UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Sistema Detector de Contaminación Acústica para el Hogar

“Ruidoso"

Autor(es): Diego Ferrada

Javier Huanca

Asignatura: Proyecto II

Profesor: Diego Aracena P.

ARICA, 3 diciembre 2024

Historial de Cambios

Tabla N°1: Fecha en que se han realizado cambios en el informe

Fecha	Versión	Descripción	Autor(es)
10/09/2024	1.1	Definición de la estructura del informe (primera fase)	Diego Ferrada Javier Huanca
24/09/2024	1.2	Finalización del informe (primera fase)	Diego Ferrada JavIer Huanca
28/09/2024	1.3	Revisión del informe (primera fase)	Diego Ferrada JavIer Huanca
29/10/2024	1.4	Corrección del informe (primera fase)	Diego Ferrada Javier Huanca
01/11/2024	1.5	Finalización del informe (segunda fase)	Diego Ferrada Javier Huanca
02/11/2024	1.6	Revisión del informe (segunda fase)	Diego Ferrada Javier Huanca
28/11/2024	1.7	Corrección del informe (segunda fase)	Diego Ferrada Javier Huanca
30/11/2024	1.8	Finalización del informe (tercera fase)	Diego Ferrada Javier Huanca
01/12/2024	1.9	Revisión del informe (tercera fase)	Diego Ferrada Javier Huanca

Nota: Muestra las fechas en que se han realizado cambios en el informe desde su creación hasta la fecha más reciente.

Índice de contenidos

Índice de tablas

Índice de figuras

1. Panorama General

1.1 Resumen del Proyecto

En zonas residenciales, el ambiente puede ser perturbado por ruidos fuertes provenientes tanto del interior como del exterior de los hogares. Estos ruidos pueden ser ladridos de un perro, obras en construcción cercanas o el volumen elevado de algún electrodoméstico. Lo que resulta perjudicial a la salud auditiva y al bienestar de quienes los sufren.

1.1.1 Propósito

El propósito de la realización de este proyecto es proteger la salud auditiva de los usuarios a través del monitoreo y control de ruidos fuertes, tanto del exterior como del interior del hogar.

El proyecto consta de un dispositivo programado con Raspberry Pi que, con la ayuda de varios sensores de ruido dispersados en el interior del hogar, indique el nivel de decibeles actual y, cuando este llegue a niveles dañinos (80dB o más), notifique al usuario para que este haga una acción al respecto.

1.1.2 Alcance

El proyecto, conformado por un dispositivo programado con Raspberry y sensores de ruido, cumple con las siguientes funcionalidades:

- Detectar ruidos del exterior e interior del hogar.
- Mostrar en el celular el nivel de decibeles actual en el ambiente.
- Notificar al usuario cuando el nivel de decibeles llegue a niveles dañinos.
- Mostrar consejos al usuario para tomar alguna acción respecto al ruido.
- Mostrar información respecto a los efectos de distintos niveles de decibeles en la salud auditiva.

1.1.3 Objetivo General

Desarrollar un sistema que detecte la contaminación acústica para el hogar, en situaciones en las que el nivel de decibeles en el ambiente es perjudicial para la salud auditiva, de tal manera que el usuario tome medidas oportunas.

1.1.4 Objetivos Específicos

- Estudiar y utilizar herramientas como Raspberry Pi y sensores de ruido para la implementación del sistema.
- Investigar la normativa respecto a la contaminación acústica en zonas residenciales para definir los parámetros del sistema de detección.
- Desarrollar un sistema de software que cumpla con las funcionalidades mencionadas anteriormente.
- Realizar pruebas del sistema, evaluando su rendimiento y precisión en la detección de contaminación acústica en diferentes situaciones.

1.1.5 Suposiciones

- Se asume que la conexión entre los sensores y la Raspberry Pi será estable, permitiendo que la transmisión de datos de los niveles de ruido se realice sin interrupciones.
- Se asume que los usuarios del sistema tienen conocimientos básicos para interpretar las recomendaciones proporcionadas por el dispositivo.
- Se asume que la aplicación móvil estará disponible en sistemas operativos comunes (Android), y que los usuarios tendrán acceso a un dispositivo móvil compatible para utilizar el sistema.

- Se asume que el sistema será escalable, permitiendo agregar más sensores en el futuro si el usuario desea monitorear áreas adicionales del hogar o espacios exteriores.

1.1.6 Restricciones

- El sistema debe estar desarrollado usando Raspberry Pi.
- Las limitaciones de espacio para la instalación de sensores, tanto en el interior como en el exterior del hogar, que podrían afectar la precisión en la detección de ruido en ciertas áreas.
- El sistema debe estar diseñado específicamente para ser utilizado en un entorno residencial.
- La utilidad del sistema depende de la presencia de ruido en el ambiente para funcionar, ya que su propósito es notificar y medir niveles de decibeles.

1.1.7 Entregables del proyecto

Los entregables del proyecto son los siguientes:

- Maqueta del proyecto.
- Bitácoras.
- Informes.
- Presentaciones.
- Wiki.
- Carta Gantt.
- Manual de usuario.
- Poster.

2. Organización del Proyecto

2.1 Personal y entidades internas

El equipo cuenta con roles que se enfocan en distintos ámbitos del proyecto:

- Jefe de proyecto: Encargado de representar el grupo, de dirigir las acciones y decisiones a tomar.
- Programador: Encargado de desarrollar el código del proyecto y de planificar su estructura lógica, además de testear su estado actual.
- Documentador: Encargado de escribir y redactar entregables que contienen información respecto al proyecto y las acciones tomadas en su desarrollo.

- Ensamblador: Encargado de integrar los componentes físicos como la Raspberry Pi y los sensores de ruido, asegurando que funcione correctamente.

2.2 Roles y Responsabilidades

Tabla N°2: Roles del equipo y sus responsables correspondientes

Rol	Responsables
Jefe de Proyecto	Diego Ferrada
Programador	Diego Ferrada Javier Huanca
Documentador	Diego Ferrada Javier Huanca
Ensamblador	Diego Ferrada Javier Huanca

Nota: Muestra qué miembro se encarga de cada rol existente en el proyecto.

2.3 Mecanismos de comunicación y organización

Principalmente se utilizan las plataformas Discord, WhatsApp, Google Drive y GitHub, facilitando la coordinación entre miembros.

- Discord se emplea para compartir archivos y realizar reuniones donde se discute respecto al desarrollo técnico del sistema.
- WhatsApp es utilizado para la comunicación rápida y diaria, permitiendo resolver dudas inmediatas, compartir actualizaciones, coordinar tareas y reuniones de manera ágil.
- Google Drive permite trabajar en tiempo real documentos y otros entregables del proyecto, así como compartirlos con el equipo completo.
- GitHub es usado para guardar los códigos a realizar en el proyecto y para que estos se puedan compartir con el equipo a modo de mejorar tales códigos.

3. Planificación de los procesos de gestión

3.1 Planificación inicial del proyecto

3.1.1 Planificación de estimaciones

Tabla N°3: Estimación de cada costo de Software

Costos de Software	Estimación
VS Code	$ 0 CLP
Python	$ 0 CLP
GitHub	$ 0 CLP
Raspberry Pi OS	$ 0 CLP
Total	$ 0 CLP

Nota: Muestra el costo estimado de cada pieza de Software usada en el proyecto.

Tabla N°4: Estimación de cada costo de Hardware

Costos de Hardware	Estimación
Raspberry Pi 3	$ 100.000 CLP
Notebook (2)	$ 1.000.000 CLP
Sensor de ruido (2)	$ 40.000 CLP
Celular	$ 150.000 CLP
Protoboard	$ 8.500 CLP
Adaptador Wifi	$ 1.500 CLP
Total	$ 1.300.000 CLP

Nota: Muestra el costo estimado de cada pieza de Hardware usada en el proyecto

3.1.2 Planificación de Recursos Humanos

Tabla N°5: Estimación del costo total de recursos humanos

Integrante	Rol(es)	Valor (por hora)	Hora mensual (48 horas)	Costo mensual
Diego Ferrada	Jefe de Proyecto	$9.200 CLP	10	$92.000 CLP
Diego Ferrada Javier Huanca	Programador	$5.200 CLP $5.200 CLP	20	$104.000 CLP $104.000 CLP
Diego Ferrada Javier Huanca	Documentador	$4.000 CLP $4.000 CLP	15	$60.000 CLP $60.000 CLP
Diego Ferrada Javier Huanca	Ensamblador	$2.800 CLP $2.800 CLP	20	$56.000 CLP $56.000 CLP
			Total (4 meses):	$532.000 CLP $2.128.000 CLP

Nota: Muestra el costo total de recursos humanos calculado por el valor por hora de cada rol y las horas mensuales en total.

Tabla N°6: Estimación del costo total del proyecto

Costo total	Estimación
Software	$0 CLP
Hardware	$1.300.000 CLP
Recursos humanos	$2.128.000 CLP
Total Proyecto:	$ 3.428.000 CLP

Nota: Muestra el costo total estimado del proyecto calculado por el costo total estimado del Software, el costo total estimado del Hardware y el costo total estimado de recursos humanos.

3.2 Lista de actividades (carta Gantt)

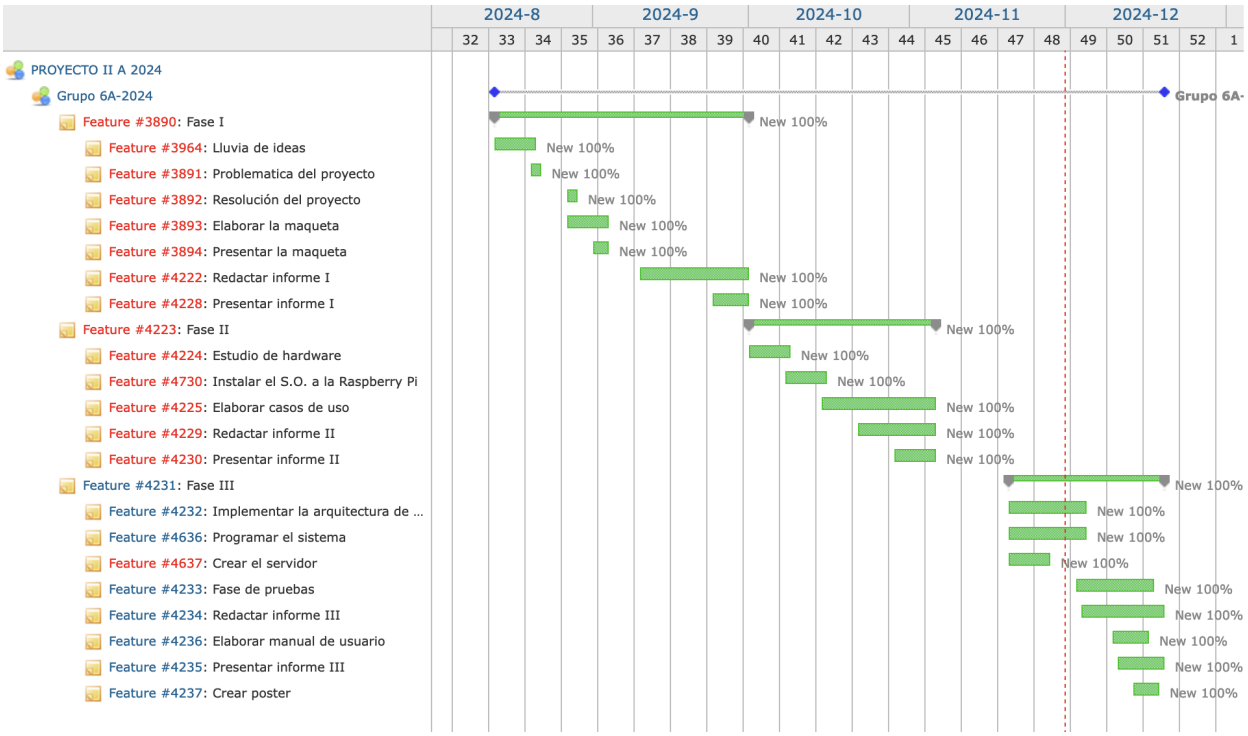
3.2.1 Actividades de trabajo

Se han concretado tareas a realizar para el desarrollo del proyecto en la fase 1:

- Lluvia de ideas.
- Concretación de problemática.
- Concretación de solución.
- Elaboración de maqueta.
- Entrega del primer informe.

3.2.2 Asignación de tiempo

Figura N°1: Carta Gantt



Nota: Muestra las actividades a realizar en el proyecto y el tiempo estimado de cada una.

3.3 Planificación de la gestión de riesgos

Se han clasificado los riesgos latentes en 4 niveles de impacto que determinan la urgencia con la que se deben remediar:

1. Catastrófico.
2. Crítico.
3. Marginal.
4. Despreciable.

Tabla N°7: Posibles riesgos en el proyecto

Riesgo	Probabilidad de ocurrencia	Nivel de impacto	Acción remedial
Insuficiencia por parte de la tecnología disponible para cubrir los requerimientos del proyecto	50%	1	Conseguir la tecnología necesaria o planificar otra forma de cubrir los requerimientos sin necesidad de esa tecnología.
Medición imprecisa de decibeles por parte de los sensores	50%	4	Revisar y actualizar los parámetros de niveles de decibeles.
Pérdida de código	40%	1	Trabajar por medio de GitHub para guardar los códigos. Si un código no ha sido guardado, repasar la información entre los miembros para volver a hacer el código
Celular no compatible con el sistema	40%	2	Buscar otro celular que sea compatible.
Disponibilidad limitada de los integrantes	40%	2	Establecer un plan para redistribuir tareas y asegurar una comunicación continua dentro del equipo, con el fin de reducir demoras.
Desconexión entre componentes de Hardware	30%	3	Revisar el estado del Raspberry Pi y modificar la conexión que tiene con los componentes.
Personal enfermo	30%	3	Trabajar de forma virtual y entregar al miembro enfermo la información de

			la clase.
Fallos de Hardware de Raspberry Pi o de sensores	25%	2	Realizar pruebas exhaustivas del Raspberry Pi y de cada sensor antes de ser implementados.
Hardware en mal estado	20%	2	Reemplazar el Hardware en mal estado y con repuestos de seguridad.
Inasistencia de personal	10%	4	Entregar a cada miembro la información necesaria para continuar con el desarrollo del proyecto.

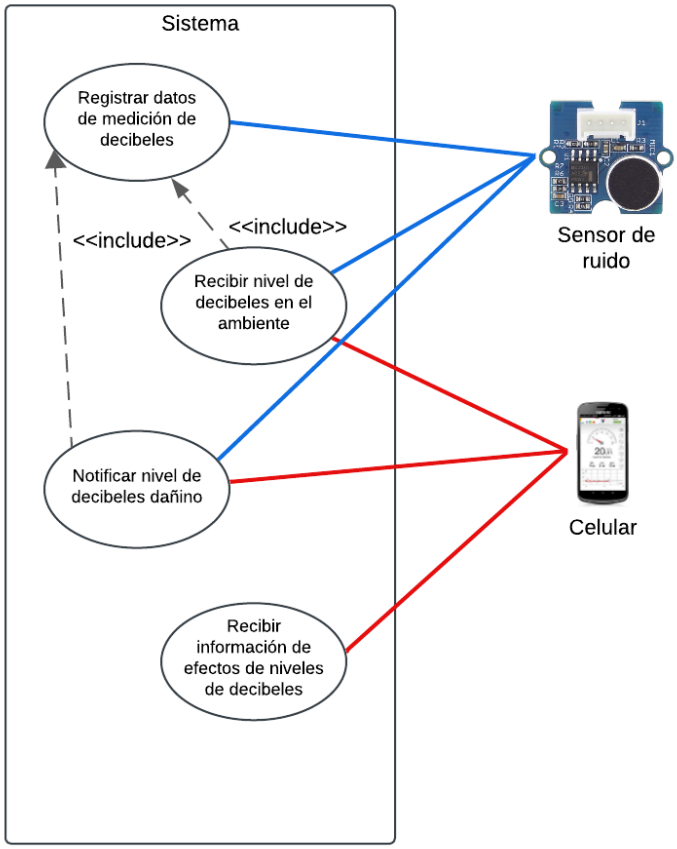
Nota: Muestra los posibles riesgos junto a su probabilidad de ocurrencia, su nivel de impacto dependiendo de cuanto afecten al desarrollo del proyecto y una acción remedial que permita mitigar o evitar sus consecuencias.

4. Planificación de los procesos técnicos

4.1 Modelos de procesos

4.1.1 Modelo de diseño (caso de uso general)

Figura N°2: Caso de uso general



Nota: Muestra el diseño de caso de uso general.

4.1.2 Modelo de diseño (casos de uso)

Tabla N°8: CUS Notificar nivel de decibeles dañino

<table>
<tr><th colspan="2">Nombre CUS: Notificar nivel de decibeles dañino</th></tr>
<tr><td colspan="2">Autor/Fecha: Diego Ferrada 29/10/2024</td></tr>
<tr><td colspan="2">Descripción: Este CUS permite indicar al usuario cuando el nivel de decibeles alcanza niveles peligrosos.</td></tr>
<tr><td colspan="2">Actor: Celular, sensor de ruido</td></tr>
<tr><td colspan="2">Precondición: La información del nivel límite letal y la información del nivel de decibeles actual.</td></tr>
<tr><td>Flujo Principal: Celular

3.- Activa un mecanismo que avisa al usuario respecto al nivel de decibeles dañino.
4.- Muestra al usuario sugerencias básicas para mitigar los efectos del nivel de decibeles.</td><td>Flujo Principal: Sistema
1.- <<include>> Registrar datos de medición de decibeles.
2.- Entrega la confirmación de que el nivel de decibeles en el ambiente es dañino.</td></tr>
<tr><td>Flujo Alternativo:</td><td></td></tr>
<tr><td colspan="2">Postcondiciones: El usuario es notificado del nivel de decibeles dañino y recibe sugerencias básicas.</td></tr>
</table>

Nota: Muestra el proceso de notificación de nivel de decibeles dañino al usuario.

Tabla N°9: CUS Recibir información de efectos de niveles de decibeles

<table>
<tr><td colspan="2">Nombre CUS: Recibir información de efectos de niveles de decibeles</td></tr>
<tr><td colspan="2">Autor/Fecha: Diego Ferrada 01/11/2024</td></tr>
<tr><td colspan="2">Descripción: Este CUS permite indicar al usuario qué efectos tienen distintos intervalos de niveles de decibeles en la salud.</td></tr>
<tr><td colspan="2">Actor: Celular</td></tr>
<tr><td colspan="2">Precondición: La información de los efectos de cada nivel deben estar en el celular.</td></tr>
<tr><td>Flujo Principal: Celular
1.- Muestra información de cada intervalo de nivel de decibeles al usuario:
● Efectos en la salud.
● Ejemplos de eventos u objetos que causan un similar nivel de decibeles.</td><td>Flujo Principal: Sistema</td></tr>
<tr><td>Flujo Alternativo:</td><td></td></tr>
<tr><td colspan="2">Postcondiciones: El usuario recibe la información de los efectos de cada intervalo de nivel de decibeles en la salud.</td></tr>
</table>

Nota: Muestra el proceso de recepción de información de efectos en la salud que tienen distintos niveles de decibeles desde el celular.

Tabla N°10: CUS Recibir nivel de decibeles en el ambiente

<table>
<tr><td colspan="2">Nombre CUS: Recibir nivel de decibeles en el ambiente</td></tr>
<tr><td colspan="2">Autor/Fecha: Javier Huanca 30/10/2024</td></tr>
<tr><td colspan="2">Descripción: Este CUS permite al usuario recibir en su dispositivo móvil la información sobre el nivel actual de decibeles en el ambiente.</td></tr>
<tr><td colspan="2">Actor: Sensor de ruido, celular</td></tr>
<tr><td colspan="2">Precondición: Los sensores de ruido deben estar conectados al sistema y el celular debe tener conexión a internet.</td></tr>
<tr><td>Flujo Principal: Celular

2.- El usuario recibe el nivel de decibeles a través del celular.</td><td>Flujo Principal: Sistema
1.- <<Include>> Registrar datos de medición de decibeles.

3.- El sistema muestra el nivel actual de decibeles en la interfaz de la aplicación.</td></tr>
<tr><td>Flujo Alternativo:</td><td></td></tr>
<tr><td colspan="2">Postcondiciones: El usuario recibe la información del nivel de decibeles en tiempo real.</td></tr>
</table>

Nota: Muestra el proceso de recepción de información del nivel de decibeles actual en el ambiente.

Tabla N°11: CUS Registrar datos de medición de decibeles

<table>
<tr><td colspan="2">Nombre CUS: Registrar datos de medición de decibeles</td></tr>
<tr><td colspan="2">Autor/Fecha: Javier Huanca 31/10/2024</td></tr>
<tr><td colspan="2">Descripción: Este CUS permite al sistema registrar y almacenar las mediciones de decibeles realizadas por los sensores de ruido.</td></tr>
<tr><td colspan="2">Actor: Sensor de ruido</td></tr>
<tr><td colspan="2">Precondición: Los sensores de ruido deben estar conectados y funcionando en el sistema.</td></tr>
<tr><td>Flujo Principal: Sensor de ruido

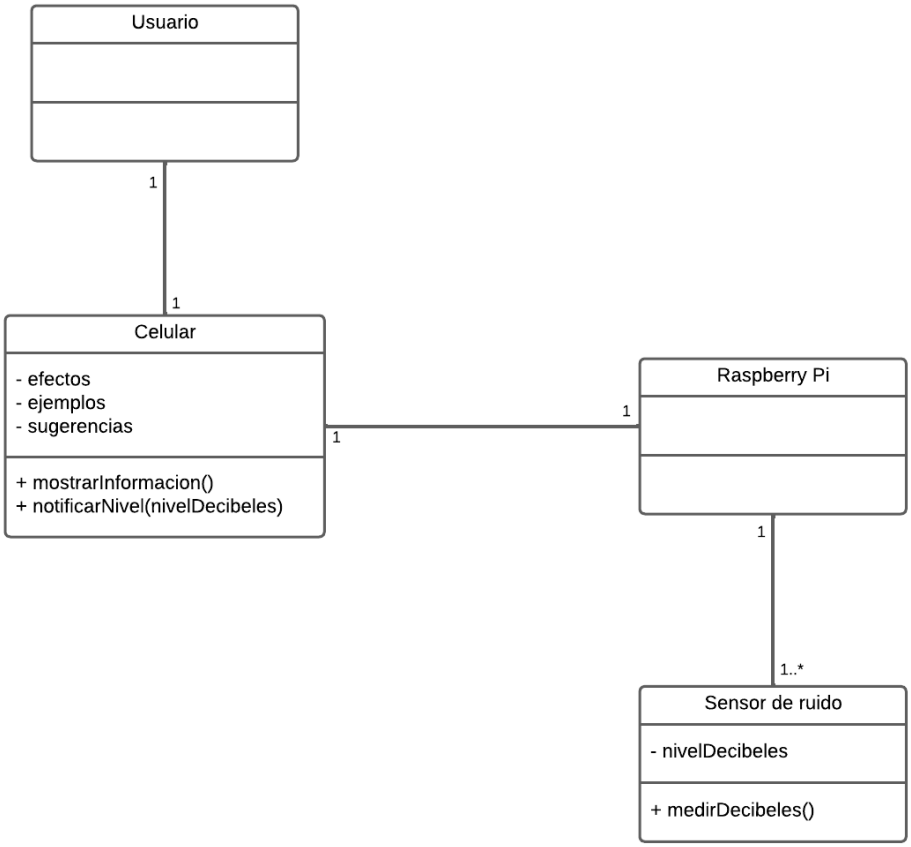
2.- El sensor realiza la medición y envía el nivel de decibeles al sistema.</td><td>Flujo Principal: Sistema
1.- El sistema solicita al sensor que realice una medición de los niveles de decibeles en el entorno.

3.- El sistema recibe y almacena los datos de decibeles.</td></tr>
<tr><td>Flujo Alternativo:</td><td></td></tr>
<tr><td colspan="2">Postcondiciones: Los datos de decibeles quedan registrados en el sistema.</td></tr>
</table>

Nota: Muestra el proceso de registro de información de nivel de decibeles actual en el ambiente.

4.1.3 Modelo de diseño (diagrama de clases)

Figura N°3: Diagrama de clases

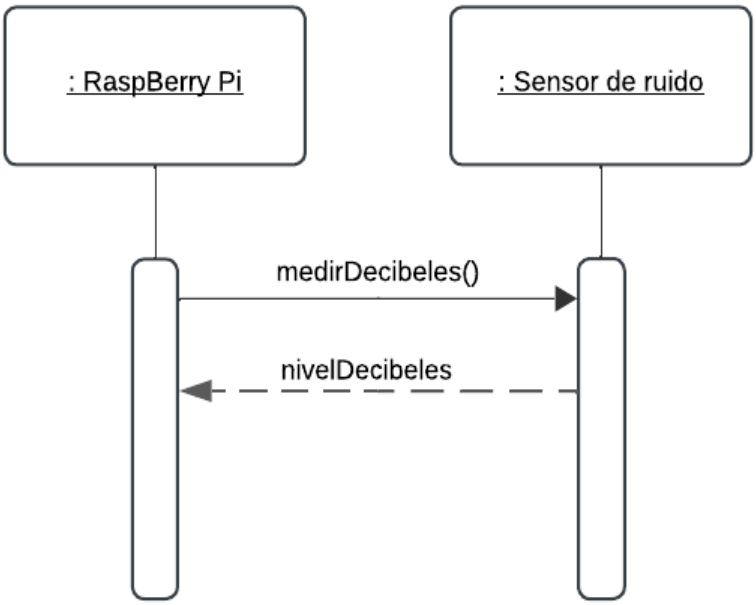


Nota: Muestra las relaciones entre las clases involucradas en el sistema.

4.1.4 Modelo de interacción (diagramas de secuencia)

Registrar datos de medición de decibeles

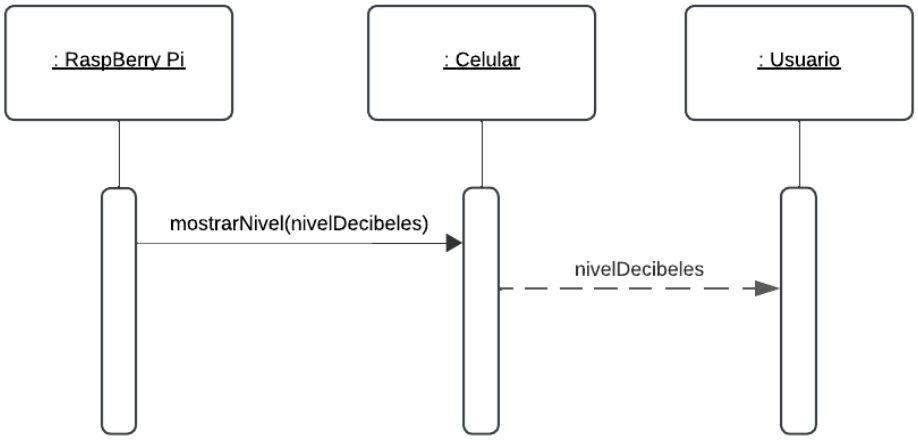
Figura N°4: Diagrama de secuencia para el caso de uso registrar datos de medición de decibeles.



Nota: Muestra el flujo de interacción entre la RaspBerry Pi y el sensor de ruido.

Recibir nivel de decibeles en el ambiente

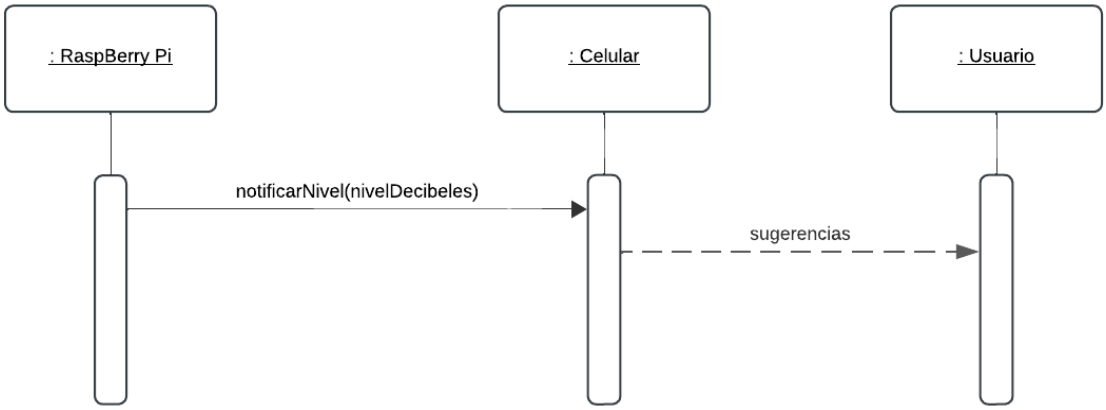
Figura N°5: Diagrama de secuencia para el caso de uso recibir nivel de decibeles en el ambiente.



Nota: Muestra el flujo de interacción entre la RaspBerry Pi, celular y el usuario.

Notificar nivel de decibeles dañino

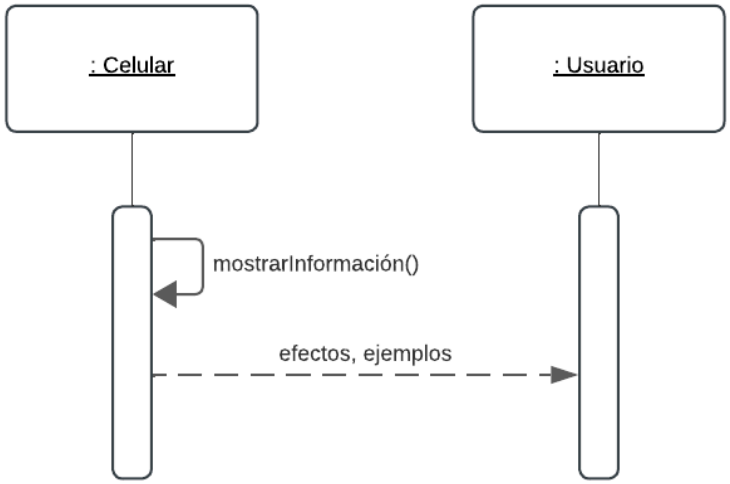
Figura N°6: Diagrama de secuencia para el caso de uso notificar nivel de decibeles dañino.



Nota: Muestra el flujo de interacción entre la RaspBerry Pi, celular y el usuario.

Recibir información de efectos de niveles de decibeles

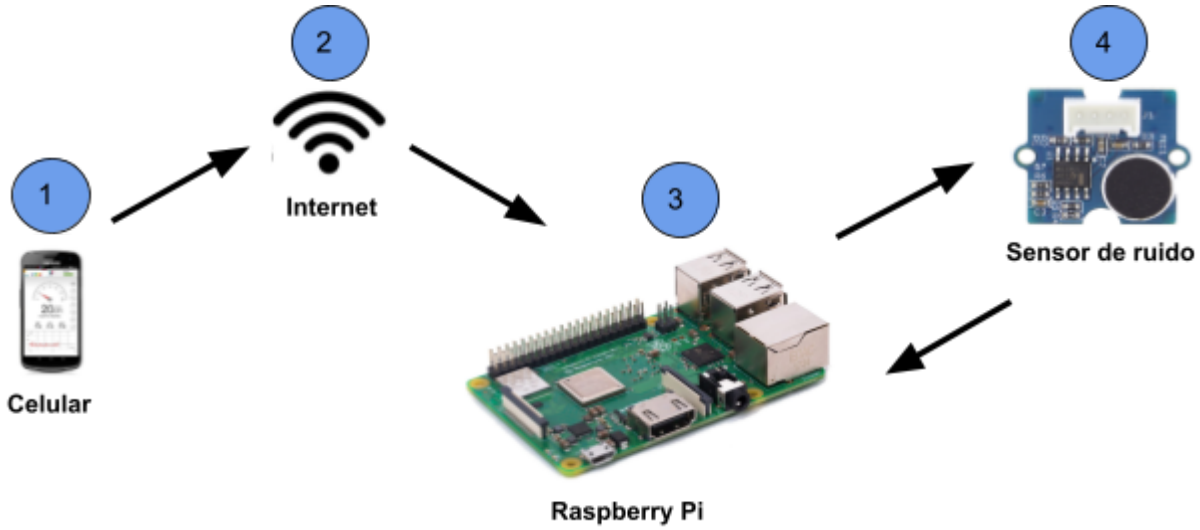
Figura N°7: Diagrama de secuencia para el caso de uso recibir información de efectos de niveles de decibeles.



Nota: Muestra el flujo de interacción entre el celular y el usuario.

4.1.5 Descripción de la arquitectura vista del modelo diseño

Figura N°8: Arquitectura vista del modelo diseño

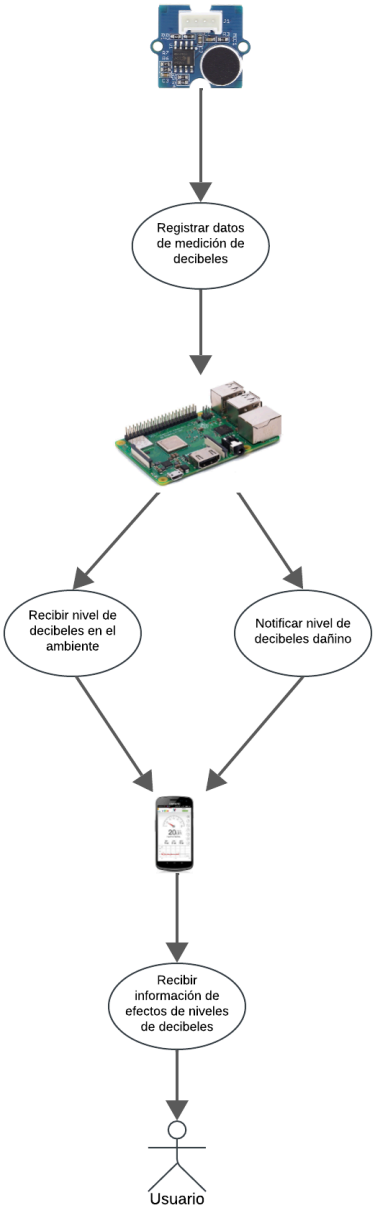


Nota: Muestra la secuencia de comunicación del sistema.

1. El usuario interactúa con la aplicación en el celular para consultar el nivel de ruido o recibe una notificación automática si se detecta un nivel de decibeles elevado.
2. El celular debe estar conectado a Internet para poder comunicarse con la Raspberry Pi.
3. La Raspberry Pi recopila los datos enviados por el sensor de ruido.
4. El sensor de ruido captura los niveles de decibeles en el entorno.

4.1.6 Descripción de la arquitectura vista de los módulos

Figura N°9: Arquitectura vista respecto de los módulos de caso de uso general



Nota: Muestra cómo la arquitectura conecta los distintos módulos para cumplir con sus funcionalidades.

4.1.7 Especificaciones de requerimientos

Requerimientos funcionales:

- El sistema debe detectar los niveles de decibeles tanto del interior como del exterior del hogar mediante sensores de ruido.
- El sistema enviará notificaciones a través de la aplicación móvil cuando el nivel de ruido supere los niveles dañinos, incluyendo sugerencias para mitigar el ruido.
- La aplicación móvil mostrará los niveles actuales de decibeles detectados en tiempo real.

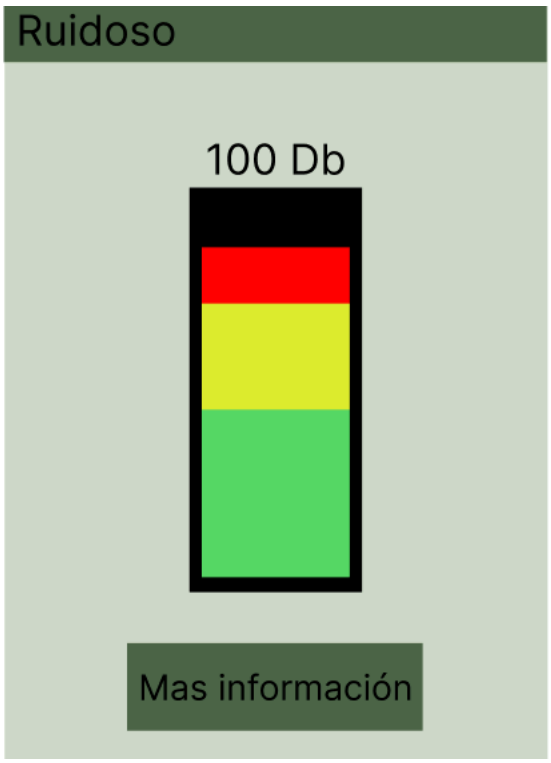
Requerimientos no funcionales:

- La aplicación debe estar disponible en todo momento para que el usuario pueda acceder a las mediciones cuando lo desee.
- La interfaz de usuario de la aplicación móvil debe ser sencilla y fácil de interpretar por usuarios con conocimientos básicos.
- El sistema debe ser capaz de agregar más sensores en el futuro sin afectar su rendimiento.
- El sistema debe ser fácil de mantener y actualizar, permitiendo cambios en los parámetros de medición y en la aplicación móvil.

4.1.8 Diseño de interfaz del usuario

Pantalla Principal

Figura N°10: Interfaz de la página principal



Nota: Muestra el diseño de la pantalla principal.

La pantalla principal muestra una barra de colores que representa visualmente el nivel de decibeles actual, ajustando su tamaño y color según el nivel de ruido, e indicando si este es bajo, moderado o peligroso, además incluye un botón para más información.

Más información

Figura N°11: Interfaz de más información

Nota: Muestra el diseño de la pantalla “Más información” .

La pantalla de "Más información" muestra los efectos de diferentes intervalos de decibeles en la salud auditiva, con ejemplos de fuentes de ruido que los causan, organizados por colores que representan la gravedad del impacto, e incluye un botón de "Volver" para regresar a la pantalla principal.

Alerta

Figura N°12: Interfaz de alerta

Nota: Muestra el diseño de la pantalla emergente.

La pantalla de alerta emergente notifica al usuario sobre un nivel de ruido peligroso y sugiere medidas preventivas para proteger su salud auditiva. Presenta una lista de acciones recomendadas que debe seguir el usuario, acompañada de un botón de confirmación etiquetado como "Entendido". Además, la lista de medidas se actualiza de manera automática para proporcionar nuevas recomendaciones si el nivel de ruido persiste elevado.

4.2 Herramientas y técnicas

Herramientas a usar:

- Python: Lenguaje de programación principal para el código del sistema.
- VS Code: Entorno de desarrollo integrado principal para el desarrollo del código.
- Raspberry Pi OS: Sistema operativo basado en Debian, diseñado específicamente para correr en la Raspberry Pi.
- GitHub: Plataforma de desarrollo colaborativo que permite la gestión de versiones de código.
- Sensor de ruido: Dispositivo de hardware encargado de medir los niveles de sonido en el entorno y enviar estos datos a la Raspberry Pi
- Celular: Usado para mostrar información al usuario.

Técnicas a usar:

- Dividir para conquistar: Se divide la programación en partes (sensor, sistema, celular) para un mejor manejo del código de cada parte.
- Programación estructurada: Se usan estructuras de control (if-else, while, for) para escribir código organizado y fácil de entender.
- Programación modular: Implica dividir un programa en partes o funciones más pequeñas, cada una con una tarea específica.

5. Implementación

5.1 Plan de integración

El plan de integración se organiza en tres fases generales para estructurar la implementación del sistema de manera eficiente:

Fase 1: Configuración inicial

- Realizar una lluvia de ideas y definir la problemática principal.
- Desarrollar los elementos base del proyecto, como la maqueta inicial y el diseño de los primeros componentes del sistema.
- Redactar y presentar el informe de la primera fase del proyecto.

Fase 2: Desarrollo del sistema

- Estudiar y configurar los componentes de hardware, como los sensores y la Raspberry Pi.
- Elaborar y validar los casos de uso para las funcionalidades principales del sistema.
- Redactar y presentar el informe de la segunda fase del proyecto.

Fase 3: Integración y finalización

- Implementar la arquitectura completa del sistema, integrando hardware y software.
- Realizar pruebas exhaustivas del sistema para garantizar su funcionamiento correcto.
- Redactar y presentar el informe de la tercera fase, manual de usuario y póster del proyecto.

5.2 Modelo de implementación

El desarrollo del sistema "Ruidoso" se realizó utilizando el lenguaje de programación Python en el entorno de desarrollo Visual Studio Code, usando las siguientes librerías:

- Tkinter: Utilizada para construir la interfaz gráfica del usuario.
- Grovepi: Utilizada para la comunicación con el sensor de ruido conectado a la Raspberry Pi.
- Threading: Utilizada para ejecutar procesos simultáneos
- Time: Utilizada para manejar las pausas entre las lecturas del sensor.

- Subprocess: Utilizada para ejecutar procesos externos desde el programa principal.

5.3 Módulos implementados

Interfaz de la página principal

Figura N°13: Función de actualizar nivel de ruido

```
def Actualizar_Nivel_Ruido():
    while True:
        try:
            # Leer el valor bruto del sensor
            raw_value = grovepi.analogRead(sound_sensor)

            # Convertir a decibeles (ajusta el rango según tu calibración)
            decibeles = (raw_value / 1023) * 100

            # Calcular el tamaño dinámico de la barra
            altura = int((decibeles / 100) * 200)  # Altura máxima: 200 px
            grosor = 40   # Grosor máximo: 50 px

            # Ajustar la barra en el canvas
            canvas.delete("barra")  # Eliminar la barra anterior
            canvas.create_rectangle(
                20 - grosor // 2,  # Izquierda
                200 - altura,       # Arriba
                20 + grosor // 2,  # Derecha
                200,                # Abajo
                fill=Color_Segun_Db(decibeles),
                tags="barra"
            )

            # Actualizar el texto de decibeles
            etiqueta_db.config(text=f"{int(decibeles)} Db")

            # Actualizar el estado (texto de arriba)
            if decibeles < 40:
                etiqueta_estado.config(text="Silencioso", bg="green")
            elif 40 <= decibeles < 70:
                etiqueta_estado.config(text="Moderado", bg="yellow")
            else:
                etiqueta_estado.config(text="Ruidoso", bg="red")

                # Abrir ventana de alerta si supera los 80 dB y no está activa
                if decibeles > 80 and not alerta_activa:
                    abrir_ventana_alerta()

            time.sleep(0.2)  # Pausa para evitar saturación
        except IOError:
            print("Error al leer el sensor.")
        except KeyboardInterrupt:
            break
```

Esta función lee los valores del sensor de sonido, calcula el nivel de decibeles y actualiza dinámicamente la barra en el canvas, junto con el estado del nivel de ruido (silencioso, moderado, ruidoso).

Figura N°14: Función de abrir ventana de alerta

```
# Función para abrir la ventana de alerta
def abrir_ventana_alerta():
    global alerta_activa, recomendaciones_set
    alerta_activa = True

    # Pasar el conjunto de recomendaciones como argumento al archivo de alerta
    proceso_alerta = subprocess.Popen(["python3", "interfaz_emergente.py", str(recomendaciones_set)])
    recomendaciones_set = (recomendaciones_set + 1) % 2  # Alternar entre 0 y 1

    # Esperar a que el proceso termine y reiniciar alerta_activa
    threading.Thread(target=esperar_cierre_alerta, args=(proceso_alerta,)).start()
```

Esta función abre una ventana emergente de alerta con un conjunto de recomendaciones, alternando entre diferentes mensajes, y evita que se activen múltiples alertas al mismo tiempo.

Figura N°15: Función de esperar cierre de alerta

```
def esperar_cierre_alerta(proceso):
    global alerta_activa
    proceso.wait()  # Esperar a que el proceso de la ventana emergente termine
    alerta_activa = False  # Reiniciar alerta_activa
```

Esta función espera a que el proceso de la ventana emergente de alerta termine y reinicia el estado global para permitir nuevas alertas.

Figura N°16: Función del color según el nivel de decibeles

```
# Función para determinar el color según el nivel de decibeles
def Color_Segun_Db(decibeles):
    if decibeles < 40:
        return "green"
    elif 40 <= decibeles < 70:
        return "yellow"
    else:
        return "red"
```

Esta función determina el color de la barra según el nivel de decibeles.

Figura N°17: Función de más información

```
# Función para ir a la interfaz de más información
def Mas_Informacion():
    ventana.destroy()
    import interfaz_informacion
```

Esta función cierra la ventana principal y redirige a una nueva interfaz.

Interfaz de más información

Figura N°18: Función de cada nivel de ruido

```
# Sección para cada nivel de ruido
def create_noise_section(container, db_range, bg_color, text_color, description):
    frame = tk.Frame(container, bg=bg_color, pady=10)
    frame.pack(fill=tk.X, padx=10, pady=5)
    db_label = tk.Label(frame, text=f"{db_range}", bg=bg_color, fg=text_color, font=("Arial", 10, "bold"))
    db_label.pack(anchor="w", padx=10)
    desc_label = tk.Label(frame, text=description, bg=bg_color, fg=text_color, justify="left", wraplength=230, font=("Arial", 10))
    desc_label.pack(anchor="w", padx=20)
```

Función que genera dinámicamente secciones informativas sobre los niveles de ruido, indicando su rango en decibeles, los efectos en la salud, y ejemplos, utilizando colores para representar visualmente la gravedad.

Figura N°19: Función de volver

```
# Función para ir a la interfaz principal
def Volver():
    ventana.destroy()
    import interfaz_principal
```

Función que cierra la ventana actual y redirige al usuario a la interfaz principal del sistema.

Interfaz de alerta

Figura N°20: Función de cerrar la alerta

Función de cerrar la alerta

```
# Botón para cerrar la alerta
def cerrar_alerta():
    ventana.destroy()
```

Función que cierra la ventana emergente de alerta cuando el usuario presiona el botón "Entendido".

5.4 Reporte de revisión

Revisión de Código:

- Se verificó la correcta implementación de las funcionalidades principales del sistema, incluyendo la lectura del sensor, la representación visual de los niveles de ruido y la actualización automática de la interfaz gráfica.
- Los módulos de código fueron probados individualmente para garantizar su correcto funcionamiento antes de integrarlos al sistema completo.

Revisión de Hardware:

- Se realizaron pruebas de conexión entre la Raspberry Pi y los sensores de ruido para confirmar la estabilidad y precisión de los datos recopilados.

Revisión de la Interfaz Gráfica:

- Se evaluó la funcionalidad y estética de las tres ventanas principales (pantalla principal, ventana de información y ventana de alerta).
- La barra dinámica que representa los niveles de ruido fue revisada para garantizar una visualización proporcional y clara.

6. Problemas encontrados

Problema 1:

Durante la instalación de RFR_Tools en la Raspberry Pi, se presentó un error debido a la falta del archivo *tmp_rfrtools.sh*, lo que impidió completar la configuración del entorno.

Problema 2:

Durante las pruebas del script sound_sensor_test.py, se presentó el error *ModuleNotFoundError: No module named 'di_i2c'*, generado por la falta del módulo requerido por grovepi.

Problema 3:

Durante la instalación del paquete *wiringpi:armhf*, se presentó un error indicando que el paquete no está disponible, posiblemente debido a que está obsoleto o eliminado de las fuentes oficiales.

7. Soluciones propuestas

Solución al problema 1:

Fue resuelto verificando la conectividad, actualizando los paquetes del sistema y descargando manualmente el script desde el repositorio oficial.

Solución al problema 2:

Para solucionarlo, se instaló el paquete faltante utilizando pip install di_i2c, asegurando la compatibilidad y restableciendo el funcionamiento del programa.

Solución al problema 3:

La solución fue clonar un repositorio alternativo que contenga el paquete.

8. Trabajo a futuro

Para el desarrollo futuro del proyecto, se han identificado las siguientes áreas de mejora y expansión:

- Incorporar un sistema que almacene las mediciones de decibeles en una base de datos, permitiendo a los usuarios analizar tendencias de ruido a lo largo del tiempo.
- Desarrollar un mecanismo automatizado que permita mitigar la entrada de ruido externo cuando se detecten niveles peligrosos.
- Implementar más sensores de sonido en diferentes ubicaciones del hogar para cubrir zonas específicas y mejorar la precisión de las mediciones.
- Adaptar el sistema para admitir múltiples tipos de sensores, como micrófonos de alta precisión, para mejorar la exactitud de las mediciones.

9. Conclusiones

Este proyecto busca concientizar a la población sobre la salud auditiva y sobre la importancia de evitar ruidos fuertes en zonas residenciales, a través del uso de sensores de ruido ubicados en el hogar que permiten monitorear los niveles de decibeles y notificar a los usuarios cuando estos superan los límites recomendados, contribuyendo así a la protección de la salud auditiva y el bienestar en el hogar.

Su capacidad de integrarse con dispositivos móviles y su facilidad de uso lo convierten en una herramienta accesible y efectiva para las personas que buscan reducir la exposición al ruido.

El proyecto nos permite recibir más experiencia y habilidad en el desarrollo de sistemas de monitoreo y control con sensores, los cuales son cruciales a día de hoy para automatizar procesos y detectar fenómenos en nuestro entorno.

En esta segunda fase, se ha avanzado en la descripción de los casos de uso que incluye el proyecto, en la descripción de su arquitectura en dos vistas distintas y en el diseño de cada interfaz de usuario que lo conforman. Además, se han especificado los requerimientos y se han definido las herramientas y técnicas a utilizar.

Durante esta tercera fase, se completó la implementación de las principales funcionalidades del sistema, incluyendo la integración del hardware con el software, el diseño de interfaces dinámicas y una ventana emergente de alertas, logrando un sistema funcional que se adapta a diferentes niveles de ruido. Además, se aseguraron la alternancia de recomendaciones y la gestión eficiente de los procesos en paralelo mediante hilos.

La realización de este sistema ha permitido adquirir experiencia significativa en el desarrollo de sistemas de monitoreo con sensores y la resolución de problemas técnicos complejos. Se espera que este aprendizaje contribuya al fortalecimiento de las habilidades de desarrollo, la gestión de tareas y el trabajo en equipo.

10. Referencias

- Sitio web de Raspberry Pi - https://raspberrypi.cl/
- Sueldo de jefe de proyecto - https://cl.talent.com/salary?job=jefe+de+proyecto
- Sueldo de programador - https://cl.talent.com/salary?job=programador
- Sueldo de documentador - https://cl.talent.com/salary?job=Documentador
- Sueldo de ensamblador - https://cl.talent.com/salary?job=ensamblador
- Sensores de ruidos - https://wiki.seeedstudio.com/Sensor_sound/
- Raspberry Pi OS - https://www.raspberrypi.com/software/
- Lucidchart https://lucid.app/lucidchart/a05d1377-0bf4-4172-a43a-a93f79cafb8d/edit?beaconFlowId=E835BB2EC807E4DC&invitationId=inv_2d0bcf14-f2b7-4c24-bca8-fa6ed8ba2673&page=0_0#
- Drive de proyecto 2 https://drive.google.com/drive/folders/1CBpNjV8yILS1EmMoBDOFYLWmxtlpdaGL
- Tecnicas de programación https://jeffrychaves.com/diccionario/tecnica-de-programacion/
- Figma para el diseño de la interfaz https://www.figma.com/design/Y8wHMpHrmgGQCJH3Ad0d6R/Untitled?node-id=0-1&node-type=canvas&t=BKb1BUsIfwzX0r5f-0
- Regulación de ruido ambiental https://www.bcn.cl/obtienearchivo?id=repositorio/10221/33047/1/BCN_Regulacion_ruidos_Chile_Union_Europea_Francia_2022_FINAL.pdf
- RVNC Viewer - https://www.realvnc.com/es/connect/download/viewer/