UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Proyecto de Sistema de climatización automatizado

Autor(es): **Angie Martinez**
Polette Montt
Bastian Sucso

Asignatura: **Proyecto 2**

Profesor(es): **Diego Alberto Aracena Pizarro**

Historial de versiones

Fecha	Versión	Descripción	Autor(es)
30/10/2024	1.0	Inicio del Documento	Angie Martinez Polette Montt Bastian Sucso
04/09/2024	1.1	Continuación del Documento	Angie Martinez Polette Montt Bastian Sucso
05/09/2024	1.2	Revisiòn y finalización del Documento	Angie Martinez Polette Montt Bastian Sucso
22/10/2024	1.3	Modificación del documento inicial	Bastian Sucso
27/10/2024	1.4	Inicio de la parte dos del documento	Angie Martinez Polette Montt Bastian Sucso
02/11/2024	1.5	Continuación del Documento dos	Angie Martinez Polette Montt Bastian Sucso
04/11/2024	1.6	Revisión final del Documento dos	Angie Martinez Polette Montt Bastian Sucso

Índice

1. Panorama General

1.1. Introducción

El presente proyecto surge ante la necesidad de un sistema autónomo de ventilación en entornos residenciales, dado que una gestión ineficiente de la climatización puede generar condiciones interiores desfavorables. Esta situación impacta directamente en el bienestar de los habitantes y puede tornarse especialmente peligrosa en climas extremos, comprometiendo la salud y la seguridad, en particular de las personas más vulnerables.

Mediante el presente informe se da a conocer el diseño y la organización del proyecto que se desarrollará durante el semestre en la asignatura de proyecto II.

1.2. Resumen del Proyecto

1.2.1 Propósito

Desarrollar un sistema de climatización automatizado que utiliza una Raspberry Pi para gestionar sensores de temperatura internos y externos, optimizando el confort de los residentes. También incluirá funcionalidades para la predicción de fallas mediante el monitoreo constante de los equipos de climatización.

1.2.2 Alcance

El proyecto se enfocará en la instalación de sensores de temperatura conectados a una Raspberry Pi, que procesará los datos y ajustará el sistema de climatización del hogar de forma automática.

1.2.3 Objetivos

1.2.3.1 Objetivo general:

Implementar un sistema de climatización automatizado utilizando una Raspberry Pi como controlador principal, que garantice un ambiente confortable.

1.2.3.2 Objetivos específicos:

- Instalación de sensores: Utilizar sensores de temperatura para medir tanto el ambiente interno como externo.
- Implementación de la Raspberry Pi: Programar la Raspberry Pi para procesar los datos recibidos de los sensores y ajustar los sistemas de climatización automáticamente.
- Interfaz de usuario: Crear una interfaz simple para que los usuarios puedan monitorear y ajustar parámetros del sistema de climatización desde un dispositivo conectado, como un móvil o una computadora

1.2.3.3 Suposiciones:

- **Monitoreo de Temperatura:**
  El sistema utilizará sensores de temperatura para medir continuamente la temperatura ambiente en tiempo real.
- **Control Automático de Climatización:**
  El sistema ajustará automáticamente los dispositivos de calefacción o enfriamiento (aire acondicionado con bomba de calor) para mantener la temperatura dentro de un rango predefinido.
- **Interfaz de Usuario:**
  Se desarrollará una interfaz de usuario accesible desde dispositivos móviles o computadoras, permitiendo a los usuarios configurar parámetros de temperatura y visualizar el estado actual del sistema

1.2.3.4 Restricciones:

- El desarrollo del proyecto se limitará a las herramientas y recursos previamente especificados en el plan.
- El cumplimiento de los plazos establecidos es fundamental para el éxito del proyecto.
- El proyecto deberá llevarse a cabo con los recursos y materiales disponibles, respetando el presupuesto asignado sin incurrir en gastos adicionales.

1.2.4 Entregables del Proyecto

Durante el transcurso del proyecto, se deberán entregar ciertos documentos para mantener actualizada la información sobre los avances y la organización. Los documentos que se entregarán incluyen:

- **Registros de Actividades**: Tras cada reunión de trabajo, se subirá a Redmine un registro detallado de los progresos logrados durante la sesión.
- **Reportes**: Se elaborará un reporte al finalizar cada fase del proyecto, el cual documentará en detalle el desarrollo del mismo.
- **Presentaciones en PowerPoint**: Se crearán presentaciones que cubren los temas tratados en los informes y que se ajusten a los requerimientos del profesor.
- **Bitácoras**: Por cada reunión de trabajo se sube a redmine una bitácora informando lo que se avanzó durante la misma.
- **Informes:** Se realiza un informe por fases, que contiene todo el proyecto documentando su desarrollo

2. Organización del proyecto

2.1. Personal

Cargos	Encargado(s)	Remuneración por hora de trabajo	Horas trabajadas semanales	Remuneración total (4 meses)
Jefe de proyecto	Angie Martinez	$15.000	10 hrs	$2.400.000
Programador	Polette Montt	$11.520	10 hrs	$1.843.200
Ensamblador	Bastian Sucso	$7.200	10 hrs	$1.152.000
Documentador	Bastian Sucso	$3.400	10 hrs	$544.000
			Total	$5.939.200

tabla 1. Asignación y costo del personal.

2.2. Roles y responsabilidades

- **Director de Proyecto**: Responsable de la coordinación y supervisión general del proyecto.
- **Desarrollador de Software**: Se encarga del análisis, desarrollo y despliegue de la aplicación sobre el hardware proporcionado para el proyecto.
- **Especialista en Ensamblaje**: Diseño, instalación y encargado de montar el hardware necesario para la realización del proyecto.
- **Responsable de Documentación**: Encargado de crear y mantener la documentación general del proyecto, incluyendo informes y wiki.

2.3. Mecanismos de comunicación

Comunicación del equipo

La comunicación del equipo se gestionó principalmente a través de WhatsApp para la coordinación de reuniones, mientras que Discord se utilizó para realizar las reuniones virtuales y compartir los avances del proyecto.

Informes y trabajos realizados

Para la elaboración de informes y creación de presentaciones, se utilizó Google Docs como herramienta principal

3. Planificación de los procesos de gestión

3.1. Planificación inicial del proyecto

- **Planificación de estimaciones**

Costo de materiales:

1 cartón piedra Cartón Piedra	$1500
Raspberry Pi Raspberry Pi 4	$53.149
Protoboard Protoboard	$1490
Resistores Resistores	$1041
Cable Jumper Dupont Line-Cable	$1000
Módulo de relé Módulo de relé de 5v	$500
Relé Relé de potencia electromagnético MY4NJ,	$1800
Adaptador Wi-Fi es.aliexpress.com/item/1	$2900
2 barras de silicona	$400
Sensores de temperatura Medidor de Sensor de temperatura Digital	$3700
2 Luz Infrarroja Coche LED T10 W5W	$3300
2 Ventiladores Ventilador ESC de Motor	$5400

Materiales Decorativo	$5000
2 computadores Lenovo V14 G2 ALG LENOVO Lenovo V14 G2	$860.000
VivoBook_ASUS ASUS Noteboo	$499.990
total	$1.441.170

tabla 2. Costo de materiales

Costo de personal:

Jefe proyecto	$2.400.000
Ensamblador	$1.152.000
Programador	$1.842.200
Documentador	$544.000

tabla 3. Costo de personal

Costo total personal: $5.939.200

Costo total del proyecto: $7.380.370

- **Planificación de Recursos Humanos.**

    - Jefe de proyecto.
    - Ensamblador.
    - Técnico en instalaciones.
    - Programador.
    - Documentador

3.2. Lista de actividades (Carta Gantt)

- ### Actividades de trabajo
  En la siguiente actividad, se desarrolló la carta gantt que contaba con la formulación de la parte inicial de proyecto, donde se incluye la definición del problema, plantear resolución de la problemática, definir los roles, asignar responsabilidades.

- ### Asignación de tiempo
  Planificación del proyecto: 7 semanas.

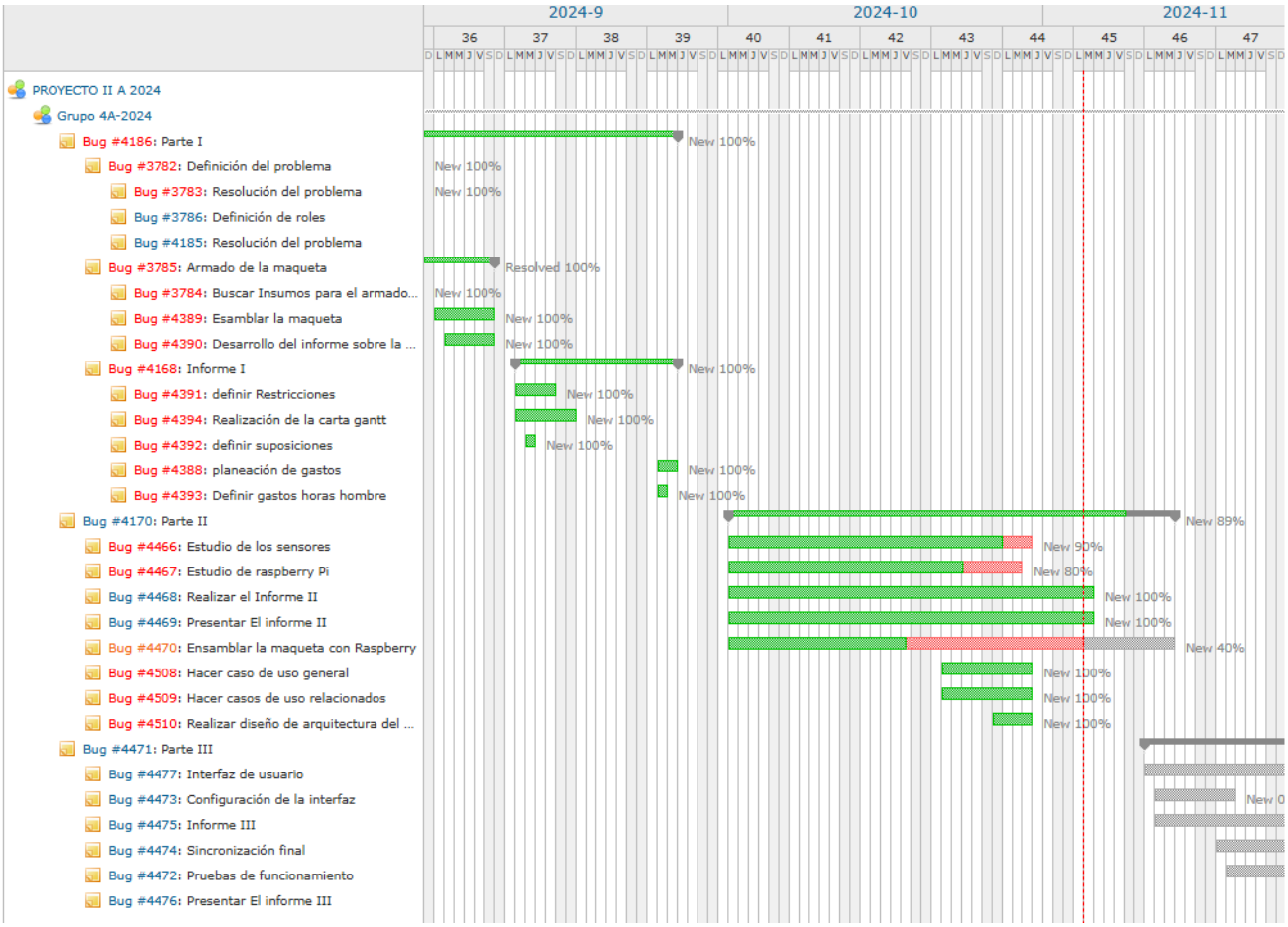


Imagen 1. Carta gantt.

3.3. Planificación de la gestión de riesgos

RIESGOS	PROBABILIDAD DE OCURRENCIA	NIVEL DE IMPACTO	ACCIÓN REMEDIAL
1.- Retrasos en la adquisición de recursos esenciales para el proyecto.	50%	2	Ajustar las prioridades del cronograma mientras se espera la llegada de los recursos.
2.- Errores imprevistos durante el desarrollo de alguna fase.	30%	2	Corregir los errores inmediatamente y ajustar el plan de trabajo según sea necesario.
3.- Falta de comunicación efectiva entre los miembros del equipo.	10%	3	Implementar reuniones regulares y canales de comunicación abiertos para evitar malentendidos.
4.- Insuficiente experiencia en las herramientas y tecnologías necesarias.	70%	2	Programar tiempo extra fuera del proyecto para que los integrantes se capaciten adecuadamente.
5.- Discrepancias en la toma de decisiones dentro del equipo.	30%	2	Fomentar el diálogo abierto y llegar a consensos que no afecten el avance del proyecto.
6.- Ausencia de algunos miembros por problemas de salud o compromisos personales.	20%	2	Redistribuir el trabajo y ajustarse a las limitaciones de tiempo sin comprometer el avance global.
7.- Pérdida de información importante o entregable debido a fallas en almacenamiento.	20%	1	Hacer respaldos constantes en la nube y mantener copias de seguridad periódicas.

8.- Cancelación de sesiones presenciales debido a imprevistos.	15%	4	Organizar reuniones virtuales para mantener el flujo de trabajo.
9.- Sobrecarga de tareas en un integrante específico.	30%	2	Redistribuir la carga de trabajo para evitar bloqueos y mantener el ritmo del proyecto.
10.- Fallo de algún componente físico del proyecto.	20%	1	Sustituir el componente y redistribuir los costos entre los miembros si es necesario.
11.- Mala gestión del tiempo, provocando retrasos en las entregas.	30%	2	Usar herramientas de gestión de proyectos para organizar mejor el cronograma.

tabla 4. tabla de riesgos considerados.

3.4. Niveles de riesgo

1.- Catastrófico

2.- Crítico

3.- Marginal

4.- Despreciable

4. CASOS DE USO

Diagrama de casos de uso base:

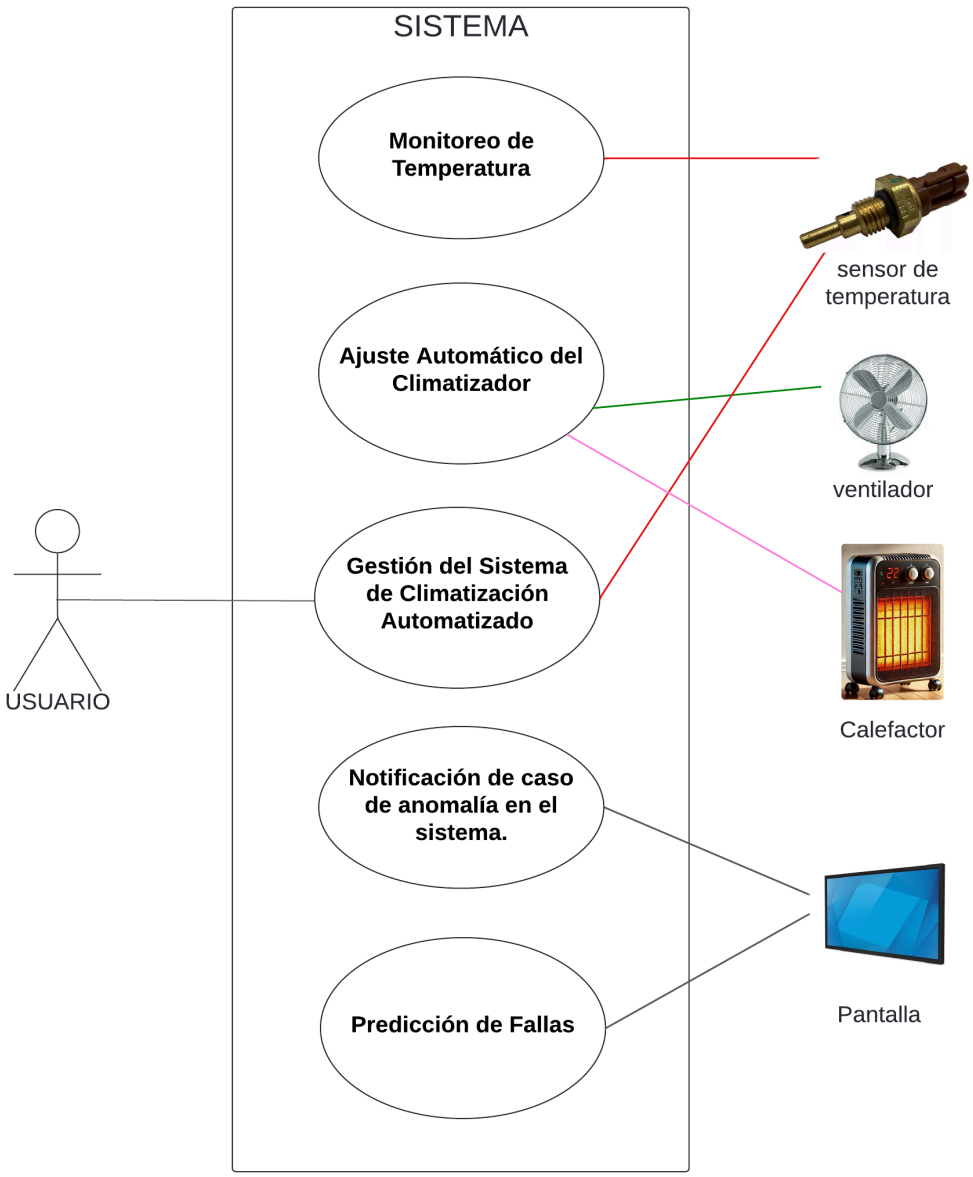


imagen 2. Diagrama general de casos de uso

5. MODELO DE DISEÑO (caso de uso general)

5.1 Gestión del Sistema de Climatización Automatizado

<table>
<tr><td colspan="2">Nombre CUS: Gestión del Sistema de Climatización Automatizado</td></tr>
<tr><td colspan="2">Resumen: El sistema monitorea continuamente la temperatura externa y ajusta automáticamente el climatizador. También permite la interacción del usuario para el encendido/apagado, configuración y obtención de datos de los estados de los dispositivos y el sistema.</td></tr>
<tr><td colspan="2">Actor: Usuario, Sensor de temperatura</td></tr>
<tr><td colspan="2">Precondición: Los sensores de temperatura están conectados y operativos.El sistema de climatización está en funcionamiento, y la Raspberry Pi está conectada y configurada.</td></tr>
<tr><td>Flujo Principal: Sistema de climatización

2. El usuario selecciona “Encender”</td><td>Flujo Principal: Sistema

1. El sistema muestra las siguientes opciones:
a. Encender/Apagar
b. Configurar Temperatura
c. Mostrar datos

3. El sistema activará los sensores</td></tr>
<tr><td>Flujo Alternativo:
2.1 El usuario selecciona “Configurar Temperatura”

4.1 El usuario ingresa el rango de temperatura</td><td>3.1 El sistema solicita ingresar el rango de temperatura adecuado

5.1 Guarda y actualiza la información.</td></tr>
<tr><td>Flujo Alternativo:
2.2 El usuario selecciona “Mostrar Datos”</td><td>3.2 El sistema muestra un resumen de la información del sistema; estado de los sensores; si el sistema funciona correctamente;</td></tr>
<tr><td colspan="2">Postcondiciones: —</td></tr>
</table>

5.2 Modelo de Secuencia del caso general

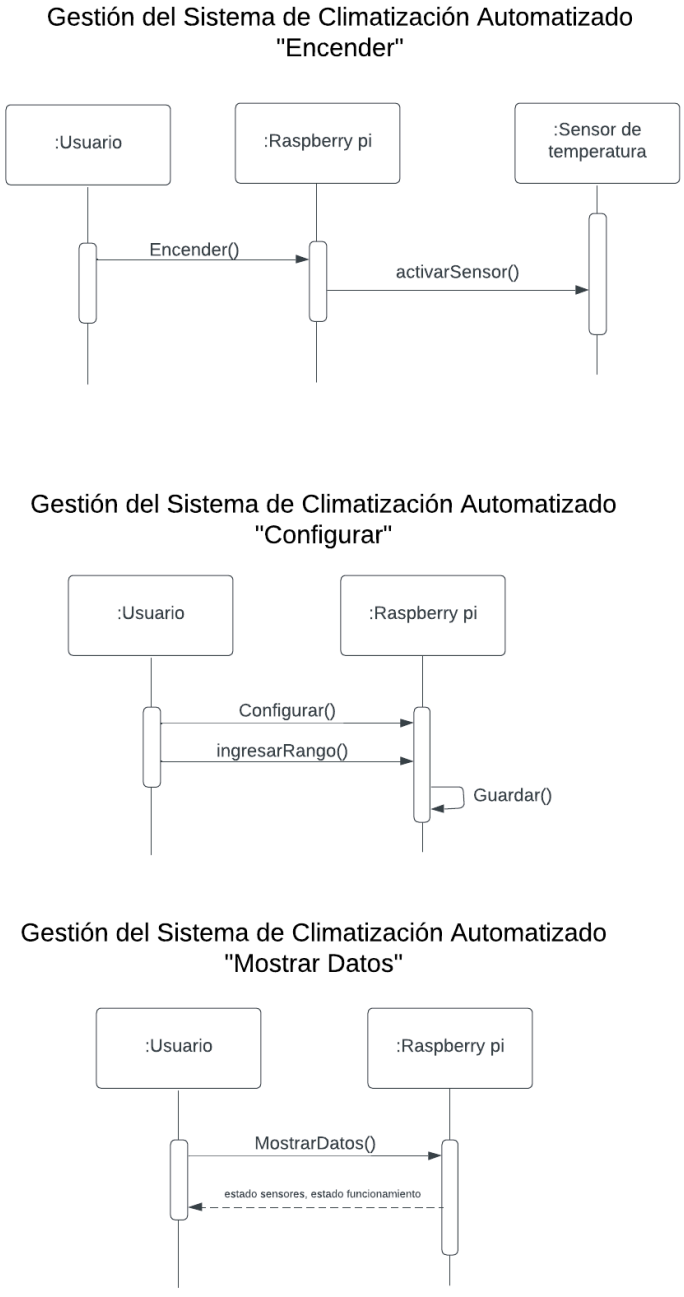


imagen 3. Diagrama de flujo, caso de uso: Gestión del sistema de climatización

5.1 Descripción de casos de uso:

5.1.1 Monitoreo de temperatura

1.

Nombre CUS: Monitoreo de Temperatura	
Resumen: El sistema registra las lecturas de temperatura externa a través de sensores.	
Actor: Sensor de Temperatura	
Precondición: Los sensores están conectados y funcionando.	
Flujo Principal: Sensor de temperatura 1. El sensor mide la temperatura (interna y externa) y envía los datos a la Raspberry Pi.	Flujo Principal: Sistema 2. La Raspberry Pi registra y almacena los datos
Flujo Alternativo:	
Postcondiciones: Se almacenan los datos de temperatura en el sistema.	

Monitoreo de Temperatura

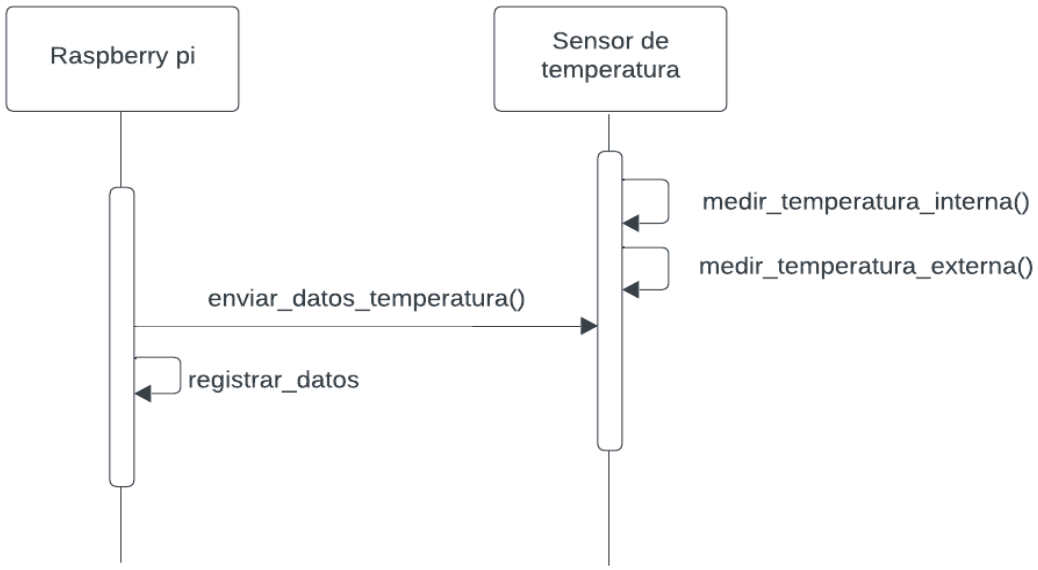


imagen 4. diagrama de flujo, caso de uso: Monitoreo de Temperatura

5.1.2 Ajuste Automático del Climatizador

Nombre CUS: Ajuste Automático del Climatizador	
Resumen: La Raspberry Pi ajusta el sistema de climatización basado en las lecturas de temperatura.	
Actor: Ventilador, Calefactor	
Precondición: El sistema tiene acceso a los datos de temperatura y el rango de confort	
Flujo Principal: 2. Activa el Ventilador	Flujo Principal: Sistema 1. La Raspberry Pi compara los datos actuales con el rango de confort.
Flujo Alternativo: *Si temperatura < rango confort* 2.1 Activa el Calefactor	
Postcondiciones: —	

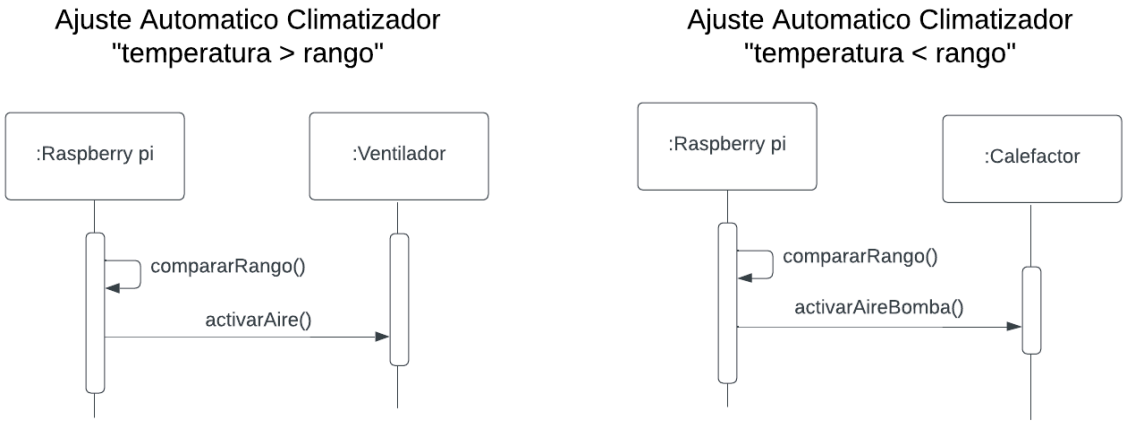


imagen 5. diagrama de flujo, caso de uso: Ajuste Automático del Climatizador

5.1.3 Predicción de Fallas

Nombre CUS: Predicción de Fallas	
Resumen: Monitoreo continuo del sistema de climatización para anticipar fallos.	
Actor: Pantalla	
Precondición: Se cuenta con la información entregada por los sensores.	
Flujo Principal: Pantalla 2. Mostrar alerta en caso de Anomalía	Flujo Principal: Sistema 1. La Raspberry Pi analiza patrones de uso y respuesta de los sensores.
Flujo Alternativo: Sin anomalía 2.1 Mostrar un mensaje indicando que se encuentra correctamente.	
Postcondiciones: —	

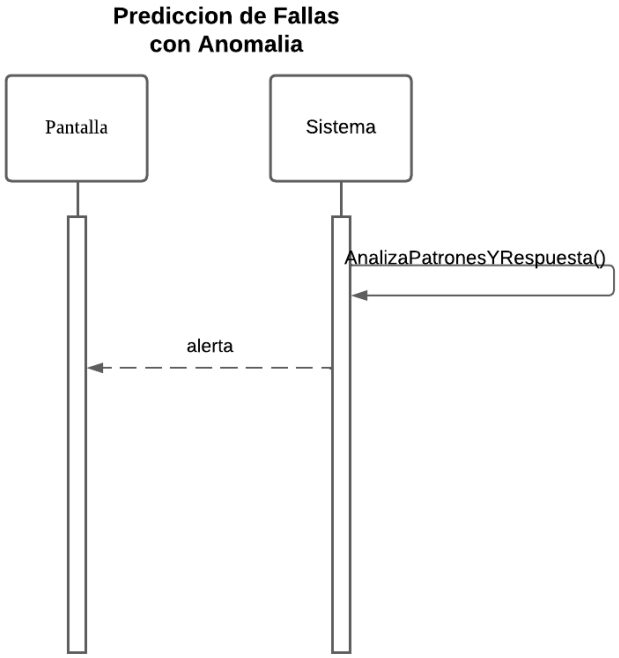


imagen 6. diagrama de flujo, caso de uso: predicción de fallas con anomalía

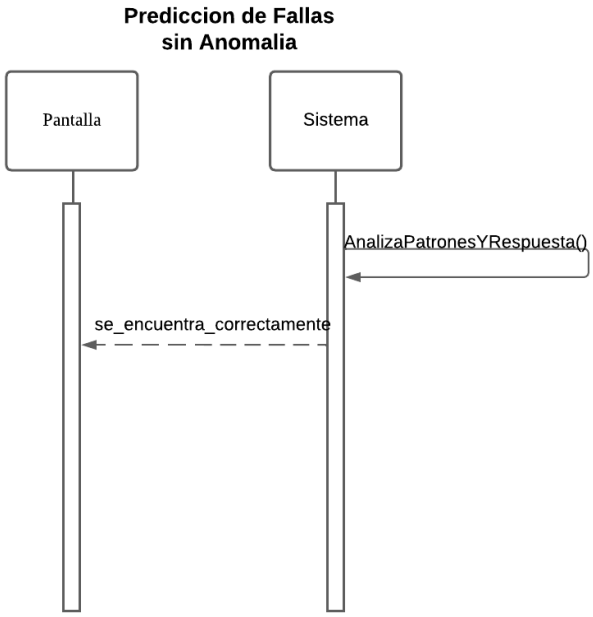


imagen 7. diagrama de flujo, caso de uso: predicción de fallas sin anomalía

5.1.4 Notificación de caso de anomalía en el sistema

Nombre CUS: Notificación de caso de anomalía en el sistema	
Resumen: En caso de anomalías, el sistema alerta al usuario	
Actor: Pantalla	
Precondición: El sistema detecta anomalías en el sistema de monitoreo	
Flujo Principal: Pantalla 2. Muestra una notificación al usuario por medio de la pantalla.	Flujo Principal: Sistema (Raspberry pi) 1. La Raspberry Pi detecta una anomalía.
Flujo Alternativo:	
Postcondiciones: El usuario recibe la notificación.	

Notificación de caso de anomalía en el sistema

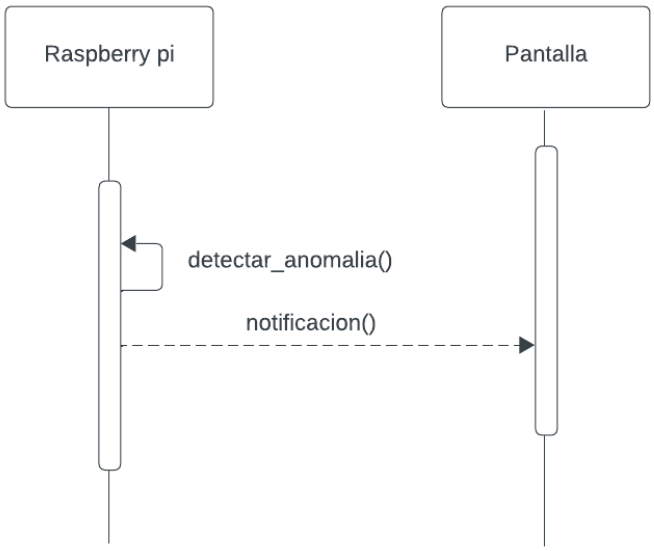


imagen 8. diagrama de flujo, caso de uso: notificación de caso de anomalía

6. Descripción de la Arquitectura

La Raspberry Pi es el núcleo controlador que aloja toda la lógica del sistema, mientras que interactúa con el sistema a través de la pantalla de usuario.

A continuación, se describe en detalle el papel de cada componente y cómo se interrelacionan.

1. Raspberry Pi microcontrolador/ cerebro del sistema. Recibe información de los sensores y toma decisiones sobre cómo ajustar la temperatura. Inserta el microcontrolador que tomará información de los sensores y del módulo de comunicación.
2. Sensor, tenemos: sensor de temperatura; mide la temperatura en el entorno monitoreado. Envía datos de temperatura en tiempo real. Para que puedas ajustar la climatización según sea necesario.
3. Pantalla; la interfaz de usuario con la que el cliente interactúa. Modifique los rangos de temperatura a su gusto, lapsos de tiempo y configuraciones varias del sistema y sus sensores.

Flujo de funcionamiento

1. El monitoreo continuo por parte de la Raspberry Pi de las lecturas tomadas por el sensor de temperatura; si la biometría que utiliza el sistema indica que la temperatura está fuera del rango deseado, puede usar la Raspberry Pi para activar el sistema de climatización.
2. El usuario final puede observar y seguir el curso de las mediciones en la pantalla.
3. En caso de detectar anomalías en los sensores, en primera instancia, el usuario recibe una notificación vía Raspberry Pi en la interfaz de la pantalla, lo que le permite mantenerse siempre bajo control en caso de situaciones anormales y realizar acciones.

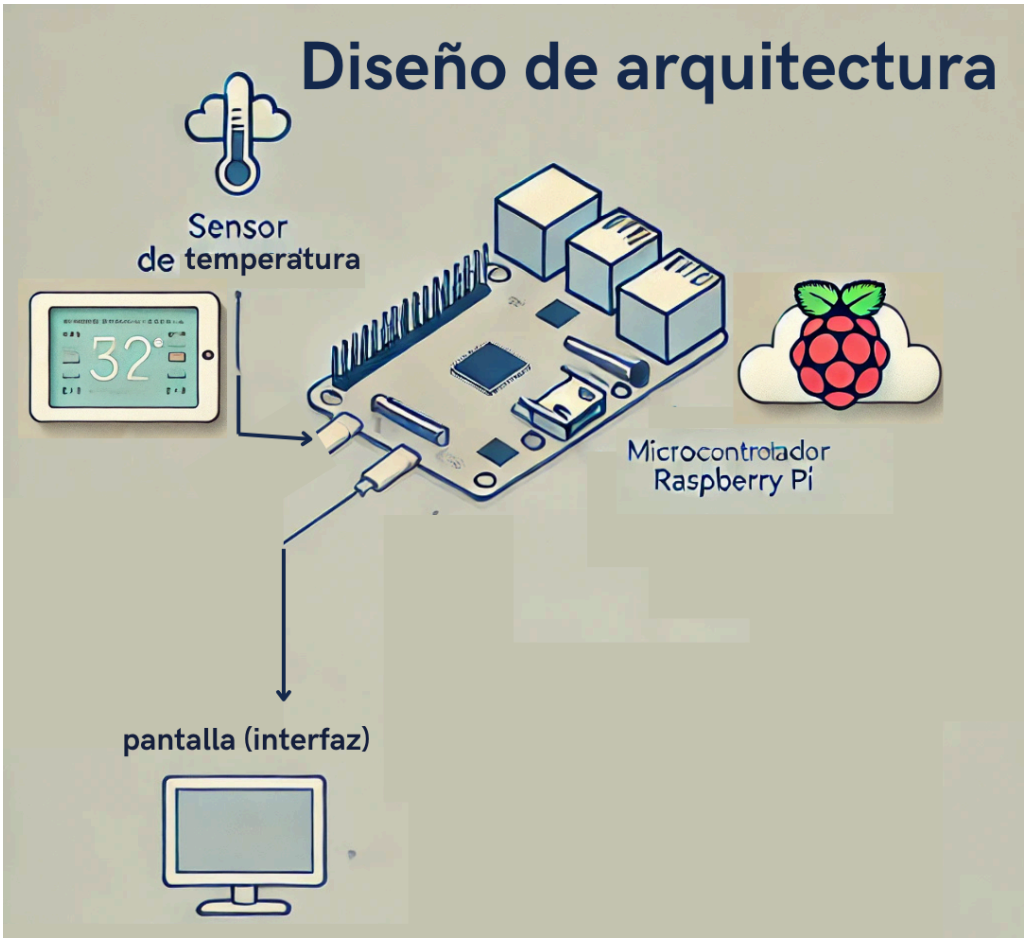


imagen 9. diseño de arquitectura

7. Documento de Diseño de Interfaz Usuario

7.1 Descripción de Interfaz de usuario

Para el diseño de la interfaz de usuario del sistema de climatización automatizado, se pueden considerar los siguientes elementos basados en las características descritas en el informe:

- **Pantalla Principal**:
  - **Resumen de estado**: Una sección que muestre en tiempo real la temperatura externa, el estado del sistema (activo/inactivo), y cualquier alerta de fallas o necesidades de mantenimiento.
  - **Gráfica de temperatura**: Un gráfico de línea que refleje el historial de temperaturas recientes, permitiendo al usuario visualizar cambios y tendencias en el corto plazo.
- **Controles Manuales**:
  - **Botón de encendido/apagado** del sistema de climatización.
  - **Campo de rango de temperatura interna**: Dos cajas de texto para que el usuario ingrese el rango de temperatura deseado (por ejemplo, 18-24°C).
  - **Botón de Guardar**: Guarda los ajustes y establece automáticamente el rango ingresado como límite de operación para la temperatura interna.
- **Sección de Mantenimiento Predictivo**:
  - **Indicador de estado del sistema**: Informa si el sistema está funcionando correctamente o si se ha detectado algún problema.

Esta estructura de interfaz asegura una experiencia intuitiva y funcional para el usuario, cumpliendo con los objetivos de monitoreo y control accesible desde cualquier dispositivo, según lo especificado en el informe del sistema.

7.2 Interfaz de usuario

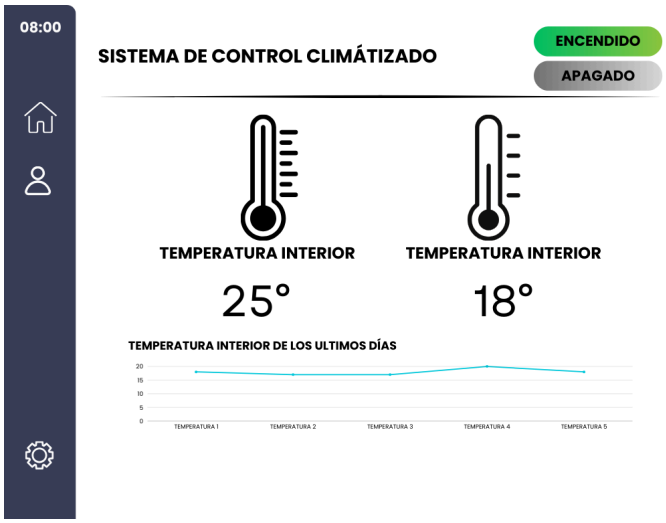


imagen 10. página home

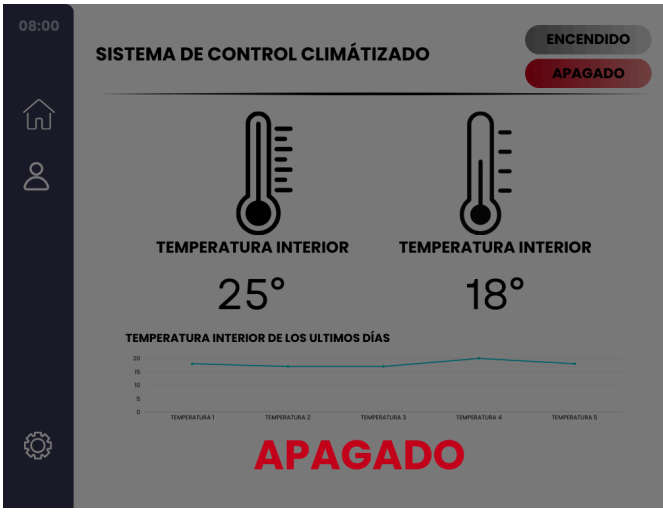


imagen 11. página apagada

imagen 12. Configuración de rangos de temperatura temperatura

imagen 13. Configuración de rangos de

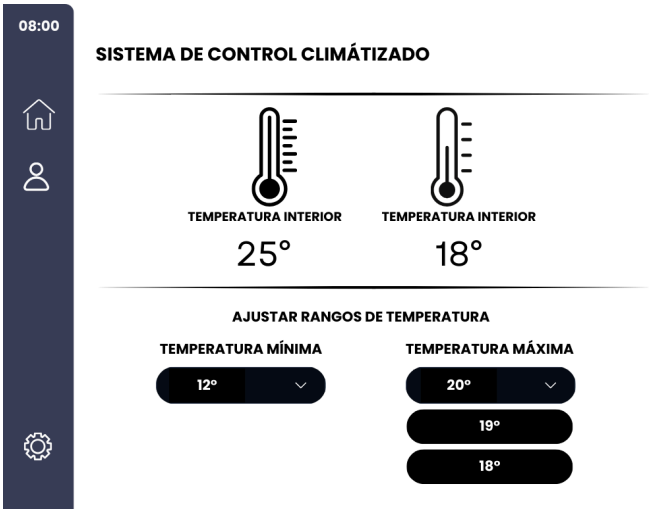


imagen 14. Configuración de rangos de temperatura

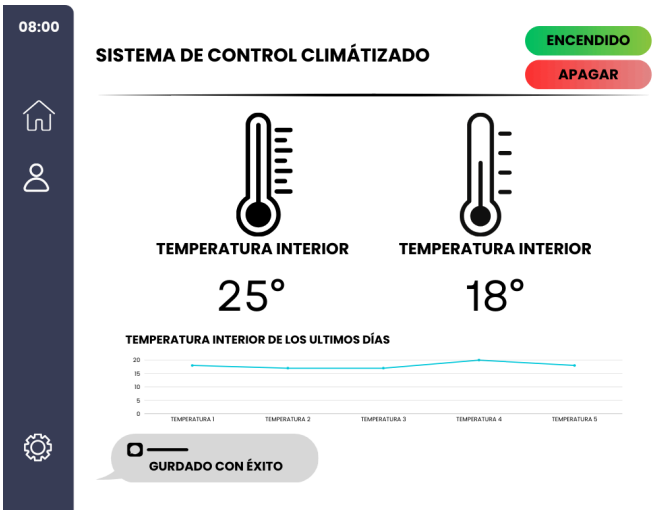


imagen 15. Configuración exitosa

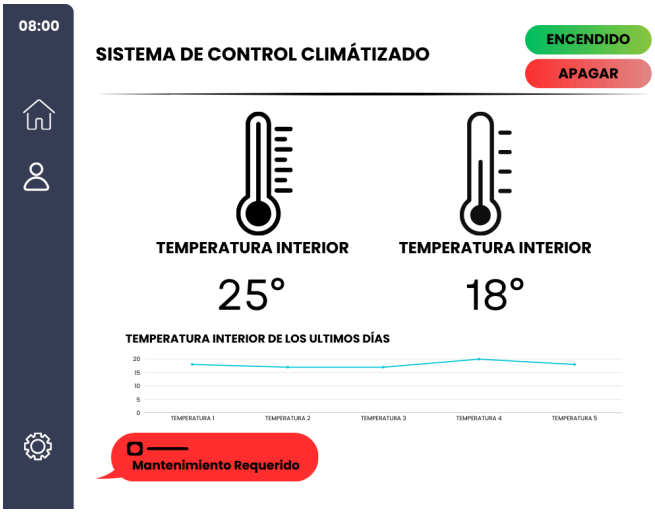


imagen 16. Error notificado

imagen 17. Información del error notificado

8. Especificación de Requerimientos

8.1 Requerimientos Funcionales

- **Monitoreo de Temperatura**
    - El sistema debe medir la temperatura externa utilizando sensores de temperatura conectados a una Raspberry Pi.
    - El sistema debe mostrar la temperatura externa en tiempo real en la interfaz de usuario.
- **Control de Climatización**
    - El sistema debe permitir al usuario encender y apagar el sistema de climatización a través de un botón en la interfaz.
    - El sistema debe permitir que el usuario ingrese un rango de temperatura interna deseado mediante dos campos de entrada: uno para la temperatura mínima y otro para la máxima.
    - Al seleccionar el botón de "Guardar Rango", el sistema debe aplicar y actualizar el rango de temperatura.
- **Mantenimiento Predictivo**
    - El sistema debe monitorear el estado de los equipos de climatización y enviar una alerta en caso de detectar fallas potenciales o necesidades de mantenimiento.
- **Interfaz de Usuario**
    - La interfaz debe mostrar en una sola pantalla el estado del sistema, la temperatura interna y externa, y las alertas de mantenimiento.

8.2 Requerimientos No Funcionales

- **Rendimiento**
    - El sistema debe actualizar los datos de temperatura y estado del sistema con un retraso máximo de 3 segundos.
- **Usabilidad**
    - La interfaz debe ser simple e intuitiva, con opciones de fácil acceso y navegación clara.
    - Debe incluirse una guía de usuario en línea o tutorial para los nuevos usuarios.
- **Confiabilidad**
    - El sistema debe estar operativo al menos el 98% del tiempo, garantizando un funcionamiento continuo y estable.

Estos requerimientos aseguran que el sistema no solo cumpla con sus funciones de monitoreo y ajuste automático, sino que también ofrezca una experiencia segura, rápida y accesible para los usuarios.

9. Plan de Integración

1. **Integración de Hardware y Software**
    - **Sensores de temperatura:** Verificar la conexión y calibración de los sensores con la Raspberry Pi.
    - **Raspberry Pi:** Instalar el sistema operativo, los controladores y las librerías necesarias para la comunicación con los sensores y actuadores.
    - **Actuadores (ventilador y calefactor):** Asegurar la conexión y funcionamiento bajo diferentes condiciones simuladas.
2. **Integración de la Interfaz de Usuario**
    - **Diseño y pruebas:** Validar que la interfaz permita monitorear y configurar el sistema desde dispositivos conectados.
    - **Pruebas de conectividad:** Garantizar que la Raspberry Pi y la interfaz se comuniquen de manera eficiente.
3. **Pruebas del Sistema Completo**
    - Simulación de diferentes rangos de temperatura para verificar los ajustes automáticos del climatizador.
    - Pruebas de mantenimiento predictivo y notificaciones ante fallos simulados.

10. Modelo de Implementación

Objetivo: Proveer una guía clara sobre cómo llevar a cabo la instalación y puesta en marcha del sistema en un entorno residencial.

- **Fases de Implementación**
    - **Fase 1: Instalación física**
        - Montaje de sensores en ubicaciones estratégicas del hogar.
        - Conexión de actuadores al sistema eléctrico y a la Raspberry Pi.
    - **Fase 2: Configuración inicial**
        - Configuración del rango de temperaturas en la interfaz de usuario.
        - Ajuste de los parámetros del mantenimiento predictivo.
    - **Fase 3: Pruebas funcionales**
        - Simular escenarios de temperatura para evaluar la respuesta del sistema.
        - Validar que las alertas y notificaciones funcionen correctamente.
- **Capacitación del Usuario Final**
    - Proveer un manual de usuario detallado y realizar una demostración práctica de las funciones principales.

11. Módulos Implementados

El sistema de climatización automatizado presenta una estructura en base a módulos bien delimitados que garantizan su funcionamiento:

Interfaz de Usuario:
La interfaz se desarrolló en tkinter y PIL para ofrecer una experiencia amigable. Permite visualizar la temperatura actual, establecer rangos deseados, recibir mensajes de error o alarmas en el caso de errores de funcionamiento.
Controlador central (Raspberry Pi):
Se implementó la biblioteca de RPi.GPIO, la cual controla la lógica del sistema haciendo reaccionar los actuadores, en este caso la luz LED y el ventilador, que permiten mantener las condiciones de temperatura fijadas por el usuario final.
Mantenimiento predictivo:
Monitoreos de patrones de funcionamiento en el tiempo para prevenir errores y aumentar la vida del sistema. Módulo que avisa anomalías a través del interfaz de usuario.
Sensores y actuadores:
Los sensores recogen información sobre las temperaturas externas. Los actuadores generan respuestas automáticas conforme las temperaturas que caen dentro del rango establecido.

12. Códigos

12.1 Código del sensor DS18B20

Conexión de la Raspberry Pi 3 al sensor de temperatura DS18B20. Además de la conexión del activador luz led, al cumplir la condición

temp.py

C: > Users > angie > temp.py

```python
import os
import glob
import time
import RPi.GPIO as GPIO

# Configuración del GPIO para el LED
LED_PIN = 17  # Cambia esto según el pin que estés usando
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

# Configuración del sensor
base_dir = '/sys/bus/w1/devices/'
device_folders = glob.glob(base_dir + '28*')

if len(device_folders) == 0:
    raise RuntimeError("No se encontraron sensores de temperatura!")

device_folder = device_folders[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    with open(device_file, 'r') as f:
        lines = f.readlines()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos + 2:]
        temp_c = float(temp_string) / 1000.0
```

imagen 18. Código Temperatura

temp.py 1

C: > Users > angie > temp.py > ...

```
26  def read_temp():
35          return temp_c
36      return None
37
38  try:
39      while True:
40          temp_c = read_temp()
41          print(f"Temperatura actual: {temp_c:.2f}°C")
42
43          # Control del LED según el rango de temperatura
44          if temp_c > 20.0:  # Cambia este valor al rango deseado
45              print("Temperatura baja, encendiendo LED")
46              GPIO.output(LED_PIN, GPIO.HIGH)  # Enciende el LED
47          else:
48              print("Temperatura normal o alta, apagando LED")
49              GPIO.output(LED_PIN, GPIO.LOW)  # Apaga el LED
50
51          time.sleep(1)
52  except KeyboardInterrupt:
53      print("Finalizando programa...")
54  finally:
55      GPIO.cleanup()  # Limpia la configuración de los pines
```

imagen 19. Código Temperatura

12.2 Interfaz Gráfica

temp.py 1 tk_temp.py 1

C: > Users > angie > tk_temp.py > ...

```python
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
import random


# Función para actualizar las temperaturas
def actualizar_temperatura():
    global rango_interno
    temp_interna = round(random.uniform(rango_interno[0], rango_interno[1]), 2)
    temp_externa = round(random.uniform(20, 30), 2)

    etiqueta_temp_interna.config(text=f"{temp_interna}°C", fg="blue")
    etiqueta_temp_externa.config(text=f"{temp_externa}°C", fg="green")


# Función para guardar el rango ingresado
def guardar_rango():
    global rango_interno
    try:
        rango_texto = entrada_rango.get()
        limites = list(map(float, rango_texto.split("-")))
        if len(limites) == 2 and limites[0] < limites[1]:
            rango_interno = limites
            rango_label.config(text=f"Rango de Temperatura Interna: {limites[0]}°C - {limites[1]}°C")
        else:
            rango_label.config(text="Error: Formato inválido. Use el formato Min-Max.")
    except ValueError:
        rango_label.config(text="Error: Formato inválido. Use el formato Min-Max.")

```

imagen 20. Código Interfaz

```
rango_interno = [15, 25]

# Crear la ventana principal
root = tk.Tk()
root.title("Control de Temperatura")
root.geometry("800x600")

# Cargar la imagen de fondo
bg_image = Image.open("fondo.png")  # Asegúrate de tener un archivo llamado fondo.png en la misma ca
bg_image = bg_image.resize((800, 600), Image.LANCZOS)
bg_photo = ImageTk.PhotoImage(bg_image)

# Crear etiqueta para la imagen de fondo
bg_label = tk.Label(root, image=bg_photo)
bg_label.place(relwidth=1, relheight=1)

# Crear marco principal para los controles
frame = tk.Frame(root, bg="white", bd=2)
frame.place(relx=0.5, rely=0.5, anchor="center", width=400, height=300)

# Etiqueta principal
titulo = tk.Label(frame, text="Control de Temperatura", font=("Arial", 16, "bold"), bg="white")
titulo.pack(pady=10)

# Temperatura interna
etiqueta_interna = tk.Label(frame, text="Temperatura Interna", font=("Arial", 12), bg="white")
etiqueta_interna.pack()
etiqueta_temp_interna = tk.Label(frame, text="--°C", font=("Arial", 24), bg="white", fg="blue")
etiqueta_temp_interna.pack()

# Temperatura externa
etiqueta_externa = tk.Label(frame, text="Temperatura Externa", font=("Arial", 12), bg="white")
etiqueta_externa.pack()
```

imagen 21. Código Interfaz

```
63
64    # Temperatura externa
65    etiqueta_externa = tk.Label(frame, text="Temperatura Externa", font=("Arial", 12), bg="white")
66    etiqueta_externa.pack()
67    etiqueta_temp_externa = tk.Label(frame, text="--°C", font=("Arial", 24), bg="white", fg="green")
68    etiqueta_temp_externa.pack()
69
70    # Rango de temperatura interna
71    rango_label = tk.Label(frame, text=f"Rango de Temperatura Interna: {rango_interno[0]}°C - {rango_interno[1]}°C",
72                           font=("Arial", 10), bg="white")
73    rango_label.pack(pady=5)
74    entrada_rango = tk.Entry(frame, font=("Arial", 12))
75    entrada_rango.pack()
76
77    # Botones
78    guardar_btn = ttk.Button(frame, text="Guardar Rango", command=guardar_rango)
79    guardar_btn.pack(pady=5)
80    actualizar_btn = ttk.Button(frame, text="Actualizar Temperatura", command=actualizar_temperatura)
81    actualizar_btn.pack(pady=5)
82
83    # Iniciar bucle principal
84    root.mainloop()
```

imagen 22. Código Interfaz

13. Comparación de Requerimientos.

Una manera efectiva de evaluar el desempeño del grupo es realizar una comparación entre lo estipulado en los requisitos iniciales y el producto final. La tabla a continuación se detalla:

Requerimientos	Cumplido	No cumplido	Alternativa
El sistema debe medir y mostrar la temperatura externa utilizando sensores de temperatura conectados a una Raspberry Pi.	X		
Permitir al usuario encender y apagar el sistema	X		
Permitir que el usuario ingrese un rango de temperatura interna deseado mediante dos campos de entrada.	X		
Alerta en caso de detectar fallas potenciales o necesidades de mantenimiento.		X	
La interfaz debe mostrar en una sola pantalla el estado del sistema, la temperatura interna y externa, y las alertas de mantenimiento.	X		

Debe incluirse una guía de usuario en línea o tutorial para los nuevos usuarios.	X		
El sistema debe actualizar los datos de temperatura y estado del sistema con un retraso máximo de 3 segundos.	X		
El sistema debe estar operativo al menos el 98% del tiempo, garantizando un funcionamiento continuo y estable.	X		

14. Problemas Encontrados

Durante el desarrollo del sistema de climatización automatizado, se enfrentaron los siguientes desafíos técnicos:

1. **Compatibilidad del GrovePi**: El módulo no funcionó de manera óptima con los sensores seleccionados y la Raspberry Pi 4.
2. **Raspberry Pi 4**: Hubo problemas para lograr una comunicación estable entre la Raspberry Pi 4 y los sensores de temperatura y humedad.
3. **Sensores de Temperatura y Humedad**: Los sensores no respondían de forma consistente, lo que ocasionó fallas en la lectura de datos.

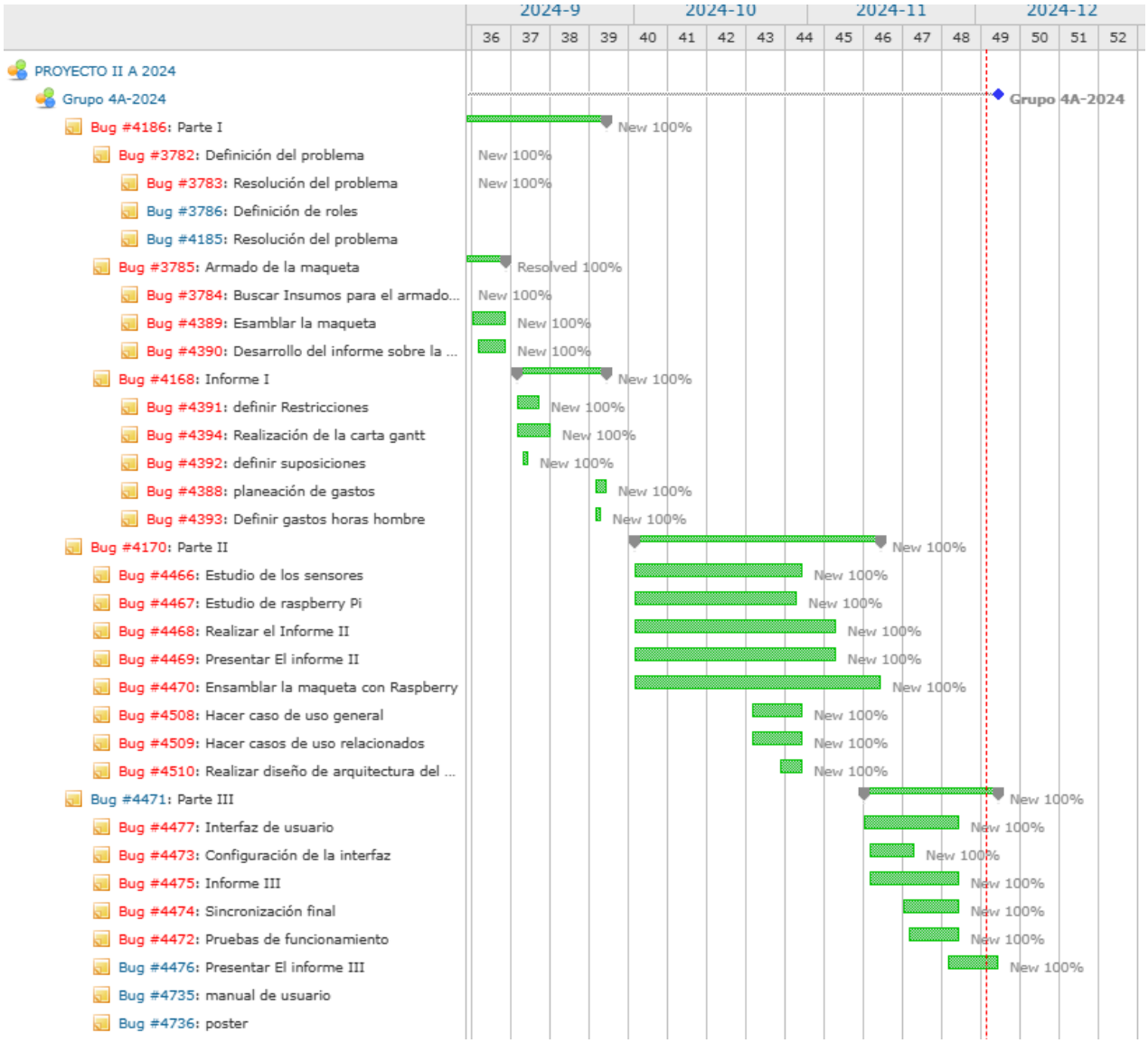
15. Soluciones Propuesta

Para abordar los problemas mencionados, el equipo implementó las siguientes soluciones:

1. **Sustitución de hardware**:
    - Se optó por usar una **Raspberry Pi 3** en lugar de la versión 4, logrando mayor estabilidad en las conexiones.
    - Se descartó el GrovePi y se trabajó directamente con una **protoboard** para facilitar las conexiones.
2. **Cambio de sensores**:
    - Se utilizó el sensor **DS18B20**, que ofreció lecturas más confiables para las pruebas del sistema.
3. **Indicadores visuales**:
    - Se incorporó una **luz LED roja** como indicador visual del estado del sistema, facilitando la identificación de errores o estados críticos.

Estas medidas permitieron avanzar en el desarrollo del proyecto y resolver los inconvenientes técnicos más críticos.

16. Actualización de la Carta Gantt



17. Trabajo Futuro

Con base en los resultados obtenidos, se identificó como la principal debilidad del presente proyecto la falta de cumplimiento adecuado en la integración entre el usuario y la Raspberry Pi mediante conexión Wi-Fi. Esta deficiencia genera dificultades significativas en términos de usabilidad y simplicidad para el usuario final. En futuros desarrollos, se busca abordar y resolver esta limitación, con el objetivo de optimizar sustancialmente la experiencia de usuario.

18. Conclusión

La implementación de un sistema de climatización automatizado con monitoreo y ajuste en tiempo real, como el desarrollado en este proyecto, representa una solución innovadora y eficiente para mejorar el confort en entornos residenciales. Este sistema aborda la necesidad de gestionar adecuadamente la climatización en interiores, optimizando tanto el bienestar de los habitantes como el uso de recursos energéticos. A través del uso de una Raspberry Pi que procesa datos de sensores internos y externos, el sistema permite mantener un rango de temperatura adecuado sin intervención manual, incrementando la eficiencia en el consumo de energía.

No obstante, durante el desarrollo del proyecto se enfrentaron desafíos técnicos significativos que afectaron el cumplimiento de los objetivos inicialmente propuestos. El principal obstáculo fue la conexión del sensor de temperatura con la Raspberry Pi, ya que el dispositivo no era detectado correctamente. Esto llevó al equipo a realizar múltiples pruebas, incluyendo el cambio de Raspberry Pi (de modelo 4 a modelo 3), la sustitución de GrovePi por protoboard, la instalación de diversas librerías, y el uso de diferentes sensores. Finalmente, se logró establecer la conexión, pero este proceso demandó una considerable cantidad de tiempo, generando retrasos significativos en el cronograma.

Como consecuencia de estos retrasos, no fue posible implementar adecuadamente la conexión remota entre el usuario y la Raspberry Pi mediante Wi-Fi, un componente clave para garantizar una experiencia de usuario amigable e intuitiva. Aunque el sistema base es funcional, la falta de esta funcionalidad limita su accesibilidad y versatilidad en su estado actual.

A pesar de estas limitaciones, el proyecto demuestra su viabilidad como una solución adaptable y escalable para la gestión de climatización en diversos entornos residenciales. La inclusión de mantenimiento predictivo sigue representando una ventaja significativa, al prevenir fallas y extender la vida útil de los equipos, minimizando los costos asociados a reparaciones imprevistas.

De cara al futuro, se recomienda priorizar mejoras en la planificación y gestión de riesgos técnicos, así como en la conectividad remota. Estas actualizaciones permitirán superar las limitaciones actuales y optimizar la experiencia del usuario. Con los ajustes adecuados, el sistema tiene el potencial de consolidarse como una herramienta efectiva y sustentable para la gestión de climatización en interiores.

19. Referencias

1. Ruiz, C. (2020). Implementación de sensores de temperatura con Raspberry Pi para control de climatización. *Revista Electrónica de Tecnologías Avanzadas, 5*(2). https://www.retavanzadas.com/implementacion-sensores-raspberry-pi-climatizacion
2. Pérez, A. (2021). Uso de sensores de temperatura y humedad para control inteligente de climatización con Raspberry Pi. *Electrónica y Hogar Inteligente*. https://www.electronicahogar.com/control-climatizacion-raspberry-pi
3. López, J. (2021, julio 12). Mantenimiento predictivo en sistemas de climatización usando Raspberry Pi. *Ingeniería y Automatización*. https://www.ingautomatizacion.com/mantenimiento-predictivo-climatizacion-raspberry
4. Smith, K. (2019). *Advanced Temperature Control with Raspberry Pi*. International Journal of IoT Applications, 7(3), 45-58. Recuperado de https://www.ijota.com/advanced-temperature-control-raspberry-pi
5. González, R. (2020). *Optimización de sistemas de climatización mediante machine learning y Raspberry Pi*. Innovación y Tecnología, 12(1), 20-33. Recuperado de https://www.innovtecnologia.com/climatizacion-machine-learning-raspberry-pi
6. Thompson, L. (2021). *Energy-efficient Climate Control using IoT Devices*. Journal of Sustainable Technologies, 9(4), 78-90. Recuperado de https://www.journalsts.com/energy-efficient-climate-iot
7. Martínez, F. (2020). *Diseño de un sistema de climatización autónomo basado en Raspberry Pi*. Tesis de Grado, Universidad Tecnológica. Recuperado de https://www.ut.edu/design-climate-system-raspberry-pi
8. Brown, T., & Green, H. (2021). *Integration of IoT for Smart Home Climate Systems*. Smart Home Journal, 14(2), 101-115. Recuperado de https://www.smarthomejournal.com/iot-climate-systems
9. Hernández, P. (2020). *Control de temperatura en edificios inteligentes utilizando Raspberry Pi*. Tecnología y Ciencia, 15(6), 120-130. Recuperado de https://www.tecnologiayciencia.com/temp-control-buildings-raspberry-pi
10. Wilson, J. (2022). *Predictive Analytics for Climate Control with Raspberry Pi*. Journal of Predictive Maintenance, 6(1), 60-72. Recuperado de https://www.jpredictivemaint.com/climate-control-analytics-raspberry-pi