El primero es el receptor y base de datos temporal conformado por la Raspberry Pi mostrado en la Figura 2.5 y su respectivo UPS.

Figura 2.5

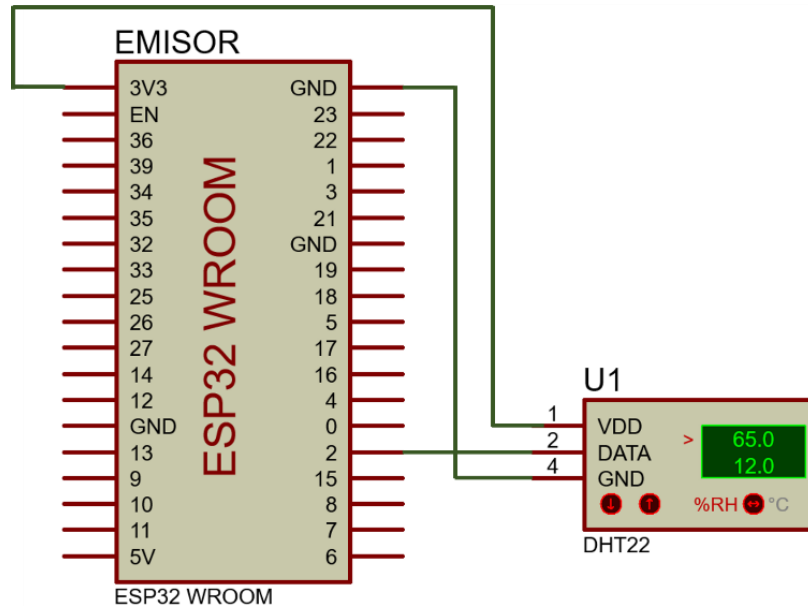
Esquema en Proteus de Raspberry Pi 4



El segundo circuito es el recolector y emisor de datos conformado por la placa ESP32, el sensor de temperatura y humedad DHT22 como se muestra en la Figura 2.6.

Figura 2.6

Esquema en Proteus del circuito emisor de datos (ESP32 + DHT22)



2.5.4 Estudio energético de los circuitos

Esta información es proporcionada en la Tabla 2.4 y Tabla 2.5, donde se mencionan los amperajes de consumo de cada uno de los circuitos y su voltaje de operación, así como la extensión de cableado y demás características técnicas.

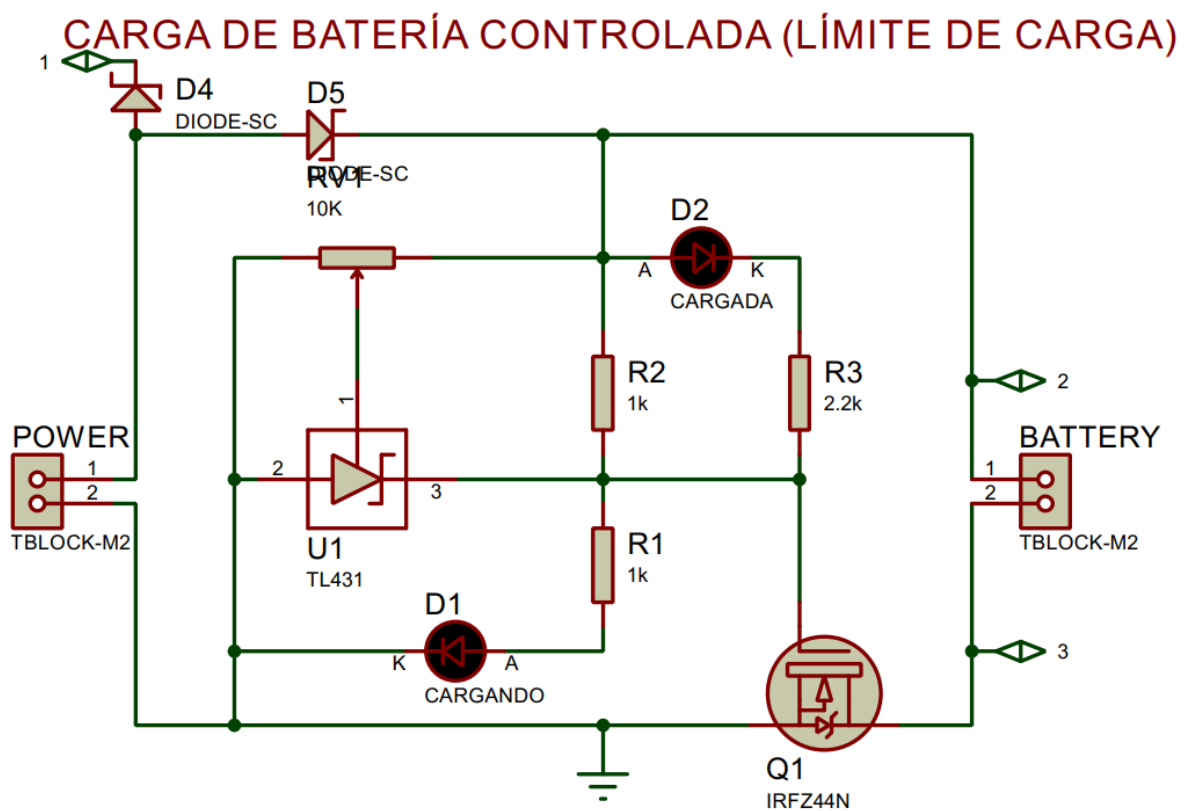
2.5.5 Elección de fuentes energéticas de respaldo

La elección de fuentes de energía de respaldo permite seguir obteniendo información en caso de algún corte energético repentino, por lo cual los equipos deben tener la capacidad de hacer el cambio entre la fuente fija y el respaldo sin perder información, así como tener una duración considerable hasta el restablecimiento de la energía. Tomando la información de la sección 2.5.4 y teniendo en cuenta que un corte energético tendría una duración máxima de 1 a 2 horas, se determinó que son necesarios los siguientes circuitos.

Un circuito que permita la carga de la batería de litio y que una vez cargada pare de suministrar corriente para evitar el sobrecalentamiento, daño o explosión de la batería. Se opta por el circuito de la Figura 2.7 diseñado con un sistema de corte en determinado voltaje de la batería, establecido mediante un potenciómetro de 10K Ohm, con una batería de 3.7V se prepara el corte en el máximo voltaje cargada que es 4.2V. Adicionalmente se estableció una sección en el circuito que permita el intercambio entre la fuente de energía directa y la batería.

Figura 2.7

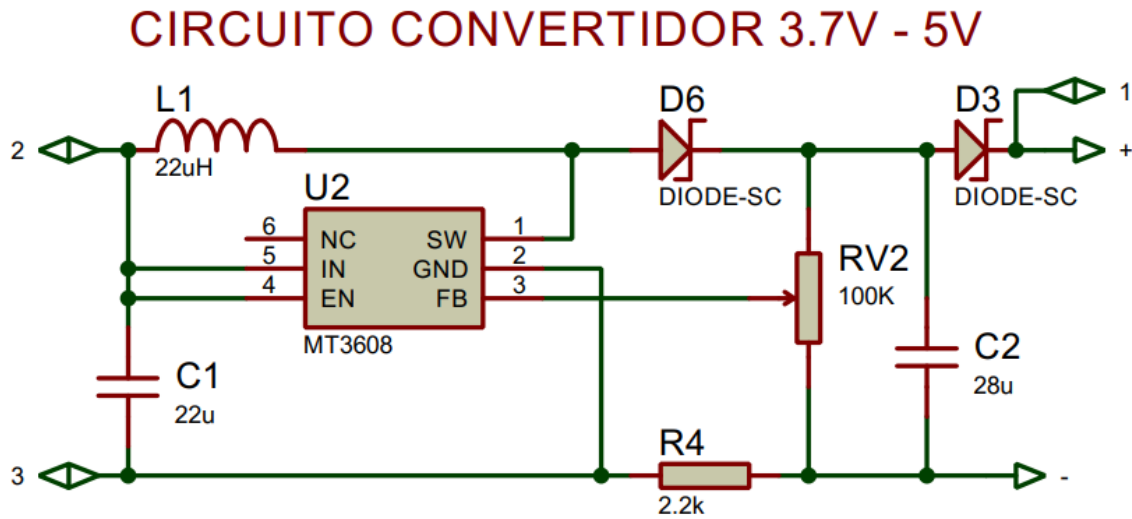
Circuito de carga de batería con sistema de corte automático



Se consideró un sistema capaz de elevar la tensión proveniente de la batería de 3.7V a 5V y que a su vez permita alimentar el circuito con un voltaje estable. En el diseño que se aprecia en la Figura 2.8 se utilizó un módulo MT3608 que permite regular y convertir el voltaje DC.

Figura 2.8

Circuito convertidor y estabilizador de voltaje



Finalmente, para definir la capacidad mínima de las baterías se calculó de la siguiente manera:

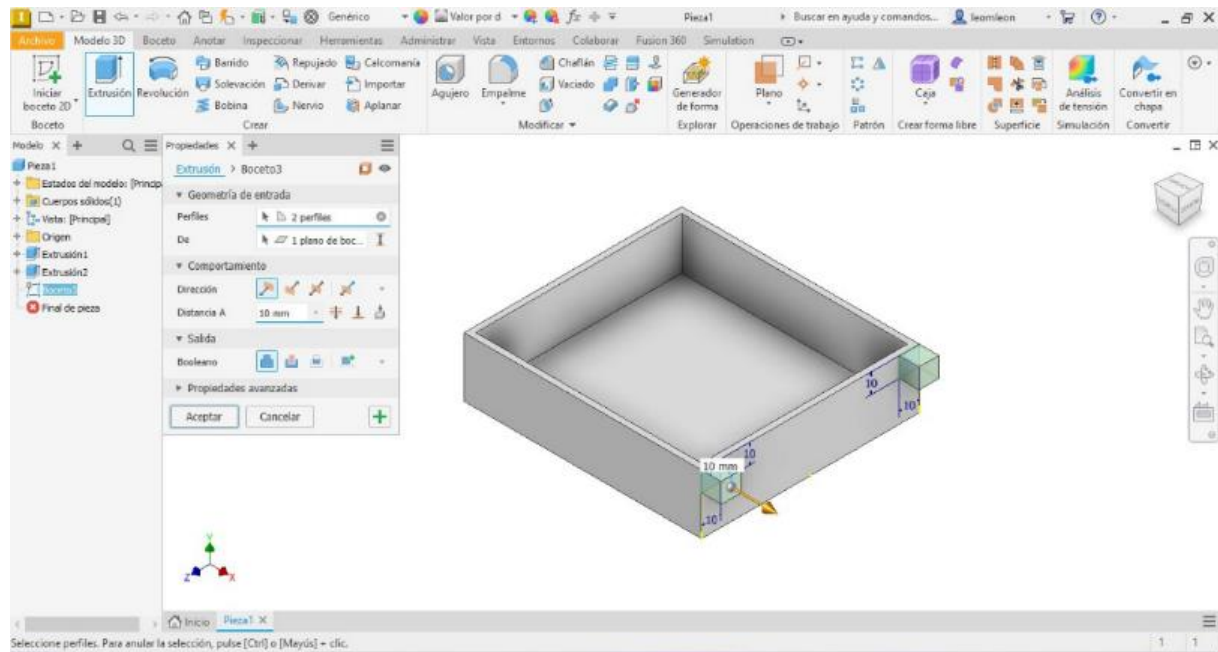
$$T = \frac{I_b \times V_b \times n \times e}{P} \quad (2.1)$$

Donde T es el tiempo de duración de la batería en horas, I_b es la capacidad de la batería en Ah, V_b es el voltaje de la batería, n el número de baterías a utilizar, e la eficiencia de la batería (en baterías de litio se encuentra entre el 95% y 98%. Finalmente, P es el consumo del circuito en W. Despejando para obtener los Ah de la batería del circuito emisor se obtiene la siguiente fórmula:

$$I_b = \frac{T \times P}{V_b \times n \times e} \quad (2.2)$$

2.5.6 Diseño de las cajas protectoras de circuitos

Para el circuito receptor se consideró la extensión total de la Raspberry Pi con el UPS y su cableado. Para el circuito emisor se consideraron las dimensiones de la PCB y un alojamiento para la batería. Los diseños se realizaron utilizando la herramienta Inventor de Autodesk como se muestra en la Figura 2.9.

Figura 2.9*Interfaz de herramienta Inventor 2024*

2.5.7 Programación de microcontroladores para envío y recepción de datos

Se realizó la programación de la ESP32 por medio de Arduino IDE, con las librerías "DHT.h", <WiFi.h> y <PubSubClient.h>, para la obtención de datos del sensor, la conexión a la red WiFi establecida para el intercambio de datos y la dinámica de envío y recepción de datos mediante el protocolo MQTT, respectivamente.

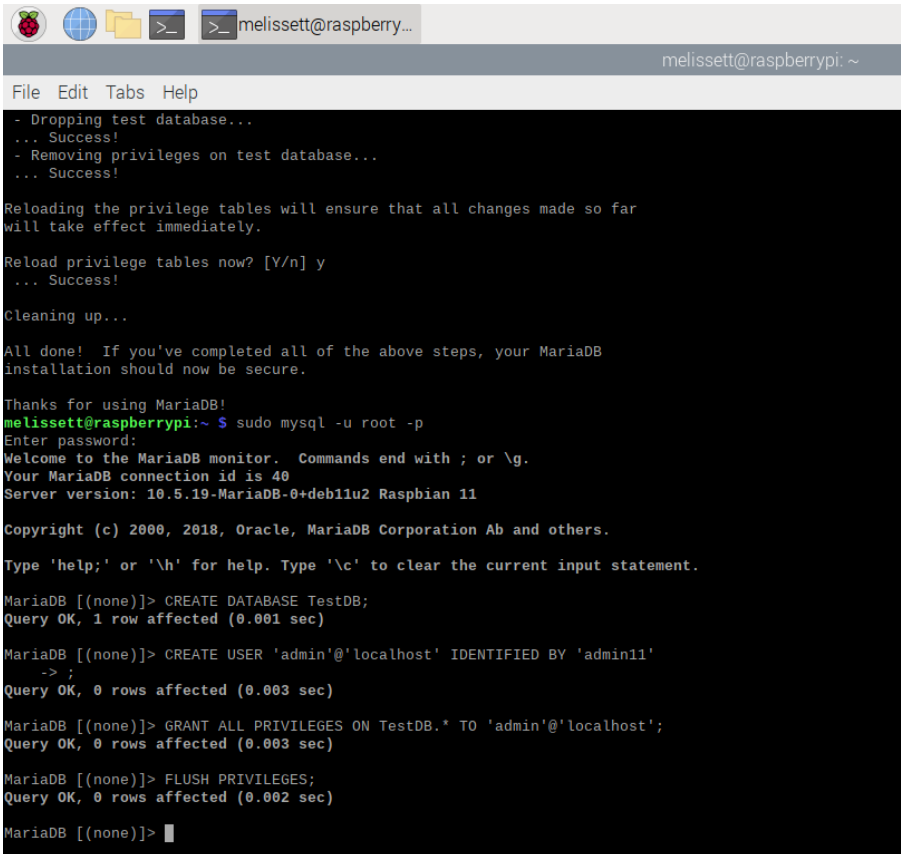
La programación de la Raspberry Pi 4 se realizó por medio de su interfaz gráfica, con la ayuda del software VNC para facilitar su acceso. El protocolo de comunicación utilizado entre los circuitos emisor y receptor fue el de Mosquitto o también conocido como MQTT. Se utilizó el lenguaje de programación Python para la recepción de los datos por medio de la librería paho-mqtt y el ingreso de los datos a la base de datos local y remota por medio de la librería mysql.connector.

2.5.8 Diseño de la base de datos

Para las bases de datos se utilizó mysql. La base de datos local se manejó por medio del terminal como se muestra en la Figura 2.10 y Figura 2.11. Para la base de datos remota se utilizó la plataforma phpmyadmin para las configuraciones correspondientes. En la Figura 2.12 se evidencia la creación de una tabla con la misma configuración que la base de datos local.

Figura 2.10

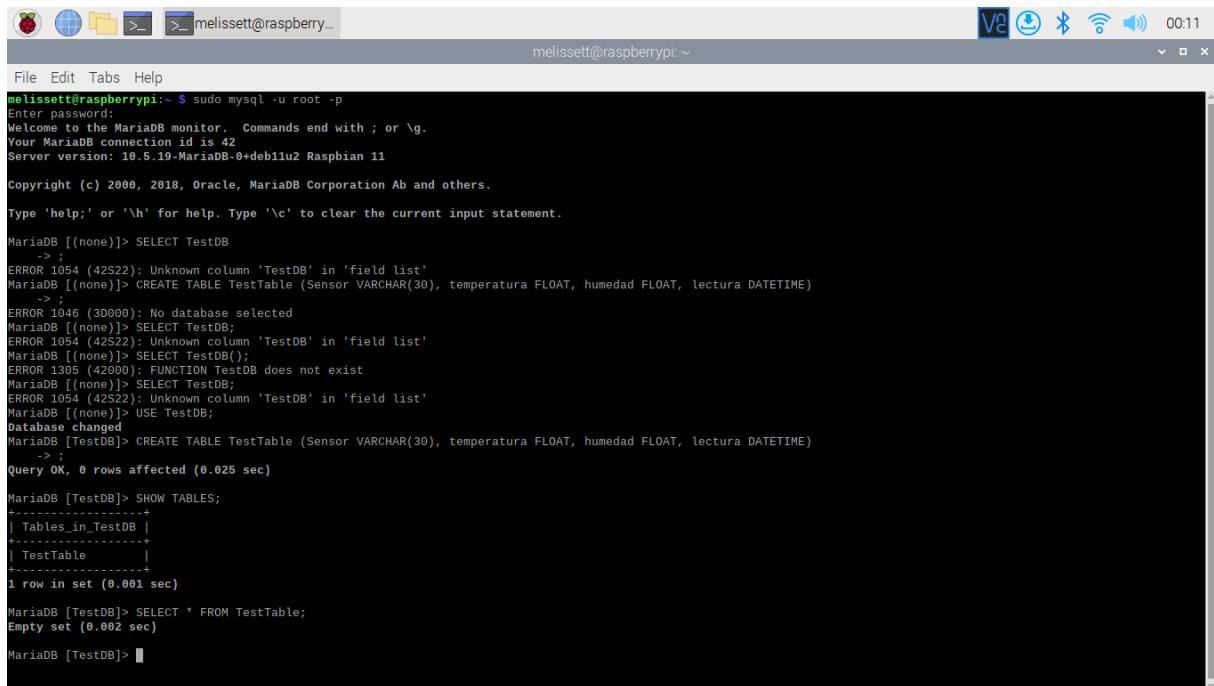
Creación y configuración de la base de datos local



```
melissett@raspberrypi: ~  
File Edit Tabs Help  
- Dropping test database...  
... Success!  
- Removing privileges on test database...  
... Success!  
Reloading the privilege tables will ensure that all changes made so far  
will take effect immediately.  
Reload privilege tables now? [Y/n] y  
... Success!  
Cleaning up...  
All done! If you've completed all of the above steps, your MariaDB  
installation should now be secure.  
Thanks for using MariaDB!  
melissett@raspberrypi:~$ sudo mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 40  
Server version: 10.5.19-MariaDB-0+deb11u2 Raspbian 11  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MariaDB [(none)]> CREATE DATABASE TestDB;  
Query OK, 1 row affected (0.001 sec)  
MariaDB [(none)]> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin11'  
-> ;  
Query OK, 0 rows affected (0.003 sec)  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON TestDB.* TO 'admin'@'localhost';  
Query OK, 0 rows affected (0.003 sec)  
MariaDB [(none)]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.002 sec)  
MariaDB [(none)]>
```

Figura 2.11

Creación de la tabla de almacenamiento de datos



```

melissett@raspberrypi:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 42
Server version: 10.5.19-MariaDB-0+deb11u2 Raspbian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT TestDB
->
ERROR 1054 (42S22): Unknown column 'TestDB' in 'field list'
MariaDB [(none)]> CREATE TABLE TestTable (Sensor VARCHAR(30), temperatura FLOAT, humedad FLOAT, lectura DATETIME)
->
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> SELECT TestDB;
ERROR 1054 (42S22): Unknown column 'TestDB' in 'field list'
MariaDB [(none)]> SELECT TestDB();
ERROR 1305 (42000): FUNCTION TestDB does not exist
MariaDB [(none)]> SELECT TestDB;
ERROR 1054 (42S22): Unknown column 'TestDB' in 'field list'
MariaDB [(none)]> USE TestDB;
Database changed
MariaDB [TestDB]> CREATE TABLE TestTable (Sensor VARCHAR(30), temperatura FLOAT, humedad FLOAT, lectura DATETIME)
->
Query OK, 0 rows affected (0.025 sec)

MariaDB [TestDB]> SHOW TABLES;
+-----+
| Tables_in_TestDB |
+-----+
| TestTable         |
+-----+
1 row in set (0.001 sec)

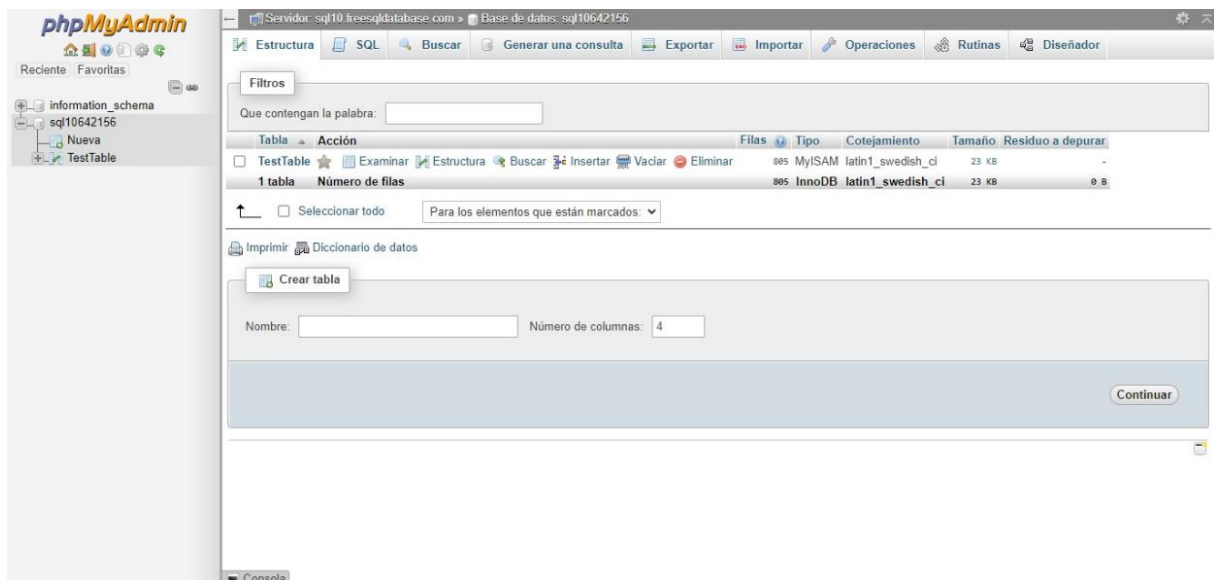
MariaDB [TestDB]> SELECT * FROM TestTable;
Empty set (0.002 sec)

MariaDB [TestDB]>

```

Figura 2.12

Creación de base de datos remota y tabla correspondiente



phpMyAdmin interface showing the 'TestTable' structure and the 'Crear tabla' (Create table) form.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
TestTable	Examinar Estructura Buscar Insertar Vaciar Eliminar	885	MyISAM	latin1_swedish_ci	23 KB	-
1 tabla		Número de filas	885	InnoDB	latin1_swedish_ci	23 KB 0 B

Crear tabla

Nombre: Número de columnas:

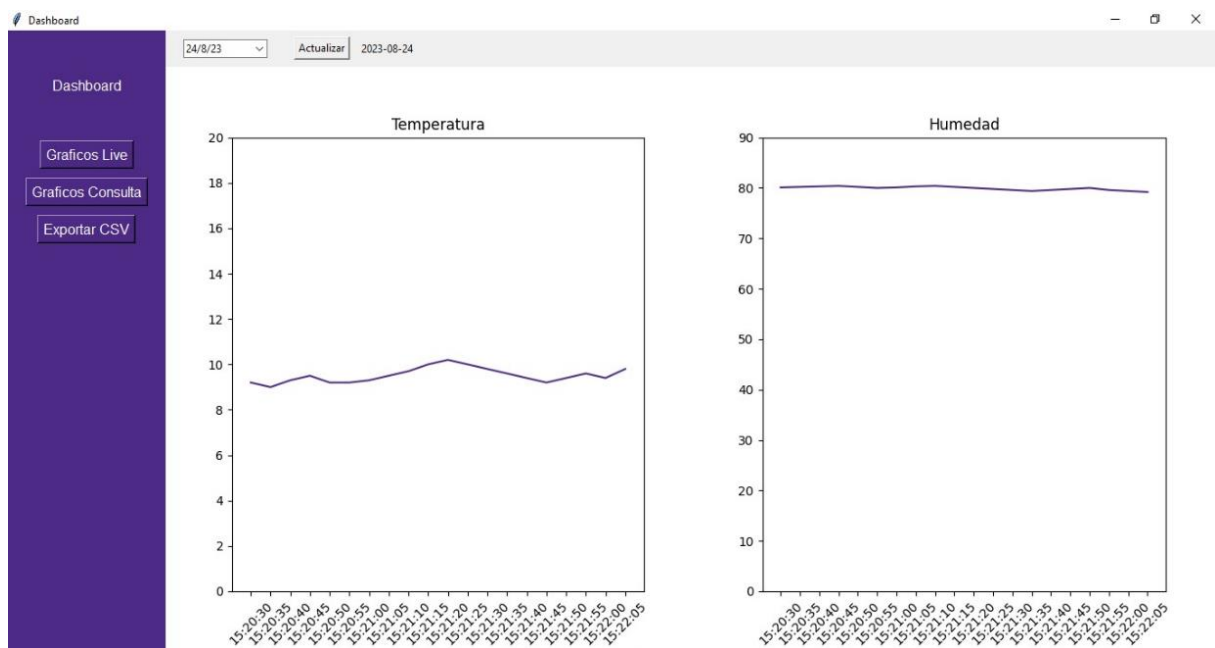
Continuar

2.5.9 Creación de interfaz de usuario

Para la implementación de la interfaz se creó una aplicación utilizando la biblioteca Tkinter de Python, además de librerías como matplotlib para la elaboración de gráficas y mysql.connector para realizar las conexiones con la base de datos. El diseño de la interfaz se visualiza en la Figura 2.13.

Figura 2.13

Diseño de la interfaz gráfica para la visualización de datos



Capítulo 3

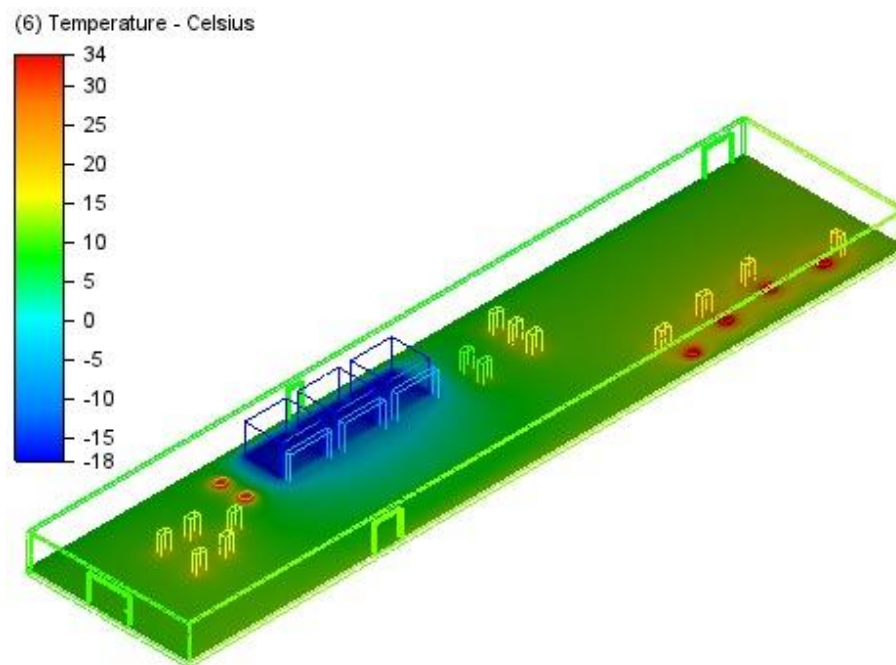
3.1 Resultados y análisis

3.1.1 Simulaciones de análisis termodinámico

Se procedió a realizar la simulación en Autodesk CFD para visualizar el comportamiento de la temperatura en el ambiente. Se tomó en cuenta como temperatura de prueba -18°C ya que esta es la temperatura aproximada a la que pueden llegar los equipos de enfriamiento.

Figura 3.1

Simulación de temperatura en ambiente de planta procesadora

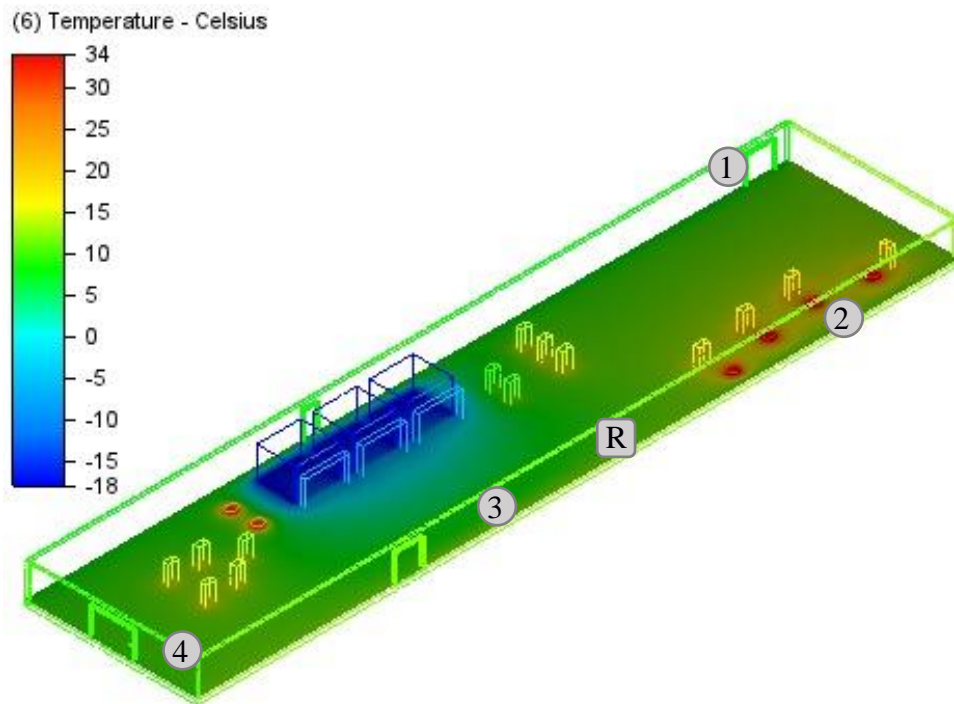


Se puede notar en la Figura 3.1 que las temperaturas altas se concentran en la zona de recepción/clasificación, que corresponden a las zonas más alejadas de los equipos de frío. La temperatura llega a su punto mínimo en el tratamiento debido a que en este punto se ubican los equipos de frío. Las temperaturas se vuelven a elevar en el área de empaque. A pesar de ello, las temperaturas en cada una de estas etapas se mantienen por debajo de los 10°C , lo cual indica una temperatura ideal en todas las áreas de proceso, según lo que se establece en la norma del GSA [4].

Los puntos de control identificados fueron 4, correspondientes a cada uno de los gradientes principales de color, tal como se muestra en la Figura 3.2. Estos puntos se pueden zonificar en sensor 1 – recepción, sensor 2 – clasificación, sensor 3 – tratamiento, sensor 4 – empaque.

Figura 3.2

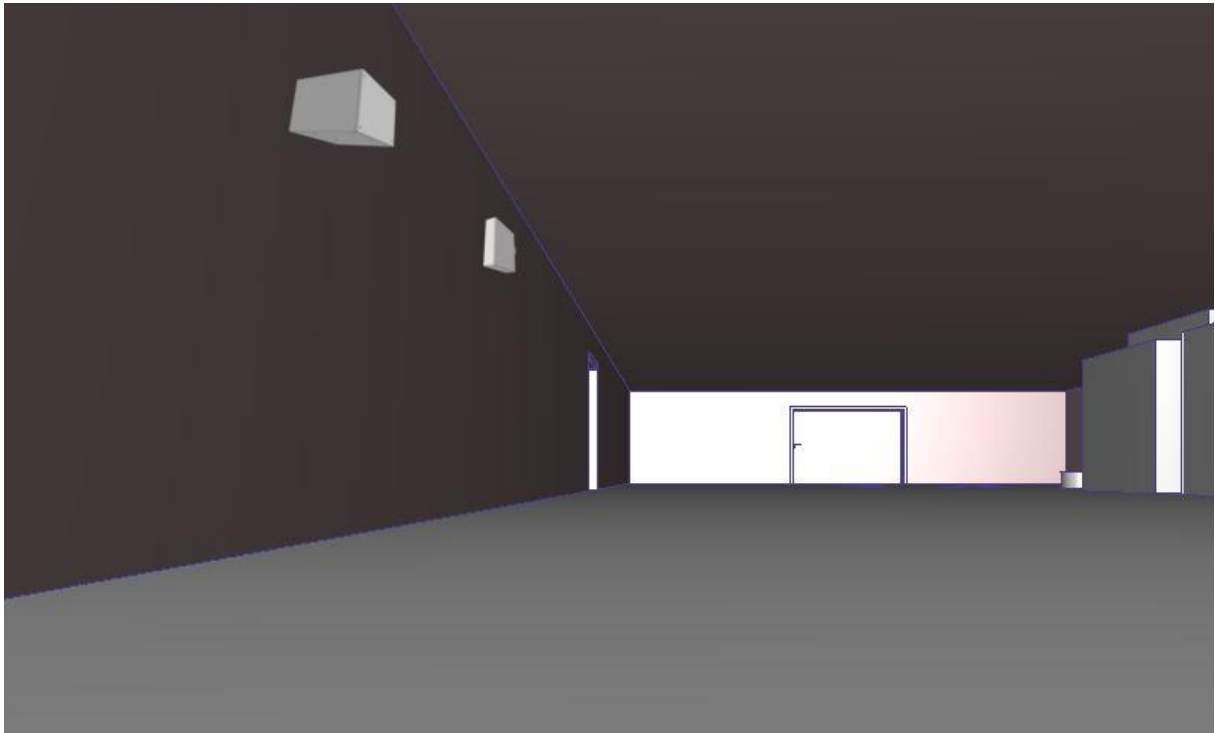
Ubicación de los sensores de temperatura y humedad



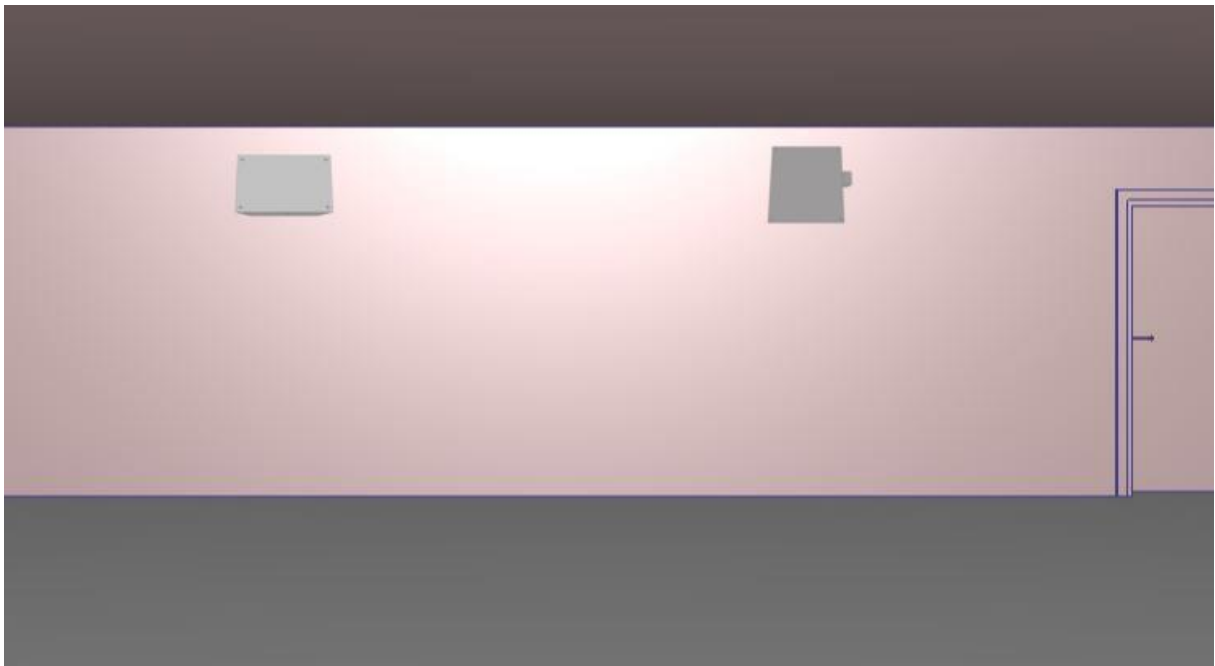
Como la Raspberry es el receptor de información, se ubicó en una zona central para abarcar todos los dispositivos emisores (ESP32). Estos equipos deben ser ubicados a al menos 2m de distancia del piso para evitar cualquier manipulación por parte del personal de proceso, como se observa en la Figura 3.3 y Figura 3.4. Adicionalmente, se consideró que el aire en las secciones cercanas al techo presenta una temperatura mayor debido a factores físicos, pues, el aire caliente al tener menor densidad tiende a subir. Por lo tanto, si estas temperaturas permanecen en el rango establecido, se tiene la seguridad de que las temperaturas al nivel del proceso son las adecuadas.

Figura 3.3

Ubicación recomendada de los dispositivos en la planta de proceso

**Figura 3.4**

Vista frontal de la ubicación sugerida de los dispositivos.

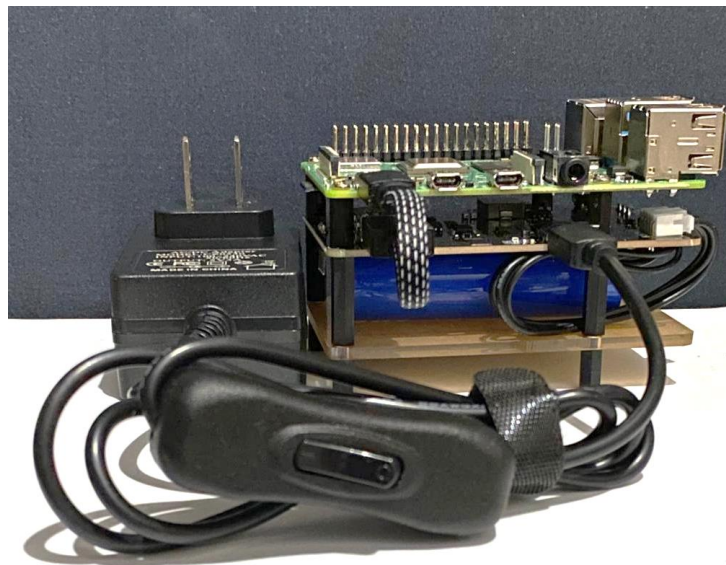


3.1.2 Establecimiento de circuitos.

El circuito receptor se formó a partir de la Raspberry Pi4, su UPS llamado PiPower con una batería de ión-litio de 2000 mAh a 7.4V con un tiempo estimado de 3 a 4 horas de duración y el respectivo cargador. Este circuito se muestra en la Figura 3.5.

Figura 3.5

Circuito físico de Raspberry Pi 4 con su respectivo UPS y cargador

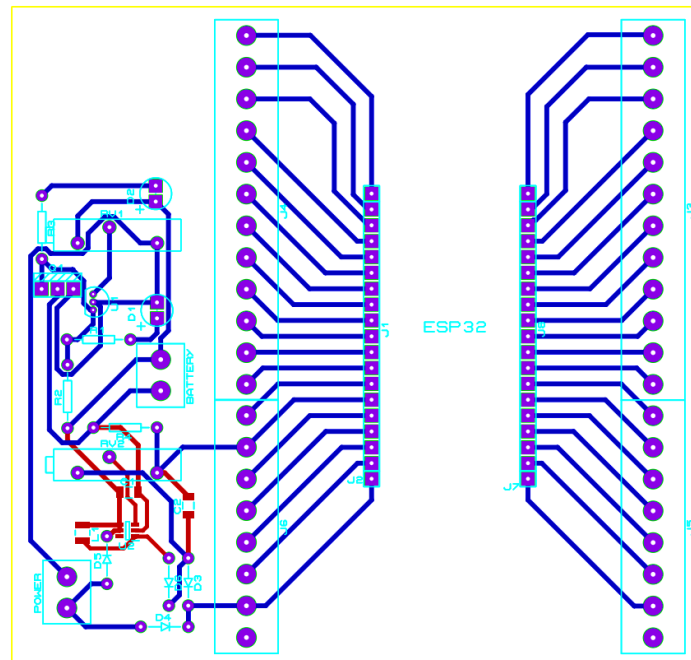


Una conexión adicional y recomendada es la del cableado ethernet para la RaspberryPi, de esta manera se asegura la conexión a la red sin intermitencias o interferencias.

Para el circuito emisor se diseñó la PCB para la integración de los circuitos, se estableció de esta manera para un fácil montaje de la placa ESP32 y asegurar de mejor manera el respectivo cableado. La PCB diseñada se puede visualizar en las Figura 3.6 y Figura 3.7.

Figura 3.6

Diseño de PCB (ESP32 + DHT22 + UPS)

**Figura 3.7**

Modelo 3D de PCB del circuito emisor



3.1.3 Consumos y baterías

Para el circuito de receptor: La Raspberry Pi 4, UPS y sus periféricos se abastecen con 3A y una tensión eléctrica de 5V.

Para el circuito emisor:

El requerimiento energético del ESP32 es 180 mA en utilización del WiFi, más el DHT22 con 2.5 mA, agregando el sistema de respaldo energético da como resultado aproximadamente 300 mA, necesitando una tensión eléctrica de 5V.

$$I_b = \frac{2 \times (0.3 \times 5)}{3.7 \times 1 \times 0.95} = 0.85 \text{ Ah}$$

Se obtuvo que la batería necesaria es de 850 mAh a 3.7V para que el circuito UPS del emisor dure al menos dos horas. Por lo tanto, se escogió la batería de ión-litio 18650 que tiene una capacidad de 1200 mAh a 3.7V.

Verificando la duración por medio de la ecuación propuesta en la sección 2.5.5:

$$T = \frac{1.2 \times 3.7 \times 1 \times 0.95}{0.3 \times 5} = 2.81 \text{ h}$$

Se obtiene que la duración aproximada de la batería bajo condiciones normales es de 2 a 3 horas.

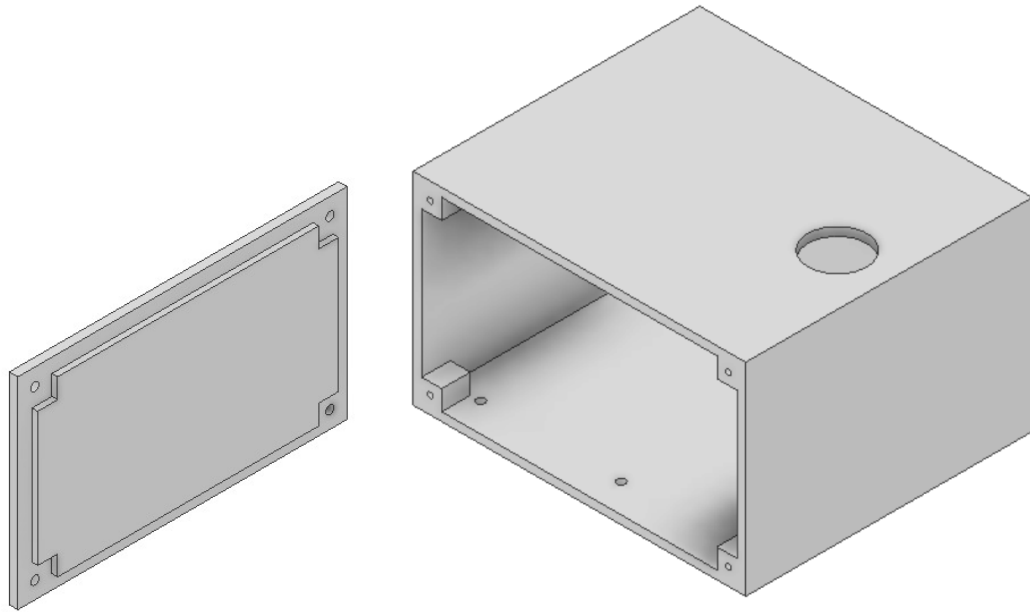
3.1.4 Diseño de protecciones físicas

Para el circuito receptor, en la parte superior se coloca un prensaestopa PG-13.5 para el cableado y la tapa contempla la colocación de un empaque para evitar filtraciones de humedad. De igual manera como el circuito anterior, en la caja protectora del circuito emisor se coloca una prensaestopa PG-13.5 para el cableado y un empaque para el sellado en la tapa de servicio. El alojamiento en la parte derecha corresponde al sitio del sensor DHT22 con un pequeño orificio para su fijación. Dichas características se visualizan en la Figura 3.8 y

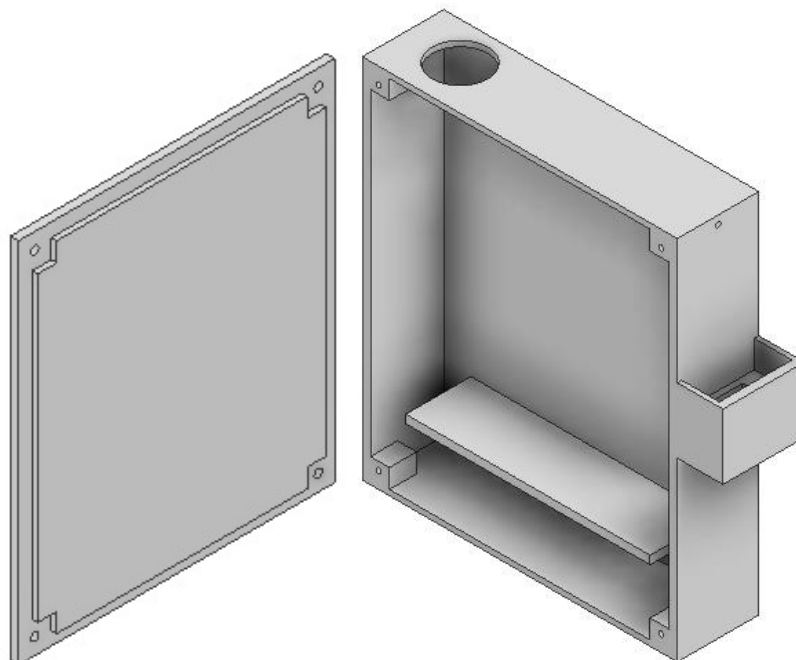
Figura 3.9.

Figura 3.8

Caja protectora para circuito receptor

**Figura 3.9**

Caja protectora para circuito emisor

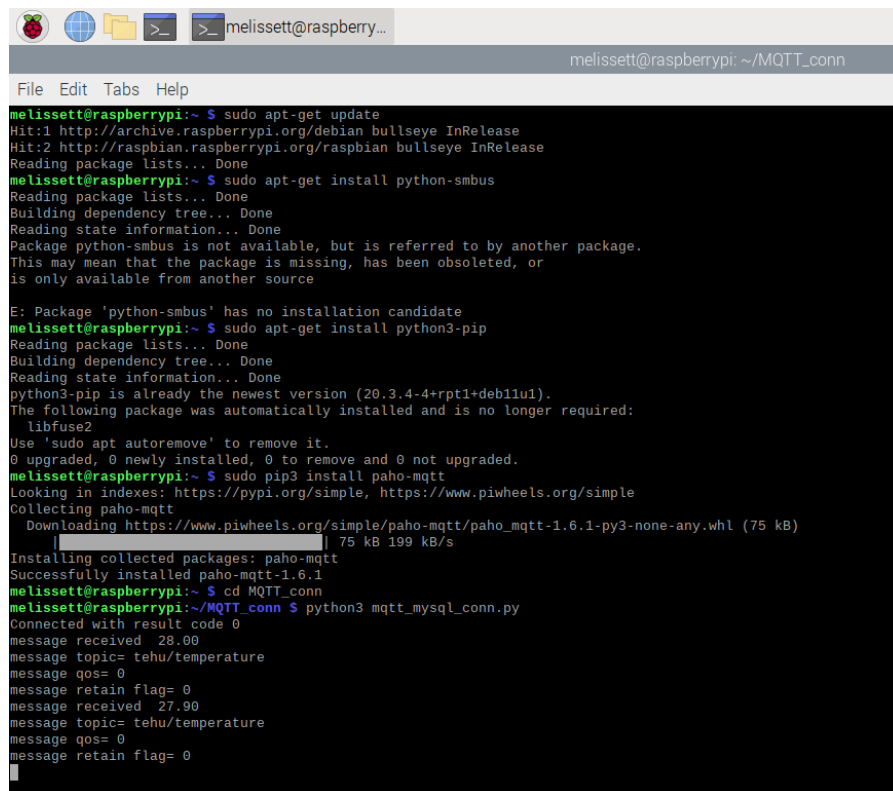


3.1.5 Programación

Se logró constatar el envío y recepción de datos a través de los dispositivos Raspberry Pi y ESP32 tal como se muestra en la Figura 3.10.

Figura 3.10

Pruebas de recepción de datos en la Raspberry Pi 4



```

melissett@raspberrypi:~$ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Reading package lists... Done
melissett@raspberrypi:~$ sudo apt-get install python-smbus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package python-smbus is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'python-smbus' has no installation candidate
melissett@raspberrypi:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-pip is already the newest version (20.3.4-4+rpt1+deb11u1).
The following package was automatically installed and is no longer required:
 libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
melissett@raspberrypi:~$ sudo pip3 install paho-mqtt
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-1.6.1-py3-none-any.whl (75 kB)
    |#####| 75 kB 199 kB/s
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.6.1
melissett@raspberrypi:~$ cd MQTT_conn
melissett@raspberrypi:~/MQTT_conn$ python3 mqtt_mysql_conn.py
Connected with result code 0
message received 28.00
message topic= tehu/temperature
message qos= 0
message retain flag= 0
message received 27.90
message topic= tehu/temperature
message qos= 0
message retain flag= 0

```

Una vez comprobada la transmisión de datos se creó el algoritmo de envío a la base de datos local y remota como se muestra en la Figura 3.11. En la Figura 3.12 se visualiza la comprobación del almacenamiento de los datos en la base de datos local.

Posteriormente, se comprobó el almacenamiento de los datos en la base de datos remota, la cual se visualiza en la Figura 3.13.

Figura 3.11

Pruebas de envío de información hacia la base de datos local y remota

```

melissett@raspberrypi:~/MQTT_conn
File Edit Tabs Help
melissett@raspberrypi:~/MQTT_conn $ sudo nano mqtt_mysql_conn.py
melissett@raspberrypi:~/MQTT_conn $ python3 mqtt_mysql_conn.py
Connected with result code 0
Temperature: 27.30
Humidity: 92.30
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.30,'2023-08-17 01:25:13.981002')
Success!
Temperature: 27.30
Humidity: 92.20
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.20,'2023-08-17 01:25:18.828063')
Success!
Temperature: 27.30
Humidity: 92.30
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.30,'2023-08-17 01:25:23.883818')
Success!
Temperature: 27.30
Humidity: 92.30
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.30,'2023-08-17 01:25:28.845048')
Success!
Temperature: 27.30
Humidity: 92.30
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.30,'2023-08-17 01:25:33.857333')
Success!
Temperature: 27.30
Humidity: 92.30
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.30,'2023-08-17 01:25:38.864607')
Success!
Temperature: 27.30
Humidity: 92.40
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.40,'2023-08-17 01:25:43.983699')
Success!
Temperature: 27.30
Humidity: 92.40
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.40,'2023-08-17 01:25:48.878670')
Success!
Temperature: 27.30
Humidity: 92.40
Sending query... INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES ('Sensor1',27.30,92.40,'2023-08-17 01:25:53.888352')
Success!

```

Figura 3.12

Recepción de información en la base de datos local

```

melissett@raspberrypi:~/MQTT_conn
File Edit Tabs Help
17 rows in set (0.001 sec)
MariaDB [TestDB]> SELECT * FROM TestTable;
+-----+-----+-----+-----+
| Sensor | temperatura | humedad | lectura |
+-----+-----+-----+-----+
| Sensor1 | 27.5 | 91.6 | 2023-08-17 01:14:37 |
| Sensor1 | 27.5 | 91.6 | 2023-08-17 01:14:42 |
| Sensor1 | 27.5 | 91.7 | 2023-08-17 01:14:42 |
| Sensor1 | 27.5 | 91.7 | 2023-08-17 01:14:47 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:14:47 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:14:52 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:14:52 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:14:57 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:14:57 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:02 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:02 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:07 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:07 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:12 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:12 |
| Sensor1 | 27.5 | 91.9 | 2023-08-17 01:15:17 |
| Sensor1 | 27.5 | 91.9 | 2023-08-17 01:15:17 |
| Sensor1 | 27.5 | 91.8 | 2023-08-17 01:15:17 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:13 |
| Sensor1 | 27.3 | 92.2 | 2023-08-17 01:25:18 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:23 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:28 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:33 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:38 |
| Sensor1 | 27.3 | 92.4 | 2023-08-17 01:25:43 |
| Sensor1 | 27.3 | 92.4 | 2023-08-17 01:25:48 |
| Sensor1 | 27.3 | 92.4 | 2023-08-17 01:25:53 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:25:58 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:26:04 |
| Sensor1 | 27.3 | 92.2 | 2023-08-17 01:26:08 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:26:14 |
| Sensor1 | 27.3 | 92.3 | 2023-08-17 01:26:18 |
+-----+-----+-----+-----+
31 rows in set (0.002 sec)
MariaDB [TestDB]>

```

Figura 3.13

Recepción de información en la base de datos remota

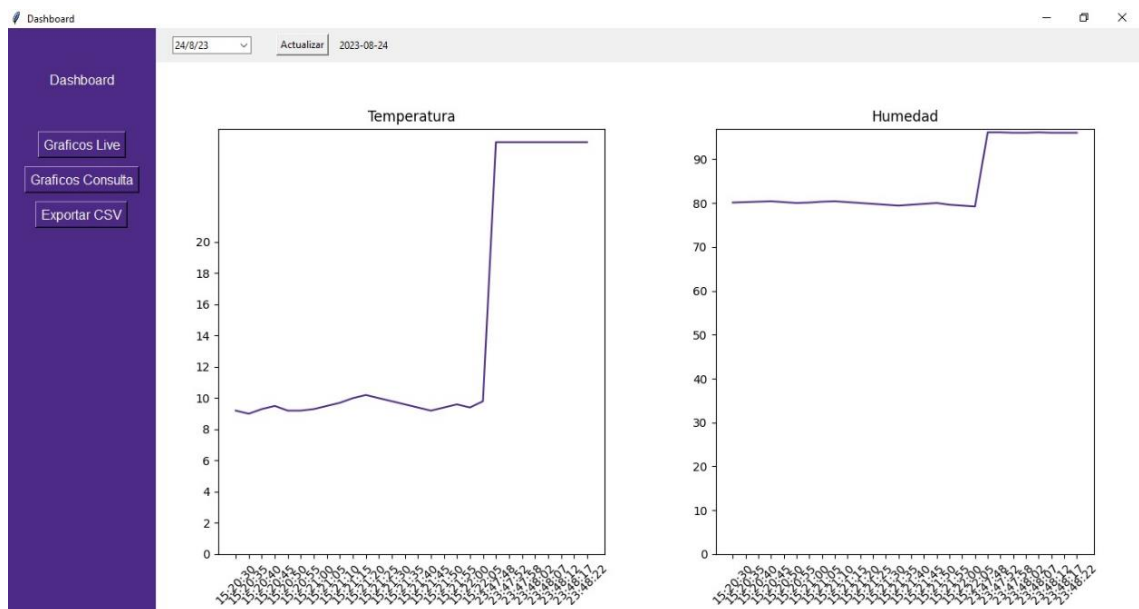
The screenshot shows the phpMyAdmin interface. The main window displays a table with the following data:

Sensor	Temperatura	Humedad	Lectura
Sensor1	9.2	80.1	2023-08-24 15:20:30
Sensor1	9	80.2	2023-08-24 15:20:35
Sensor1	9.3	80.3	2023-08-24 15:20:40
Sensor1	9.5	80.4	2023-08-24 15:20:45
Sensor1	9.2	80.2	2023-08-24 15:20:50
Sensor1	9.2	80	2023-08-24 15:20:55
Sensor1	9.3	80.1	2023-08-24 15:21:00
Sensor1	9.5	80.3	2023-08-24 15:21:05
Sensor1	9.7	80.4	2023-08-24 15:21:10
Sensor1	10	80.2	2023-08-24 15:21:15
Sensor1	10.2	80	2023-08-24 15:21:20
Sensor1	10	79.8	2023-08-24 15:21:25
Sensor1	9.8	79.6	2023-08-24 15:21:30
Sensor1	9.6	79.4	2023-08-24 15:21:35
Sensor1	9.4	79.6	2023-08-24 15:21:40
Sensor1	9.2	79.8	2023-08-24 15:21:45
Sensor1	9.4	80	2023-08-24 15:21:50
Sensor1	9.6	79.6	2023-08-24 15:21:55
Sensor1	9.4	79.4	2023-08-24 15:22:00

Al momento de realizar la conexión de la interfaz con la base de datos, se verificó la creación de gráficas y el cambio entre los datos de prueba y los datos en tiempo real, como se muestra en la Figura 3.14.

Figura 3.14

Conexión de la interfaz con la base de datos



3.1.6 Análisis de costos

El desglose de los costos se realizó dividiendo en tres secciones. Los costos de componentes para la construcción de la PCB se muestran en la Tabla 3.1.

Tabla 3.1

Componentes para la construcción de la PCB

Descripción	Precio unitario	Cantidad	Total
Capacitor 22u	\$0,75	1	\$0,75
Capacitor 28u	\$0,75	1	\$0,75
Led	\$0,30	2	\$0,60
Diodos schottky	\$0,60	4	\$2,40
Pin hembra 1x20	\$0,25	2	\$0,50
Bornera triple	\$0,21	12	\$2,52
Bornera doble	\$0,14	4	\$0,56
Inductor 22u	\$2,00	1	\$2,00
Irfz44n	\$1,40	1	\$1,40
Resistencia 1k	\$0,04	2	\$0,08
Resistencia 2.2k	\$0,05	2	\$0,10
Potenciómetro 10k	\$1,35	1	\$1,35
Potenciómetro 100k	\$1,35	1	\$1,35
TL431	\$1,50	1	\$1,50
MT3608	\$1,00	1	\$1,00
Soldadura estaño	\$4,00	1	\$4,00
Total			\$20,86

Se obtuvo el costo de dispositivos visualizado en la Tabla 3.2.

Tabla 3.2

Costos de placas de desarrollo, sensores y accesorios.

Descripción	Valor	Cantidad	Total
Raspberry Pi 4	\$125,00	1	\$125,00
Esp32	\$11,00	4	\$44,00
DHT22	\$12,00	4	\$48,00
Cargador Raspberry Pi 4	\$18,00	1	\$18,00
UPS Raspberry Pi 4	\$40,00	1	\$40,00
Cargador ESP32	\$9,80	4	\$39,20
Porta batería 18650	\$1,70	4	\$6,80
		Total	\$321,00

Se detallan los costos de materiales necesarios para la instalación de los equipos en la Tabla 3.3.

Tabla 3.3

Costos de materiales para instalación

Descripción	Valor	Cantidad	Total
Prensa estopa PG-13.5	\$0,26	5	\$1,30
Tubería funda sellada 1/2 (m)	\$1,60	5	\$8,00
Cordon de nitrilo (m)	\$5,00	3	\$15,00
		Total	\$24,30

Finalmente, el costo de los servicios de diseño, fabricación y programación se detallan a continuación en la Tabla 3.4.

Tabla 3.4*Costos de diseño y fabricación*

Descripción	Valor	Cantidad	Total
Diseño PCB	\$60,00	1	\$60,00
Fabricación PCB	\$25,00	1	\$25,00
Diseño piezas 3D	\$80,00	1	\$80,00
Impresión 3D Case Raspberry	\$40,00	1	\$40,00
Impresión 3D Tapa Raspberry	\$7,50	1	\$7,50
Impresión 3D Case ESP32	\$30,00	4	\$120,00
Impresión 3DTapa ESP32	\$15,00	4	\$60,00
Hosting BD 1 año	\$80,00	1	\$80,00
Instalación de programa personalizado	\$100,00	1	\$100,00
		Total	\$512,50

Al sumar todos los costos especificados en las tablas, se obtiene que la inversión total del sistema es de \$878,66. Siendo este un precio bastante proporcionado en relación con las opciones que existen en el mercado de dispositivos con características similares. Estos costos pueden reducirse reduciendo tamaños como el de la PCB y, por tanto, de la caja protectora permitiendo disminuir extensión de la propia PCB y el tiempo de impresión y gramos de material utilizado en las cajas protectoras.

Capítulo 4

4.1 Conclusiones y recomendaciones

4.1.1 Conclusiones

Mediante el desarrollo del presente proyecto fue posible diseñar un sistema mecatrónico de monitoreo de temperatura y humedad a ser implementado en una planta camaronera para la conservación de la calidad e inocuidad del producto mediante el uso de sensores, bases de datos, interfaces de usuario intuitivas y protecciones adecuadas para los circuitos.

Primero, se logró establecer la configuración del sistema para el monitoreo de puntos críticos por medio de un análisis de dimensiones e identificación de zonas temperatura elevada por medio de la herramienta CFD de Autodesk, estableciendo 4 puntos de diferencial de temperatura y una ubicación central para el circuito de recepción y almacenamiento de datos.

Segundo, se estableció la configuración de los circuitos emisor y receptor; posteriormente, se realizó la consulta de opciones de circuitos para el UPS de la parte emisora y receptora donde se seleccionó el PiPower para el receptor y la creación de una PCB para el emisor. Una vez obtenidas las dimensiones de los circuitos se diseñaron las cajas protectoras de los circuitos a través de la herramienta Inventor, se seleccionaron accesorios para las conexiones eléctricas y aislamientos de las cajas. Finalmente, se obtuvieron tiempos de impresión y material utilizado con la herramienta UltiMaker Cura.

Tercero, se crearon bases de datos no relacionales en mysql para la base de datos local y remota, las tablas contienen los datos de nombre del sensor, fecha y hora en formato dd/mm/aaaa hh:mm:ss y dato actual de temperatura y humedad según la fecha especificada. Estos datos ingresan y se consultan en intervalos de 1 minuto.

Finalmente, fue posible desarrollar una interfaz de usuario que facilite el proceso de consulta y análisis de datos de manera fácil e interactiva permitiendo mayor agilidad en el análisis de la información para la resolución de problemas. La conexión para la consulta se

realiza con la base de datos remota para permitir el acceso fuera de la red local, en caso de quererla limitar, se puede configurar una conexión únicamente a la base de datos local.

4.1.2 Recomendaciones

Entre las recomendaciones a realizar para mejorar el diseño del dispositivo están:

1. La reducción de tamaño de la PCB para optimizar la utilización del espacio y disminuir costos de fabricación e impresión 3D.
2. Mejorar el aspecto e interacción en la interfaz gráfica, así como la implementación de filtros de búsqueda más específicos que ayuden a la consulta de datos.

Otras recomendaciones al continuar con el desarrollo de los dispositivos son:

3. Realizar la implementación física del circuito de la PCB para pruebas de funcionamiento y verificación de consumo eléctrico real, y, por tanto, ajustes en la capacidad de batería necesaria.
4. Configurar las placas Raspberry Pi4 y ESP32 para la reducción de recursos utilizados durante el funcionamiento y limitarlos a los necesarios para la obtención, envío y recepción de datos.
5. Realizar pruebas de campo con dos o más dispositivos para verificar la dinámica de la base de datos y la interfaz en un envío y recepción de datos mucho más saturado que el actual.
6. Implementar una conexión dinámica entre la base de datos local y remota para la recuperación automática de datos en la base de datos remota luego de periodos de desconexión como es en el caso de un apagón.

Referencias

- [1] «Ecuador (ECU) Exports, Imports, and Trade Partners | OEC», *OEC - The Observatory of Economic Complexity*. <https://oec.world/en/profile/country/ecu> (accedido 4 de junio de 2023).
- [2] «Ecuador's shrimp industry clearing numerous hurdles in 2020 - Responsible Seafood Advocate», *Global Seafood Alliance*, 5 de octubre de 2020. <https://www.globalseafood.org/advocate/ecuadors-shrimp-industry-clearing-numerous-hurdles-in-2020/> (accedido 11 de junio de 2023).
- [3] «Best Aquaculture Practices», *Best Aquaculture Practices Certification*. <https://www.bapcertification.org/> (accedido 4 de junio de 2023).
- [4] «Critical decisions for shrimp harvesting and packing, Part 3 - Responsible Seafood Advocate», *Global Seafood Alliance*, 3 de octubre de 2016. <https://www.globalseafood.org/advocate/critical-decisions-for-shrimp-harvesting-and-packing-part-3/> (accedido 4 de junio de 2023).
- [5] «AQUACULTURA #139 by REVISTA AQUACULTURA - Cámara Nacional de Acuicultura - Issuu», 25 de febrero de 2021. <https://issuu.com/revista-cna/docs/edicion139> (accedido 4 de junio de 2023).
- [6] «Los alimentos en conserva envasados en la casa y el botulismo», *Centers for Disease Control and Prevention*, 27 de mayo de 2022. <https://www.cdc.gov/foodsafety/es/communication/home-canning-and-botulism.html> (accedido 4 de junio de 2023).
- [7] C. for F. S. and A. Nutrition, «Recalls, Outbreaks & Emergencies», *FDA*, 25 de enero de 2022. <https://www.fda.gov/food/recalls-outbreaks-emergencies> (accedido 4 de junio de 2023).

- [8] J. T. Afa, «The Humidity Effect on Machines in the Coastal Area of the Niger Delta», *Am. J. Electr. Electron. Eng.*, vol. 1, n.º 3, Art. n.º 3, ene. 2013, doi: 10.12691/ajeec-1-3-2.
- [9] H. Boyes, B. Hallaq, J. Cunningham, y T. Watson, «The industrial internet of things (IIoT): An analysis framework», *Comput. Ind.*, vol. 101, pp. 1-12, oct. 2018, doi: 10.1016/j.compind.2018.04.015.
- [10] J.-P. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, y M. Malli, «Cyber-physical systems security: Limitations, issues and future trends», *Microprocess. Microsyst.*, vol. 77, p. 103201, sep. 2020, doi: 10.1016/j.micpro.2020.103201.
- [11] «Internet des Objets (IoT / IIoT) - Neosoft Technologies», *Neosoft*. <https://neosoft.ca/fr/solutions/iot-iiot/> (accedido 14 de julio de 2023).
- [12] «What Is Industrial IoT (IIoT)?», *Cisco*. <https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-industrial-iiot.html> (accedido 15 de junio de 2023).
- [13] «Arduino Docs | Arduino Documentation». <https://docs.arduino.cc/> (accedido 20 de junio de 2023).
- [14] S.-M. Kim, Y. Choi, y J. Suh, «Applications of the Open-Source Hardware Arduino Platform in the Mining Industry: A Review», *Appl. Sci.*, vol. 10, n.º 14, Art. n.º 14, ene. 2020, doi: 10.3390/app10145018.
- [15] «Raspberry Pi Documentation - Microcontrollers». <https://www.raspberrypi.com/documentation/microcontrollers/> (accedido 20 de junio de 2023).
- [16] «Technical Documents | Espressif Systems». <https://www.espressif.com/en/support/documents/technical-documents> (accedido 20 de junio de 2023).

- [17] J. Xiao y J. T. Li, «Design and Implementation of Intelligent Temperature and Humidity Monitoring System Based on ZigBee and WiFi», *Procedia Comput. Sci.*, vol. 166, pp. 419-422, ene. 2020, doi: 10.1016/j.procs.2020.02.072.
- [18] T. Kalsoom, N. Ramzan, S. Ahmed, y M. Ur-Rehman, «Advances in Sensor Technologies in the Era of Smart Factory and Industry 4.0», *Sensors*, vol. 20, n.º 23, Art. n.º 23, ene. 2020, doi: 10.3390/s20236783.
- [19] H. Farahani, R. Wagiran, y M. N. Hamidon, «Humidity Sensors Principle, Mechanism, and Fabrication Technologies: A Comprehensive Review», *Sensors*, vol. 14, n.º 5, Art. n.º 5, may 2014, doi: 10.3390/s140507881.
- [20] «Humidity Sensor BME280», *Bosch Sensortec*. <https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/> (accedido 8 de junio de 2023).
- [21] «Data Logger Ecuador Registradores de Temperatura y Humedad». <http://imporcomsa.com/data-loggers-ecuador/> (accedido 8 de junio de 2023).
- [22] «Temperature and humidity monitoring for warehouses and product storages». <https://sensmax.eu/solutions/temperature-and-humidity-monitoring-for-warehouses-and-product-storages/> (accedido 8 de junio de 2023).
- [23] S. Medrano y W. Jose, «Diseño e implementación de un prototipo de monitoreo de temperatura, humedad y presión con comunicación vía radio e internet para mejorar la producción agrícola».
- [24] «6GK7177-1MA20-0AA0 - Industry Support Siemens». <https://support.industry.siemens.com/cs/pd/502356?pdtd=td&dl=en&pnid=13617&lc=en-WW> (accedido 20 de junio de 2023).

Apéndices

Apéndice A

Código circuito receptor (Raspberry Pi)

```
#!/usr/bin/env python3

import mysql.connector as db_conn
import paho.mqtt.client as mqtt
import ssl
from datetime import datetime

local_db = db_conn.connect(user='admin', password='admin11', database='TestDB')
local_cursor = local_db.cursor()

remote_db = db_conn.connect(host='sql10.freemdb.com', port=3306, user='sql10642156', password='u1Hv3aEbhd', database='sql10642156')
remote_cursor = remote_db.cursor()

temperatura = ""
humedad = ""
dict = {}
dict['sensor'] = 'Sensor1'
dict['temperatura'] = ""
dict['humedad'] = ""

def on_message(client, userdata, message):
    if message.topic == "tehu/temperature":
        temperatura = str(message.payload.decode("utf-8"))
        print("Temperature: ", temperatura)
        dict['temperatura'] = temperatura

    elif message.topic == "tehu/humidity":
        humedad = str(message.payload.decode("utf-8"))
        print("Humidity: ", humedad)
        dict['humedad'] = humedad

    dict['lectura'] = str(datetime.now())
    values = "Sensor1," + dict['temperatura'] + ", " + dict['humedad'] + ", " + str(datetime.now()) + ""
    sql = "INSERT INTO TestTable (Sensor,temperatura,humedad,lectura) VALUES (%s)" % values

    try:
        print("Sending query... ",sql)
        local_cursor.execute(sql, dict.values())
        remote_cursor.execute(sql, dict.values())
        print("Success!")
    except db_conn.Error as error:
        print("ERROR: {}".format(error))

    local_db.commit()
    remote_db.commit()

def on_connect(client,userdata,flags,rc):
    print("Connected with result code " + str(rc))
    client.subscribe("tehu/temperature",1)
    client.subscribe("tehu/humidity",1)

client = mqtt.Client("Raspi")

client.username_pw_set("tehu","tehu11")
```

```
client.connect("192.168.100.23",1883,60)
```

```
client.on_connect = on_connect  
client.on_message = on_message
```

```
client.loop_forever()
```

Apéndice B

Código circuito emisor (ESP32)

```
#include "DHT.h"
#include <WiFi.h>
#include <PubSubClient.h>

#define pinDatos 21
#define ledPin 4

DHT sensorTH (pinDatos, AM2301);
WiFiClient espClient;
PubSubClient client(espClient);

const char* ssid = "RedWiFi";
const char* password = "mypass";
const char* mqtt_server = "192.168.100.23";

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

String data_collect = "";
char data_tehu[15];

unsigned long previousMillis = 0UL;
unsigned long interval = 60000UL;

void setup() {
  Serial.begin (115200);
  sensorTH.begin();
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval){
    getTemp_Hum();
    client.publish("tehu/temp_hum", data_tehu);
    previousMillis = currentMillis;
  }
}

void getTemp_Hum(){
  float temp = sensorTH.readTemperature();
  float hum = sensorTH.readHumidity();
  data_collect = String(temp) + "_" + String(hum);
  data_collect.toCharArray(data_tehu,15);
  Serial.print("Sending temperature: ");

  Serial.println(temp);
  Serial.print("Sending humidity: ");
  Serial.println(hum);
}

void setup_wifi(){
  delay(10);
  // Connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned
int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32Client","tehu",
"tehu11")) {
      Serial.println("connected");
      // Subscribe
      client.subscribe("tehu/output");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

Apéndice C

Código conexión a base de datos (Interfaz)

```
import mysql.connector as mc
from datetime import datetime
from datetime import timedelta

tempdata={}
humdata={}
config ={
    'host':'sql110.freesqldatabase.com',
    'port': 3306,
    'user':'sql110642156',
    'password':'u1Hv3aEbhD',
    'database':'sql110642156',
}

def actdata():
    now = datetime.now()
    delta = timedelta(minutes=10)
    onehour = now - delta
    cnx = mc.connect(**config)
    cur = cnx.cursor()
    print("SELECT Sensor, Temperatura, Humedad, Lectura FROM TestTable WHERE Lectura BETWEEN '%s'
AND '%s'"%(onehour,now))

    cur.execute("SELECT Sensor, Temperatura, Humedad, Lectura FROM TestTable WHERE Lectura
BETWEEN '%s' AND '%s'"%(onehour,now))
    tempdata.clear()
    humdata.clear()
    for (Sensor, Temperatura, Humedad, Lectura) in cur:
        time=datetime.strptime(Lectura.strftime("%H:%M:%S"),"%H:%M:%S")
        tempdata.update({time:Temperatura})
        humdata.update({time:Humedad})
    cur.close()
    cnx.close()

def obtdata(fecha):
    cnx = mc.connect(**config)
    cur = cnx.cursor()
    cur.execute("SELECT Sensor, Temperatura, Humedad, Lectura FROM TestTable WHERE Lectura
BETWEEN '%s 00:00:00' AND '%s 23:59:59'"%(fecha,fecha))
    tempdata.clear()
    humdata.clear()
    for (Sensor, Temperatura, Humedad, Lectura) in cur:
        time=datetime.strptime(Lectura.strftime("%H:%M:%S"),"%H:%M:%S")
        tempdata.update({time:Temperatura})
        humdata.update({time:Humedad})
    cur.close()
    cnx.close()
```

Apéndice D

Código de interfaz

```
#Imports de Matplotlib
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.animation as animation
from matplotlib import style
import matplotlib.pyplot as plt
import matplotlib.dates as md

#Imports tkinter
import tkinter as tk
from tkinter import ttk

#Import tkcalendar
from tkcalendar import DateEntry

#Import data.py
from data import tempdata,humdata,obtdata,actdata

style.use("ggplot")
is_Live = True

#Inicializar gráficas de temperatura
temp_Figure, temp_axis = plt.subplots()

#Inicializar gráficas de humedad
hum_Figure, hum_axis = plt.subplots()

class Page(tk.Frame): #Pagina principal
    def __init__(self, *args, **kwargs):
        tk.Frame.__init__(self, *args, **kwargs)
        is_Live = False
    def show(self):
        self.lift()

class live(Page): #Pestaña de datos en vivo
    def __init__(self, *args, **kwargs):
        Page.__init__(self, *args, **kwargs)
        is_Live = True
        show_info(self)

class consu(Page): #Pestaña de consulta de datos (filtros por fecha)
    def __init__(self, *args, **kwargs):
        Page.__init__(self, *args, **kwargs)
        is_Live = False

class csv(Page): #Pestaña de exportar CSV
    def __init__(self, *args, **kwargs):
        Page.__init__(self, *args, **kwargs)
        is_Live = False

class MainView(tk.Frame):
    def __init__(self, *args, **kwargs):
```

```

tk.Frame.__init__(self, *args, **kwargs)

#Instancias de las páginas
livepage = live(self)
consumpage = consu(self)
csvpage = csv(self)

#Creación de sección de botones (color morado)
buttonframe = tk.Frame(self, bg="#4C2A85")
buttonframe.pack(side="left", fill="y")

#Creación de sección de gráficos (color blanco)
container = tk.Frame(self)
container.pack(side="top", fill="both", expand=True)

##
livepage.place(in_=container, x=0, y=0, relwidth=1, relheight=1)
consumpage.place(in_=container, x=0, y=0, relwidth=1, relheight=1)
csvpage.place(in_=container, x=0, y=0, relwidth=1, relheight=1)

#Inicialización de página principal
label = tk.Label(buttonframe, text="Dashboard", bg="#4C2A85", fg="#FFF", font=25)
label.pack(pady=50, padx=20)

btllive = tk.Button(buttonframe, text="Gráficos en vivo", bg="#4C2A85", fg="#FFF", font=25,
command=livepage.show)
btllive.pack(pady=0, padx=20)

bttgraph = tk.Button(buttonframe, text="Consultar gráficos", bg="#4C2A85", fg="#FFF",
font=25,command=consumpage.show)
bttgraph.pack(pady=10, padx=20)

btcsv = tk.Button(buttonframe, text="Exportar CSV", bg="#4C2A85", fg="#FFF",
font=25,command=csvpage.show)
btcsv.pack(pady=0, padx=20)

#Contenedor para visualización de datos en vivo
liveFrame = tk.Frame(livepage)
liveFrame.pack(fill="both", expand=True)

#Contenedor para ingreso de fechas
consumFrame1 = tk.Frame(consumpage)
consumFrame1.pack(fill="both", expand=True)

label=tk.Label(consumFrame1,text='Fecha')
label.pack(side = "left")

calendar = DateEntry(consumFrame1,selectmode='day')
calendar.pack(side = "left", padx = 20,pady =10)

#Contenedor para botón de actualización de datos
consumFrame2 = tk.Frame(consumpage)
consumFrame2.pack(fill="both", expand=True)

refresh_Button = tk.Button(consumFrame1,text='Actualizar',
command=lambda:update_Info(label,calendar,consumFrame2))
refresh_Button.pack(side ="left", padx=10)

#Estilo de gráficas
plt.rcParams["axes.prop_cycle"] = plt.cycler(

```

```
color=["#4C2A85", "#BE96FF", "#957DAD", "#5E366E", "#A98CCC"])
```

```
def configure_plots():  
    #temp_Figure.set_size_inches(6,10)  
    temp_axis.set_title("Temperatura")  
    temp_axis.set_yticks(range(0, 30,2))  
    temp_axis.xaxis.set_major_locator(md.SecondLocator(bysecond=range(60),interval = 30))  
    temp_axis.xaxis.set_major_formatter(md.DateFormatter('%H:%M:%S'))  
    temp_Figure.autofmt_xdate()  
    plt.xticks(rotation = 45)  
  
    #temp_Figure.set_size_inches(6,10)  
    hum_axis.set_title("Humedad")  
    hum_axis.set_yticks(range(0, 100,5))  
    hum_axis.xaxis.set_major_locator(md.SecondLocator(bysecond=range(60),interval = 30))  
    hum_axis.xaxis.set_major_formatter(md.DateFormatter('%H:%M:%S'))  
    hum_Figure.autofmt_xdate()  
    plt.xticks(rotation = 45)
```

```
def update_Info(label, calendar, This_Frame):  
    #Se limpian parámetros  
    temp_axis.clear()  
    hum_axis.clear()  
  
    #Configuración de gráficas  
    configure_plots()  
  
    #Se obtiene la fecha seleccionada  
    fecha=calendar.get_date()  
    label.config(text=fecha)  
  
    #Se consultan los datos de esa fecha  
    obtdata(fecha)  
  
    #Se setean parámetros  
    temp_axis.plot(list(tempdata.keys()), list(tempdata.values()))  
    hum_axis.plot(list(humdata.keys()), list(humdata.values()))  
  
    show_info(This_Frame)
```

```
def update_Temp_liveInfo(FrameData):  
    temp_axis.clear()  
  
    actdata()  
  
    #Se setean parámetros  
    temp_axis.plot(list(tempdata.keys()), list(tempdata.values()))  
  
    #Configuración de gráficas  
    configure_plots()
```

```
def update_Hum_liveInfo(FrameData):  
    hum_axis.clear()  
  
    actdata()  
  
    #Se setean parámetros  
    hum_axis.plot(list(humdata.keys()), list(humdata.values()))  
  
    #Configuración de gráficas
```

```

configure_plots()

def show_info(This_Frame):
    #Se muestran los parámetros en las gráficas
    canvas1 = FigureCanvasTkAgg(temp_Figure, This_Frame)
    canvas1.draw()
    canvas1.get_tk_widget().pack(side="left", fill="both", expand=True)

    canvas2 = FigureCanvasTkAgg(hum_Figure, This_Frame)
    canvas2.draw()
    canvas2.get_tk_widget().pack(side="left", fill="both", expand=True)

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Dashboard")
    root.state('zoomed')
    main = MainView(root)
    main.pack(side="top", fill="both", expand=True)
    root.wm_geometry("1000x600")
    configure_plots()

    if is_Live:
        temp_animation = animation.FuncAnimation(temp_Figure, update_Temp_liveInfo, interval=5000)
        hum_animation = animation.FuncAnimation(hum_Figure, update_Hum_liveInfo, interval=5000)

    root.mainloop()

```


Apéndice E

Pruebas de funcionamiento

Figura E.1

Colocación del sensor en un medio frío y húmedo para pruebas de funcionamiento



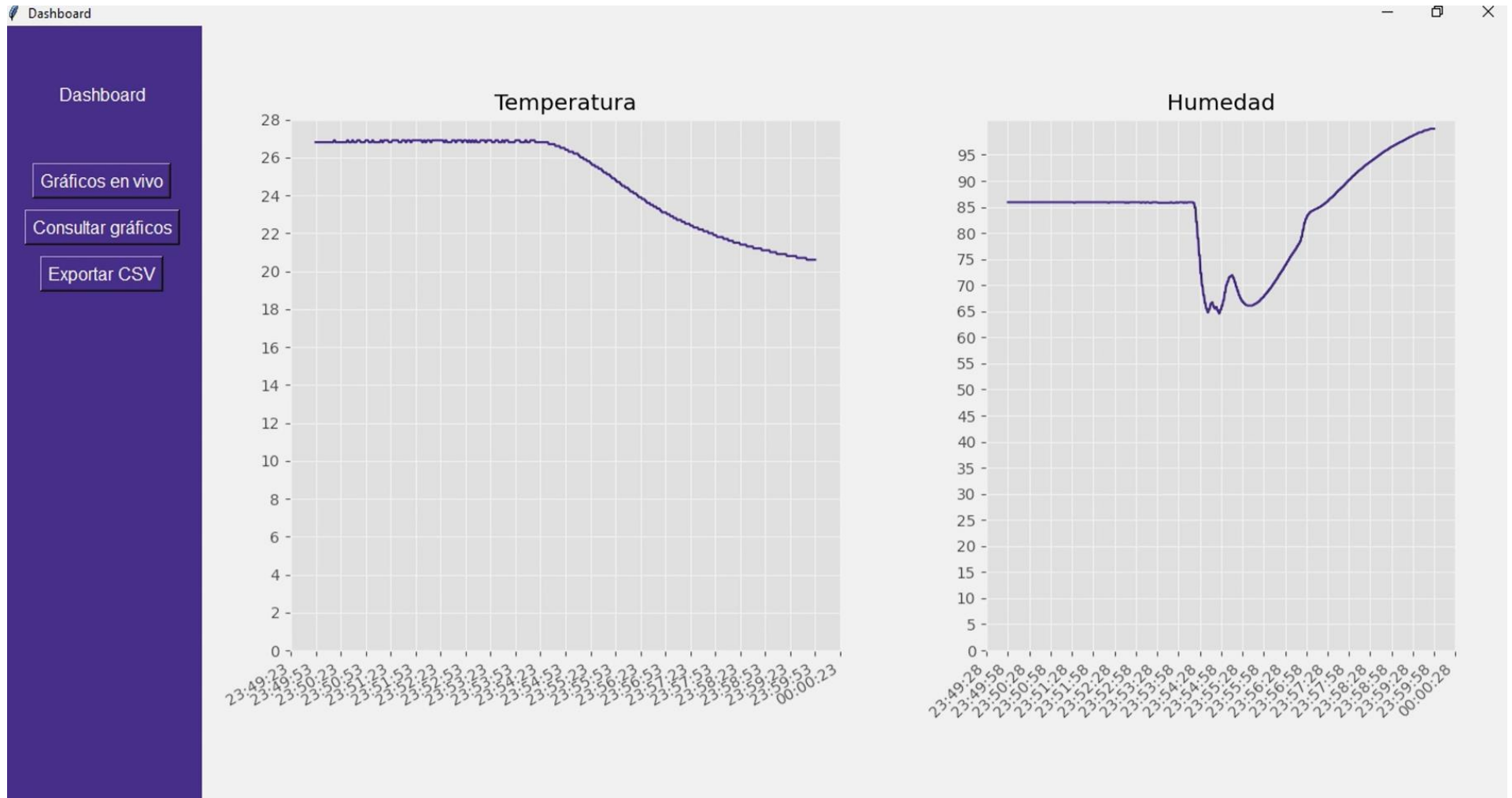
Figura E.2

Se encierra el sensor para una mejor percepción de temperatura



Figura E.3

Visualización de datos en tiempo real, paso gradual de temperatura ambiente a temperaturas menores y aumento de humedad relativa.



Apéndice F

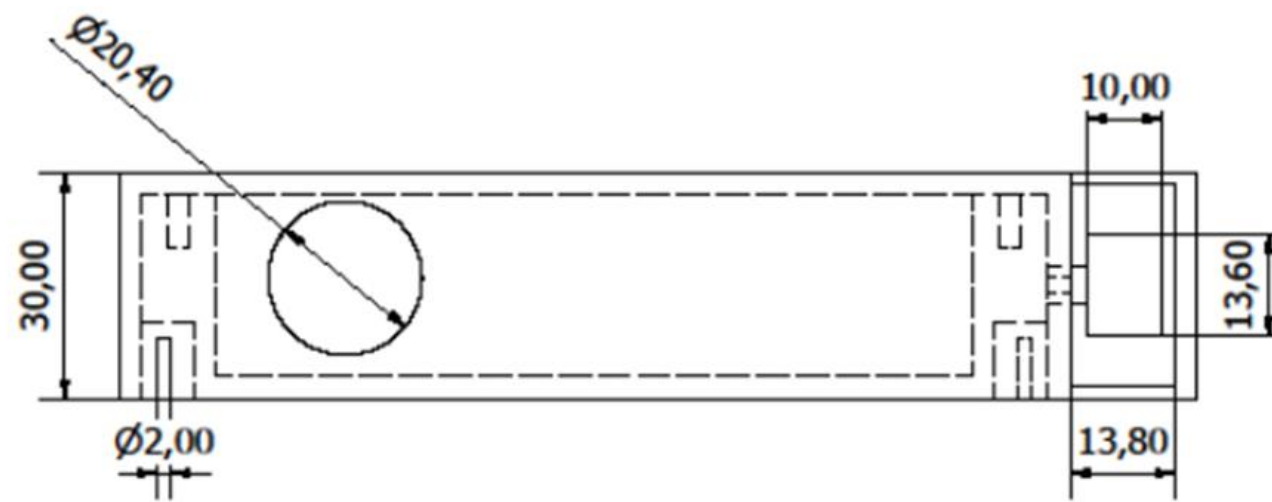
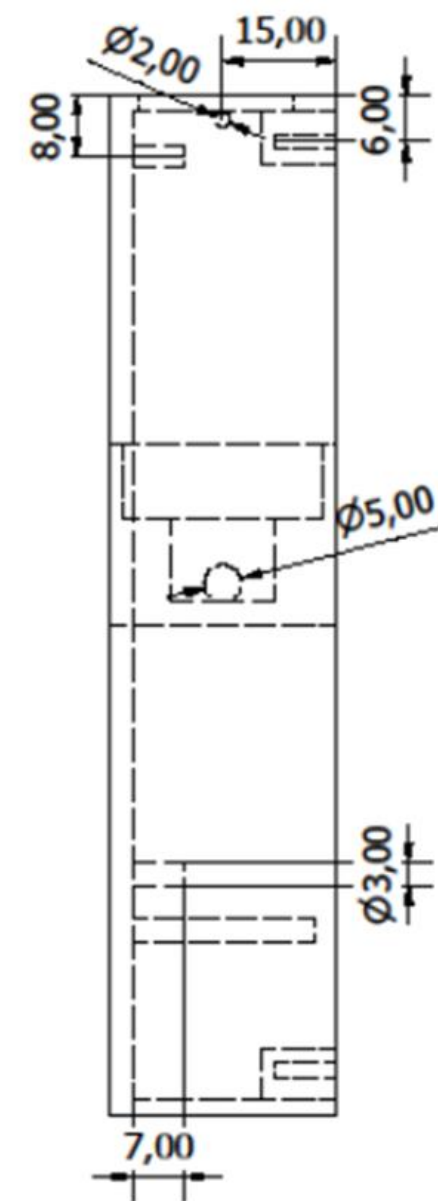
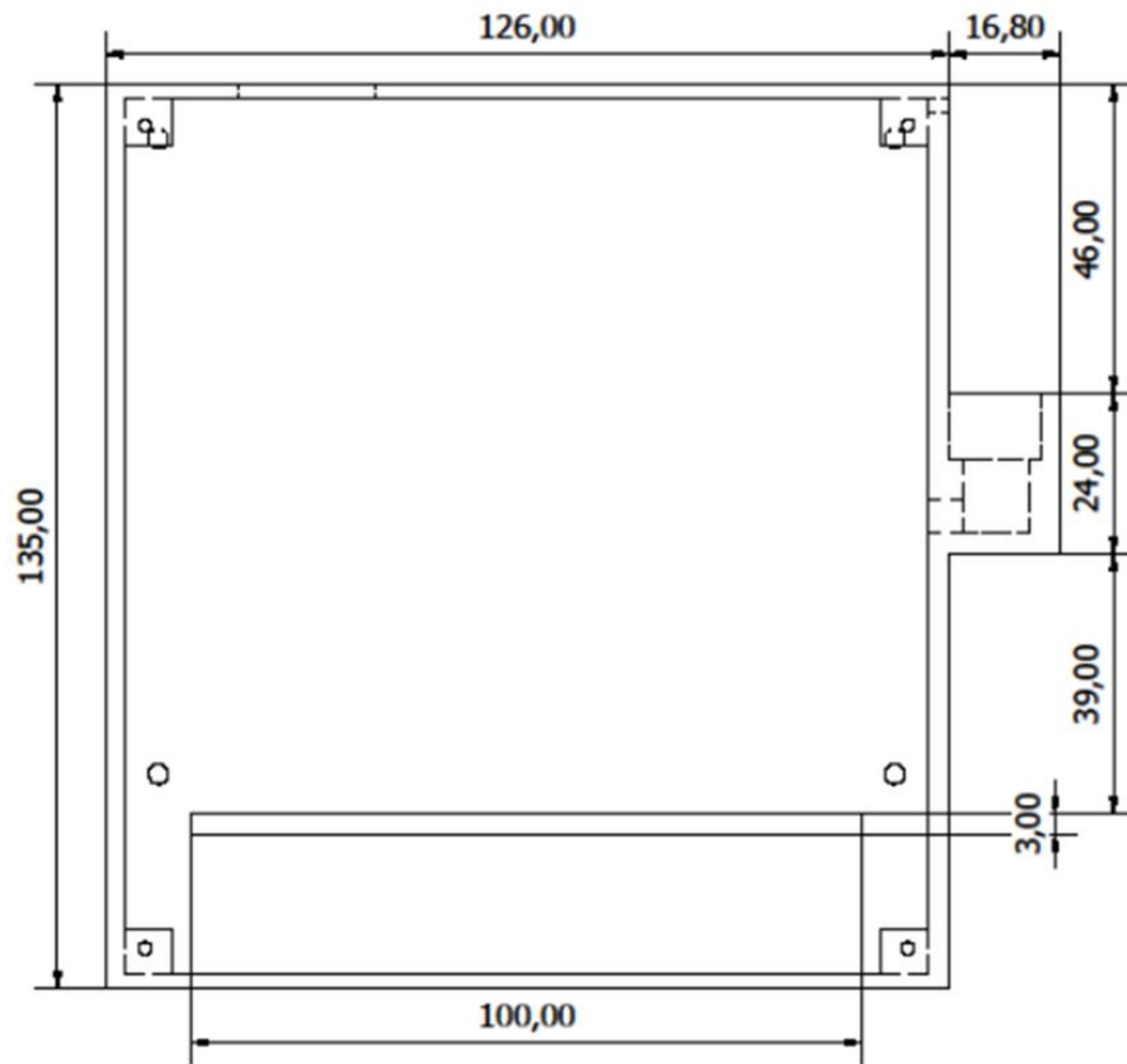
Planos de los protectores de circuitos

Plano 1. Caja protectora del circuito emisor

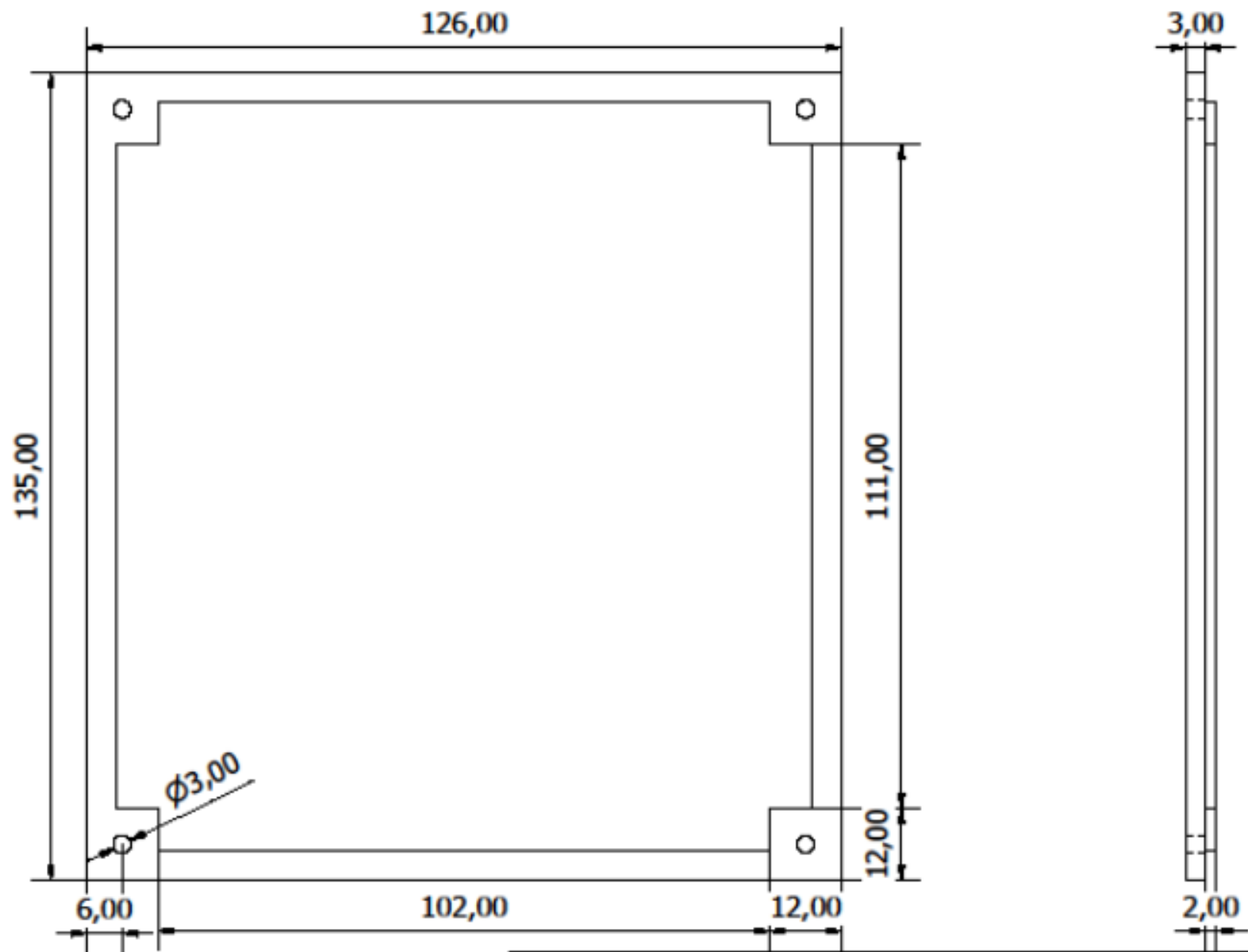
Plano 2. Tapa de la caja protectora del circuito emisor

Plano 3. Caja protectora del circuito receptor

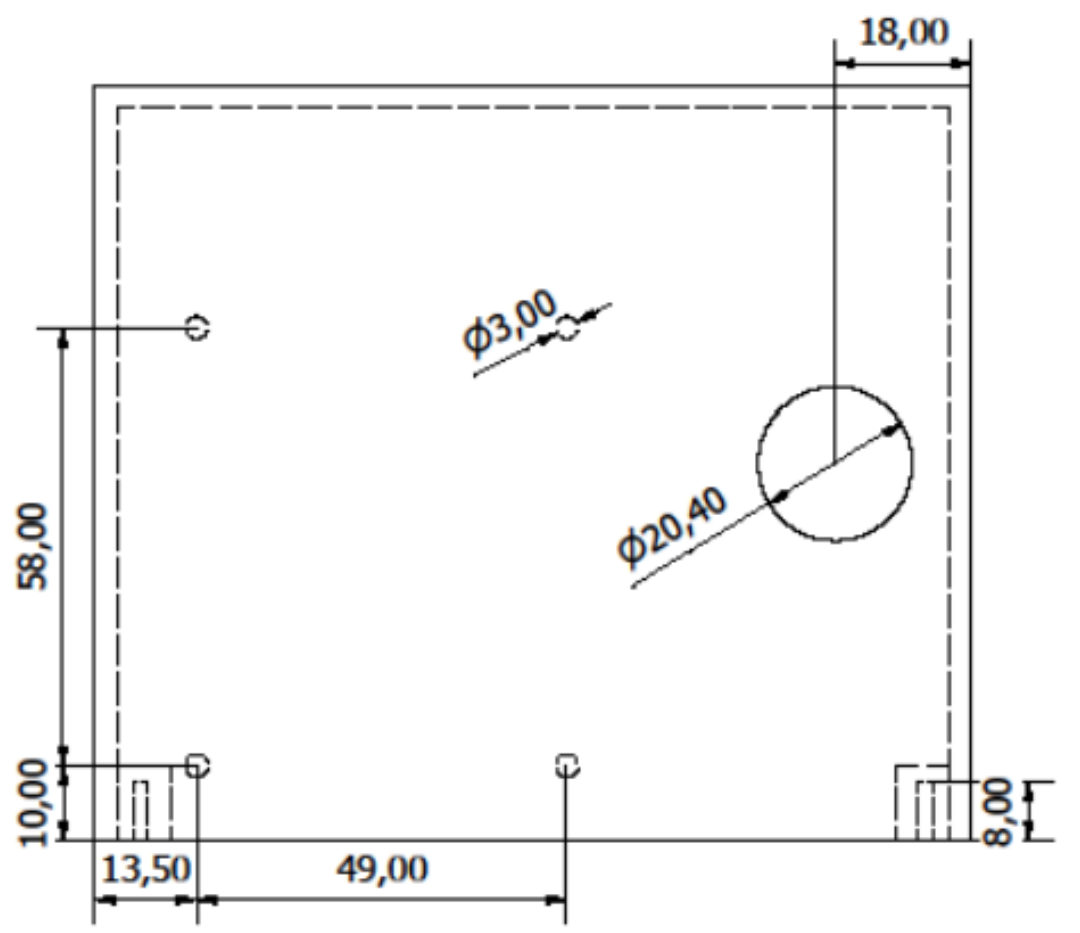
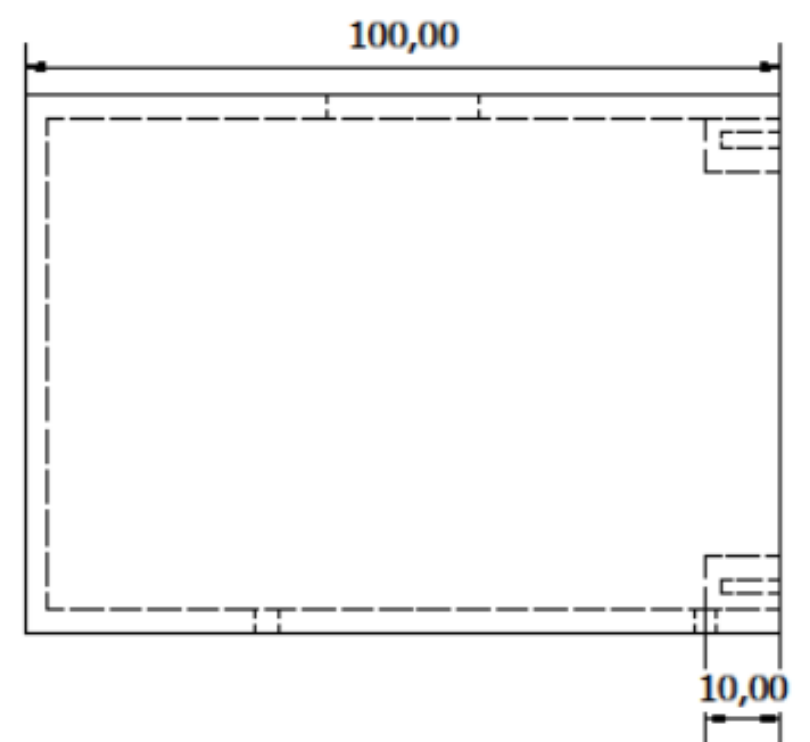
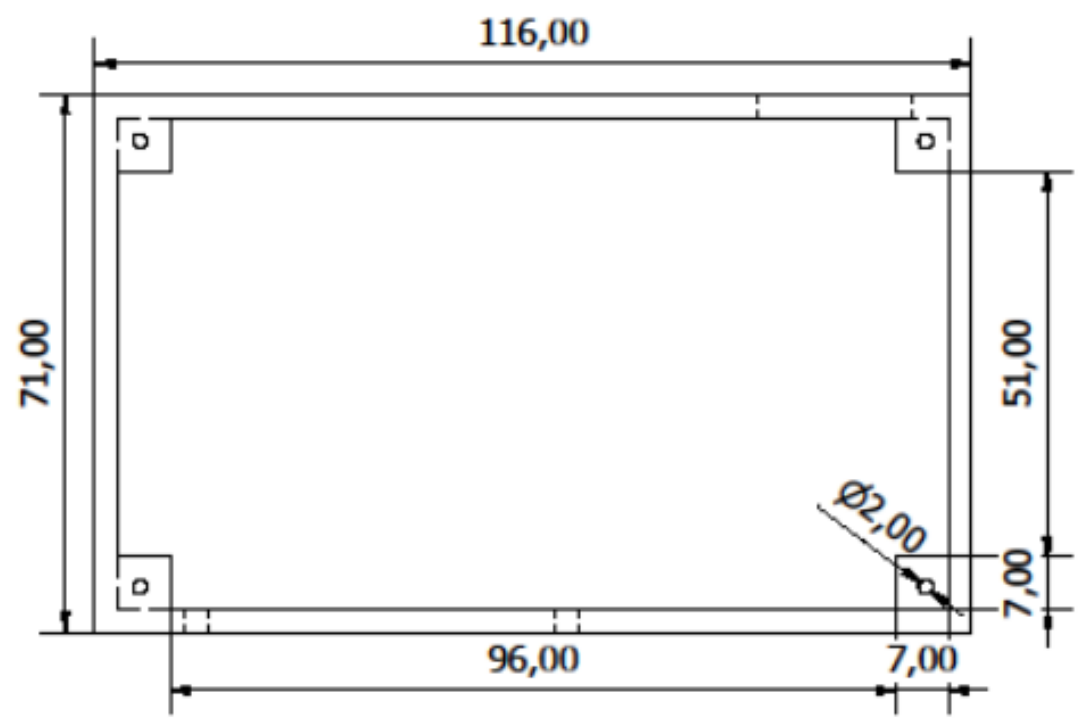
Plano 4. Tapa de la caja protectora del circuito receptor



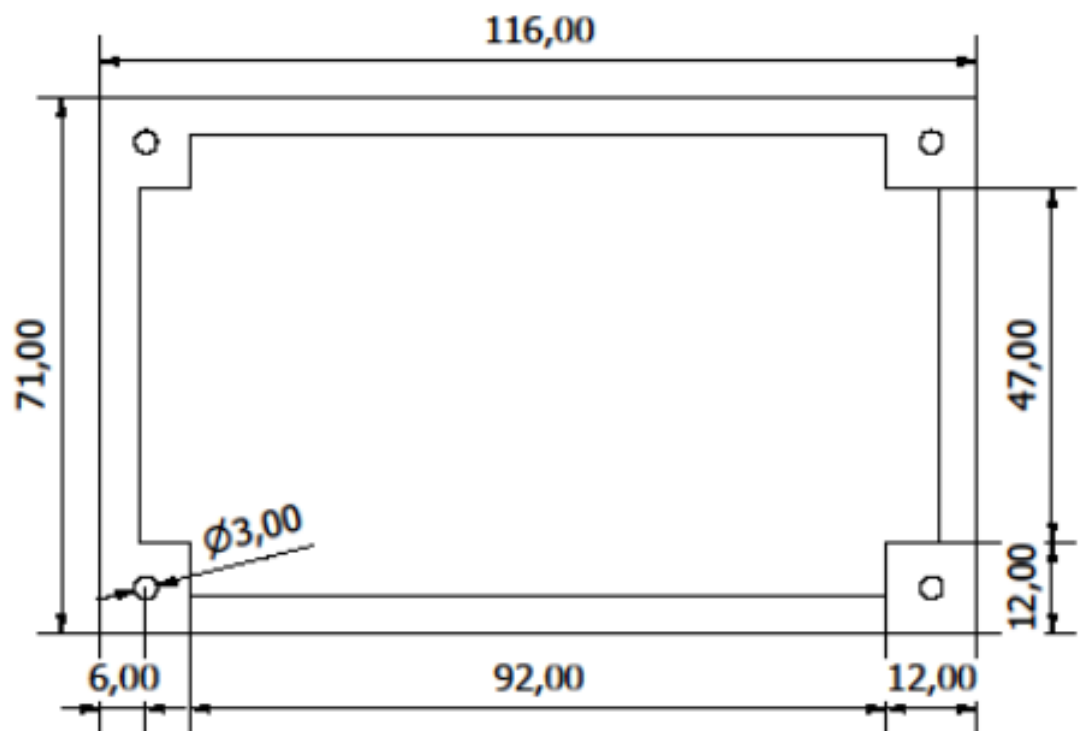
Diseño de		Fecha		Plano 1	
Melissa Lissette Banchón Valdiviezo		21/08/2023			
Proyecto integrador		Protector circuito emisor			
		Edición	Hoja		
		1	1		2



Diseño de Melissa Lissette Banchón Valdiviezo		Fecha 21/08/2023	Plano 2
Proyecto integrador		Protector circuito emisor	
		Edición 1	Hoja 2 / 2



Diseño de	Fecha	Plano 3
Melissa Lissette Banchón Valdiviezo	21/08/2023	
Proyecto integrador	Protector circuito receptor	
1	Edición	Hoja
	1	1 / 2



Diseño de Melissa Lissette Banchón Valdiviezo		Fecha 21/08/2023	Plano 4
Proyecto integrador		Protector circuito receptor	
1		Edición 1	Hoja 2 / 2