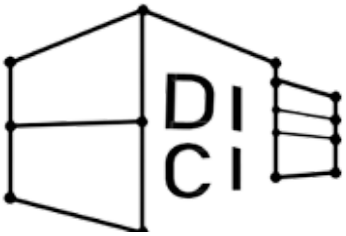
# Universidad de Tarapacá

# Facultad de Ingeniería

Departamento de Ingeniería en Computación e Informática




## Plan de Proyecto


Autor(es): Iván Collao
Sebastian Eyraud
Guillermo Pino
Giorgio Rojas

Asignatura: Proyecto 2

Profesor: Diego Aracena

Fecha: 25 / 11 / 2025

## Historial de cambios

| Fecha | Versión | Descripción | Autor(es) |
|---|---|---|---|
| 07/10/2025 | 1.0 | Inicio del informe | Guillermo Pino |
| 14/10/2025 | 1.1 | Actualización del Informe | Iván Collao<br>Sebastian Eyraud<br>Guillermo Pino<br>Giorgio Rojas |
| 21/10/2025 | 1.2 | Finalización de la parte 1 del informe | Iván Collao<br>Sebastian Eyraud<br>Guillermo Pino<br>Giorgio Rojas |
| 28/10/2025 | 2.0 | Finalización de la parte 2 del informe | Iván Collao<br>Sebastian Eyraud<br>Guillermo Pino<br>Giorgio Rojas |
| 04/11/2025 | 2.1 | Inicio de la segunda fase de entrega del informe | Guillermo Pino<br>Giorgio Rojas |
| 18/11/2025 | 2.2 | Desarrollo de los casos de uso y diagramas de secuencia | Giorgio Rojas<br>Guillermo Pino |
| 25/11/2025 | 2.3 | Desarrollo de la arquitectura, herramientas y técnicas | Iván Collao<br>Sebastian Eyraud |

# Índice

# Panorama general

## Introducción

Este proyecto surge por la necesidad de combatir la contaminación acústica que últimamente se ha convertido en un problema creciente que afecta la calidad de vida de las personas especialmente en espacios públicos o poblaciones en donde el ruido no deseado interfiere con el bienestar colectivo. El objetivo es detectar y alertar sobre emisiones sonoras perjudiciales dentro de un cierto rango dentro de la población urbana. El sistema diseñado incorpora sensores de sonido, como micrófonos capaces de identificar niveles de decibelios superiores a los umbrales establecidos como aceptables. Una vez detectado el evento acústico, se activa un mecanismo que activará una “alarma” que alertará sobre algún dispositivo que haya sobrepasado el rango establecido.

## Propósito

El propósito del dispositivo es alertar al usuario en situaciones donde se excedan los límites de ruido tolerables. De este modo, se busca fomentar el respeto por el entorno sonoro compartido, promoviendo una convivencia más armoniosa en zonas habitadas en la ciudad.

## Alcance

El Proyecto utilizará sensores de sonido además de la implementación de una Raspberry Pi 4.

## Objetivo

### Objetivo general

Desarrollar e implementar un dispositivo automatizado que detecta y alerta sobre ruidos molestos como música en excesivo volumen en áreas urbanas de alto índice de población.

### Objetivos específicos

- Implementar una programación eficiente en Raspberry Pi 4.

- Integrar correctamente los sensores de sonido.
- Diseñar y construir una maqueta funcional del dispositivo.
- Desarrollar un sistema de respuesta automatizada, que active la “alarma” sólo cuando se detecten niveles altos de decibeles (como música).
- Documentar el proceso de instalación, configuración y pruebas.
- Evaluar la efectividad del dispositivo en la reducción de la contaminación acústica.

## Suposiciones y restricciones

- Tiempo: El proyecto se debe terminar antes de la fecha límite, la cual corresponde al 24 de diciembre de 2025

- Hardware: El proyecto debe incluir a la placa Raspberry Pi 4 como su pieza de hardware principal para controlar actuadores y sensores

- Modelo Virtual: El proyecto debe ser representado y mostrado a través de un dispositivo de realidad virtual (Meta Quest 3)

- Presupuesto: Los materiales para el proyecto incluyendo una tarjeta sd no deben sobrepasar el presupuesto de 20.000$ CLP

## Entregables del proyecto

Producto funcional con Raspberry Pi 4.

Manual de usuario.

Manual de instalación.

Bitácoras de cada semana (12 en total).

Informes de avance.

Presentación de proyecto.

Informe final.

# Organización del proyecto

## Personal y entidades internas

| Rol | Encargado | Involucrado(s) |
|---|---|---|
| Jefe de Proyecto | Giorgio Rojas | Giorgio Rojas |
| Programador | Sebastian Eyraud | Sebastian Eyraud<br>Giorgio Rojas<br>Ivan Collao<br>Guillermo Pino |
| Documentador | Guillermo Pino | Guillermo Pino<br>Sebastian Eyraud |
| Analista Programador | Giorgio Rojas | Giorgio Rojas |
| Diseñador | Iván Collao | Iván Collao |

## Mecanismos de Comunicación

- Grupo de WhatsApp
- Grupo de Discord

# Planificación de los Procesos de Gestión

## Planificación de estimaciones

### Hardware

| Producto | Costo | Cantidad | Costo Total |
|---|---|---|---|
| Raspberry Pi 4 | $ 91.990 | 1 | $ 91.990 |
| Módulo Sensor de Sonido y Voz FC-04 | $ 2.890 | 1 | $ 2.890 |
| MicroSD | $ 12.000 | 1 | $ 12.000 |

| Producto | Costo | Cantidad | Costo Total |
|---|---|---|---|
| Monitor (LG) | $ 119.990 | 1 | $ 119.990 |
| Teclado | $ 8.990 | 1 | $ 8.990 |
| Mouse | $ 5.990 | 1 | $ 5.990 |
| PC1 (Acer) | $ 649.990 | 1 | $ 649.990 |
| PC2 (HP) | $ 700.000 | 1 | $ 700.000 |
| PC3 (Lenovo) | $ 549.990 | 1 | $ 549.990 |
| PC4 (Asus) | $ 499.990 | 1 | $ 499.990 |
| Total: | $ 2.641.820 | - | $ 2.641.820 |

**Software**

| Producto | Costo |
|---|---|
| Raspberry Pi OS | $ 0 |
| Google Docs | $ 0 |
| Visual Studio Code | $ 0 |
| Canva (free) | $ 0 |
| Python | $ 0 |
| Redmine | $ 0 |
| Arch Linux | $ 0 |
| Total: | $ 0 |

## Planificación de Recursos Humanos

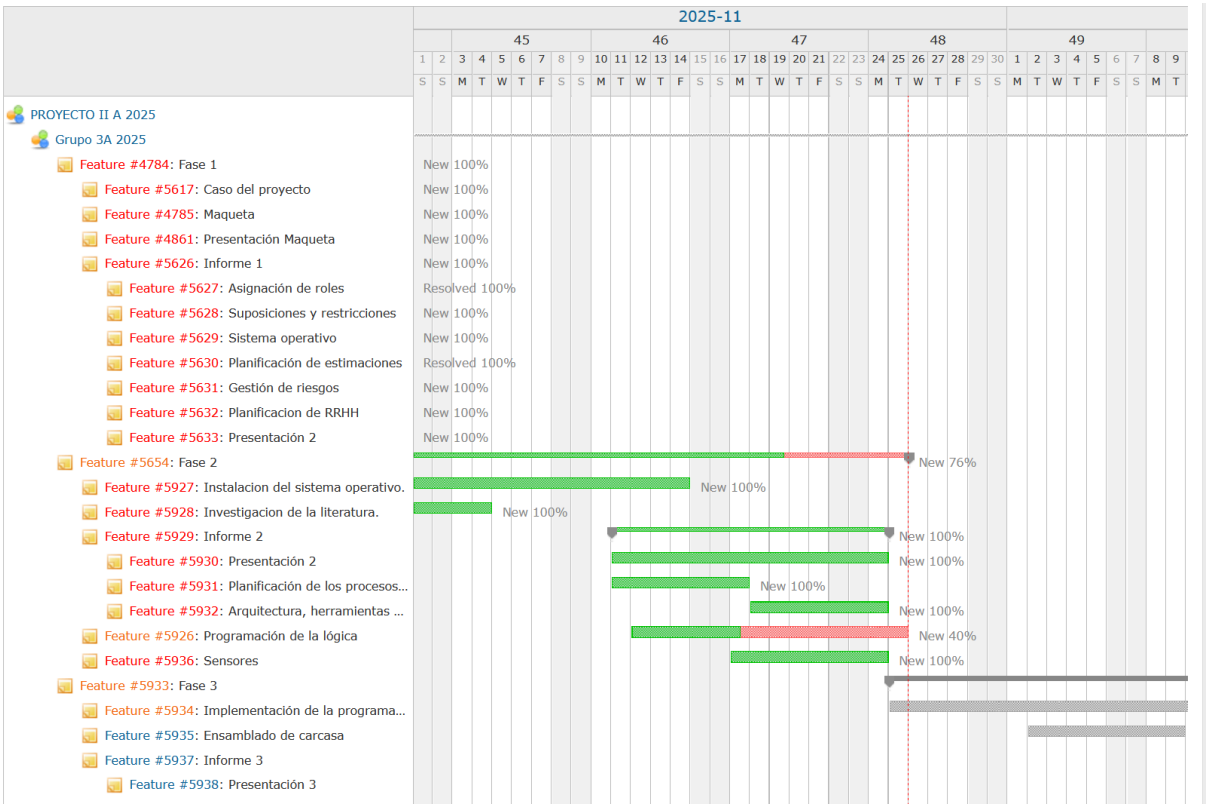
| Integrantes | Rol | Hora Total | Sueldo/Hora | Sueldo Total |
|---|---|---|---|---|
| Giorgio Rojas | Jefe de Proyecto<br>Analista<br>Programador | 84 | $ 13.500 | $ 1.134.000 |
| Iván Collao | Diseñador | 84 | $ 11.200 | $ 940.800 |
| Sebastian Eyraud | Programador | 84 | $ 9.500 | $ 798.000 |
| Guillermo Pino | Documentador | 84 | $ 7.800 | $ 655.200 |
| Total: | | | | $ 3.528.000 |

## Planificación de Costo Total

| Tipo de Costo | Costo |
|---|---|
| Costo de Hardware | $ 2.506.850 |
| Costo de Software | $ 0 |
| Costo de Recursos Humanos | $ 3.528.000 |
| Total: | $ 6.034.850 |

## Lista de actividades

### Carta Gantt (noviembre 2025)




### Actividades de trabajo

| Actividad | Descripción | Responsable |
|---|---|---|
| Bitacoras | Registro del avance de las actividades semanalmente. | Guillermo Pino |
| Organización | Planificación del proyecto, definición de roles y responsabilidades | Giorgio Rojas |
| Establecer Problemática | Identificación y definición clara del problema a resolver, análisis del contexto, establecimiento de objetivos | Sebastian Eyraud |

| Actividad | Descripción | Responsable |
|---|---|---|
| Analizar distintas soluciones | Investigación y evaluación de alternativas disponibles | Sebastián Eyraud<br>Ivan Collao<br>Giorgio Rojas<br>Guillermo Pino |
| Creación del Modelo 3D | Diseño digital de la estructura y carcasa del prototipo utilizando software (Godot) | Sebastián Eyraud<br>Ivan Collao<br>Giorgio Rojas<br>Guillermo Pino |
| Presentación | Exposición formal del proyecto ante evaluadores, demostración del funcionamiento del prototipo | Sebastián Eyraud<br>Ivan Collao<br>Giorgio Rojas<br>Guillermo Pino |
| Informe 1 | Crear el informe para la fase 1 del proyecto | Sebastián Eyraud<br>Ivan Collao<br>Giorgio Rojas<br>Guillermo Pino |

## Planificación de la gestión de riesgos

| Tipo de Riesgos | Descripción |
|---|---|
| Humanos | Bajo rendimiento, conflictos o alta rotación del equipo debido a baja moral, problemas de salud o malas relaciones. Dificultad para atraer o retener el talento necesario |
| Tecnológicos | Fallas, incompatibilidades o deficiencias en el hardware y software que provocan retrasos en el cronograma y las entregas. |
| Organización | Falta de dirección estratégica clara y comunicación deficiente por parte de la gerencia, lo que fomenta la incertidumbre y la |

| Tipo de Riesgos | Descripción |
|---|---|
| | desinformación en la organización. |
| Equipamiento | Equipo de trabajo inadecuado, obsoleto o insuficiente que merman la productividad y generan resistencia por parte del equipo. |
| Estimación | Subestimación del esfuerzo requerido para el desarrollo y la corrección de defectos, lo que resulta en el incumplimiento sistemático de los plazos acordados. |
| Requerimientos | Alta volatilidad o definición ambigua de los requerimientos, generando reprocesos constantes e insatisfacción del cliente. |

| Nivel de Impacto | Descripción |
|---|---|
| Inaceptable | Impacto determinante que puede amenazar la continuidad o el logro del proyecto. |
| Grave | Impacto significativo que necesita recursos extra para ser administrado, pero el proyecto tiene la posibilidad de seguir. |
| Leve | Efecto ligero que tiene la posibilidad de demorar algunos elementos del proyecto, pero no afecta de manera significativa los resultados. |
| Sin Importancia | Un impacto mínimo que no necesita una reacción rápida y que no tendrá un efecto significativo en el avance del proyecto. |

| Probabilidad de Ocurrencia | Probabilidad de Ocurrencia |
|---|---|
| Alta | 71% - 100% |
| Media | 31% - 70% |
| Baja | 0% - 30% |

| Riesgo | Tipo | Probabilidad de Ocurrencia | Nivel de Impacto | Acción Remedial |
|---|---|---|---|---|
| Integrante del equipo enfermo | Humano | 25% | Leve | Redistribuir tareas entre miembros disponibles |
| Falla en microSD | Tecnológico | 30% | Grave | Implementar respaldos en múltiples tarjetas y verificar integridad periódicamente |
| Falta de respaldos | Tecnológico | 45% | Inaceptable | Realizar verificación semanal para verificar que se tienen múltiples respaldos |
| Componente defectuoso | Tecnológico | 35% | Leve | Mantener stock de componentes de repuesto, probar componentes al recibirlos e identificar proveedores alternativos |
| Falta de comunicación | Organización | 40% | Grave | Realizar reuniones diarias de sincronización; |
| Retrasos con | Estimación | 35% | Leve | Revisiones semanales de |

| Riesgo | Tipo | Probabilidad de Ocurrencia | Nivel de Impacto | Acción Remedial |
|---|---|---|---|---|
| los avances | | | | progreso y hacer ajustes al plan |
| Falla en los sensores | Tecnológico | 30% | Leve | Adquirir sensores de respaldo |
| Recursos de Hardware Incompatibles | Equipamiento | 40% | Grave | Verificar compatibilidad antes de compra, mantener lista de hardware certificado |
| Errores en el código | Tecnológico | 50% | Leve | Implementar testing y revisiones de código |
| Dificultad para la obtención de sensores específicos | Tecnológico | 20% | Leve | Identificar proveedores alternativos y considerar sensores equivalentes |
| Cambio en los requerimientos | Requerimiento | 25% | Baja | Implementar control de cambios y evaluar impacto antes de aceptar. |
| Enchufes defectuosos | Tecnológico | 55% | Media | Usar fuentes de alimentación distintas |

# Planificación de los Procesos Técnicos

## Modelo de procesos

### Requerimientos

A continuación se definirán los requisitos funcionales y no funcionales del proyecto Hush que son cruciales para el buen funcionamiento del sistema desarrollado tanto de funcionalidad como de calidad esperada.

### Requerimientos funcionales

- Reconocer sonidos en niveles altos de decibeles.
- Emitir una notificación al usuario en tiempo real (alertas).
- Permite al usuario una configuración de parámetros (establecer rangos de decibeles para la alerta de sonidos).

### Requerimientos no funcionales

- Interfaz intuitiva de fácil uso para el usuario.
- Fácil mantenibilidad para actualizar y corregir el sistema.
- Indicador del estado mediante un indicador visual (LED) que muestra el estado operativo.
- El dispositivo es capaz de funcionar con una batería portable.

## Modelo de diseño (caso de uso general)

Se ha modelado el siguiente diagrama para representar el funcionamiento del sistema:


El modelo contará con los siguientes actores para su correcto funcionamiento

- Usuario
- Sensor de sonido
- Sensor Led
- Interfaz

## Casos de uso y diagrama de secuencia

## Detección de ruido

| |
|---|
| **Nombre de caso de Uso**: Detección de ruido |
| **Resumen**:<br>El sistema recibirá información cada vez que el sensor de sonido registre un ruido que haya superado el límite predefinido. |
| **Actor(es)**: Sistema y sensor de sonido. |
| **Precondición**: |

| |
|---|
| El sistema debe estar encendido y a la espera de que se produzca algún ruido. |
| **Descripción**:<br>1. El sensor de sonido detecta un valor de ruido ambiente.<br>2. El sistema compara el valor obtenido con el umbral definido por el usuario.<br>3. Si el nivel de ruido supera el umbral, el sensor entrega el valor al sistema.<br>4. El sistema recibe y valida la señal, confirmando que el límite de ruido fue excedido.<br>5. El sistema activa el LED indicador para señalar localmente la detección. |
| **Alternativa**:<br>3.1. Si el nivel de ruido detectado no supera el umbral, el sistema continúa en estado de espera sin activar el LED ni enviar notificaciones. |
| **Postcondición**:<br>El sistema queda listo para continuar monitoreando el nivel de ruido y activar nuevas alertas cuando corresponda. |

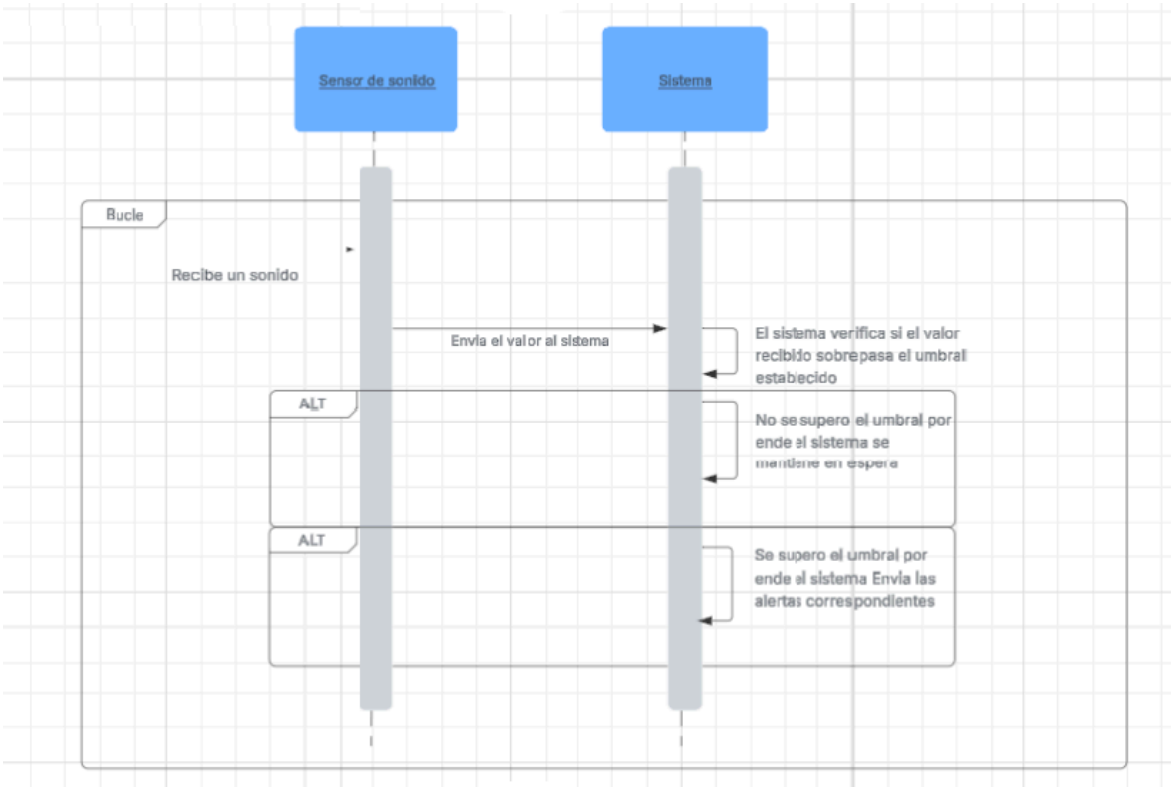



**Diagrama 1.** Diagrama de secuencia de la detección de ruido.

### Notificar al usuario

| |
|---|
| **Nombre de caso de Uso**: Notificar al usuario |
| **Resumen:**<br>El sistema, utilizando la Raspberry Pi 4B y un micrófono, detecta un nivel de sonido que excede un umbral establecido y envía una notificación de alerta al usuario por su canal preferido (e.g., email, mensaje push, Telegram). |
| **Actor(es)**: Sistema, usuario, sensor LED, Teléfono |
| **Precondición**:<br>El sistema debe estar encendido y ejecutando el software, el micrófono debe estar conectado y funcionando correctamente. El umbral de ruido (dB) ha sido configurado. La configuración de notificación del usuario (canal y destino) es válida y accesible |
| **Descripción**:<br>1. Al detectar un nivel de dB que supera el umbral definido y recibido el valor entregado por el sensor se inicia el evento de notificar al usuario.<br>2. El sistema genera un mensaje de alerta y envía una notificación al usuario y muestra un mensaje en el sensor led. |
| **Alternativa**:<br>2.1. Falla al enviar la notificación. Si el servicio de notificación no responde o falla el envío. |
| **Postcondición**: El usuario ha recibido una notificación de la alerta de ruido |

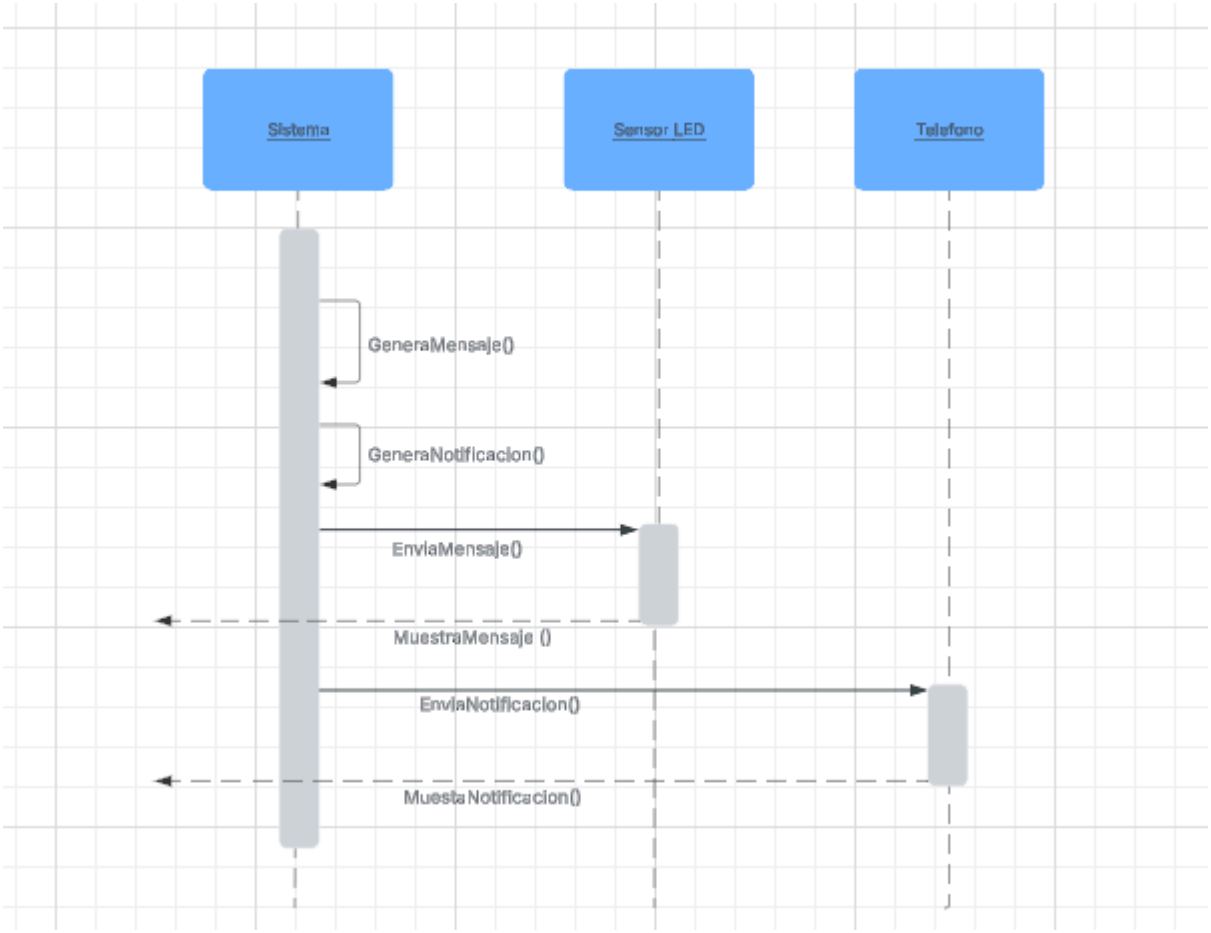



**Diagrama 2.** Diagrama de secuencia de la notificación al usuario.

### Encender y apagar el sistema

| |
|---|
| **Nombre de caso de Uso**: Encender y apagar el sistema |
| **Resumen:** El usuario puede encender o apagar el sistema mediante un botón físico o interruptor conectado a la Raspberry Pi. El encendido inicializa los componentes necesarios para el monitoreo del sensor de sonido, mientras que el apagado detiene todos los procesos y coloca al sistema en un estado seguro |
| **Actor(es)**: Sistema, usuario. |
| **Precondición**:<br>El sistema debe contar con una batería o pila instalada y con suficiente carga.<br>El usuario debe interactuar con los botones de la interfaz gráfica. |
| **Descripción**:<br>1. El usuario presiona el botón físico de encendido/apagado del dispositivo.<br>2. El sistema detecta la señal manual enviada por el usuario.<br>3. Si la señal corresponde a encender, el sistema inicializa los módulos principales, incluyendo el sensor de sonido, la interfaz de notificaciones y el LED indicador.<br>4. Si la señal corresponde a apagar, el sistema detiene de forma segura todas las funciones activas, apaga el LED e ingresa en modo de bajo consumo para preservar la batería.<br>5. El sistema verifica que cada componente haya terminado su proceso de inicialización o apagado correctamente. |
| **Alternativa**:<br>**2.1** Si el usuario presiona el botón cuando la batería se encuentra con carga insuficiente, el sistema no inicia e informa la condición mediante un parpadeo breve del LED. |
| **Postcondición**: El usuario ha recibido una notificación de la alerta de ruido |

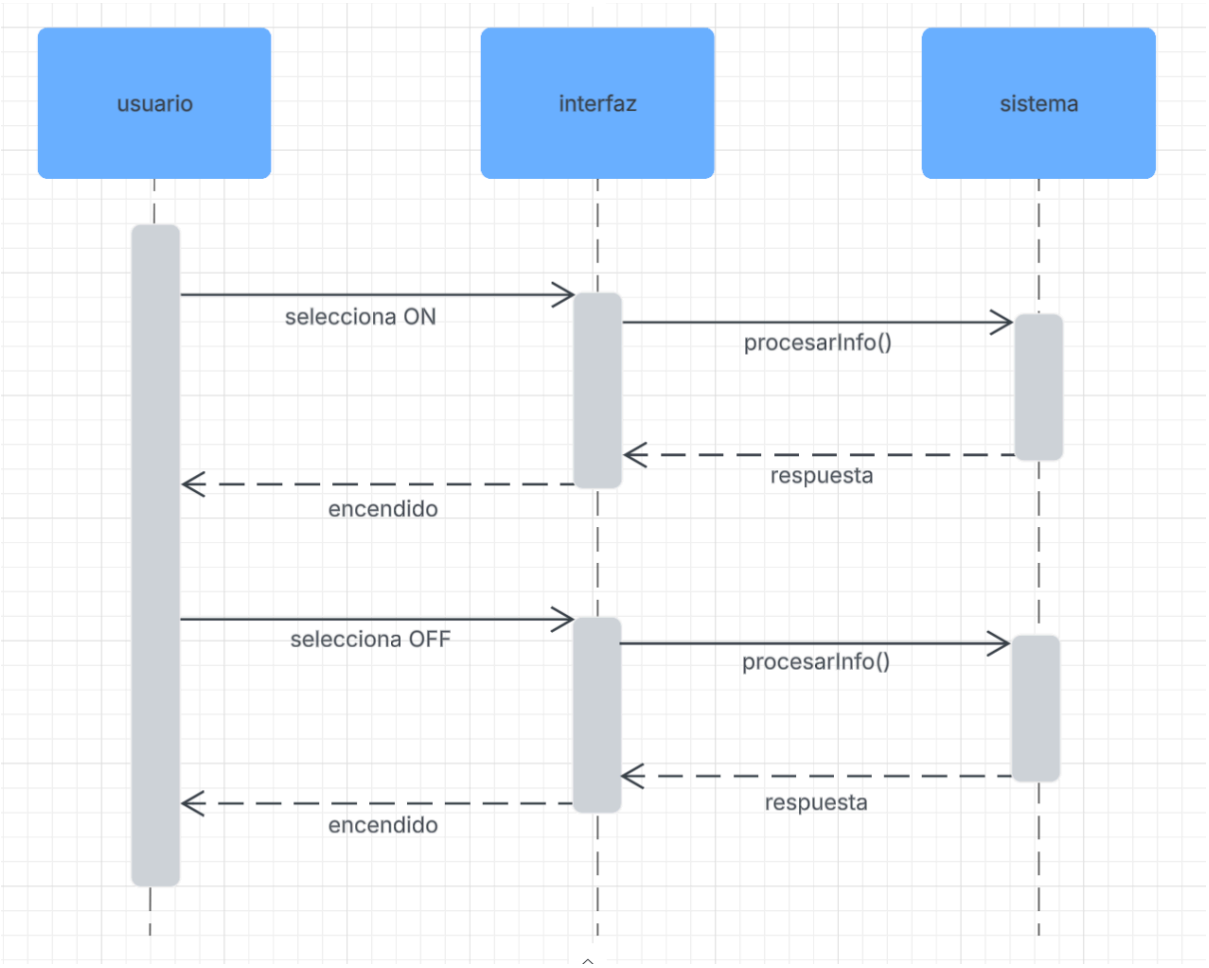



**Diagrama 3.** Diagrama de secuencia de encender y apagar el sistema.

## Descripción de la arquitectura

La arquitectura del sistema se basa en un modelo cliente-servidor embebido, donde la Raspberry Pi 4 actúa como servidor central encargado del procesamiento, coordinación y ejecución de las funciones críticas del proyecto. Los sensores y actuadores conectados funcionan como clientes que envían datos o ejecutan acciones según las instrucciones del sistema.

El servidor (Raspberry Pi 4) recibe continuamente la información entregada por el sensor de sonido mediante comunicación GPIO o I2C, interpreta los niveles de ruido y determina si se supera el umbral establecido. Tras el análisis, el servidor coordina la respuesta mediante la activación de un LED indicador o el envío de una notificación al usuario a través de los canales previamente configurados (como mensajería instantánea o correo electrónico).

La arquitectura cuenta con tres capas principales:

1. **Capa de adquisición de datos**
    - Compuesta por el sensor de sonido FC-04. (Sound Sensor Groove v1.6)
    - Captura niveles de ruido en tiempo real y envía los datos a la Raspberry Pi mediante señales digitales o analógicas procesadas por librerías de Python.

2. **Capa de procesamiento y lógica**
    - Ejecutada en la Raspberry Pi 4 usando Python. Se encarga de:
        - Interpretar valores del sensor.
        - Comparar con umbrales definidos.
        - Gestionar el encendido y apagado del sistema.
        - Controlar el LED indicador.
        - Preparar y enviar notificaciones al usuario.

3. **Capa de notificación e interacción**
    - Informa al usuario mediante:
        - LED indicador conectado a GPIO.
        - Notificaciones enviadas por la red (alertas push).
    - Permite que el usuario configure los parámetros del sistema (umbrales de ruido y ajustes de sensibilidad).

Esta arquitectura modular facilita la escalabilidad, permitiendo incorporar sensores adicionales o nuevas acciones de respuesta sin modificar la estructura principal del sistema.

## Herramientas y técnicas

Para el desarrollo del proyecto se utilizaron herramientas de software, hardware y técnicas metodológicas que permiten asegurar la correcta implementación, prueba y despliegue del sistema.

### Herramientas de Software

- **Raspberry Pi OS (Debian Buster/Bookworm):** Sistema operativo principal donde se ejecuta el programa de monitoreo y control.

- **Python 3:** Lenguaje principal de programación para la comunicación con sensores, manejo de GPIO y envío de notificaciones.
- **Librerías de Python:**
    - gpiozero/rpi.GPIO: Control de pines GPIO para el LED y el botón.
    - smbus/I2C: Comunicación con componentes conectados mediante buses digitales.
- **Thonny IDE:** Entorno de desarrollo utilizado para programar, depurar y ejecutar el código en la Raspberry Pi.
- **Visual Studio Code:** Utilizado para la edición del código y manejo de archivos del proyecto.
- **Redmine / Google Docs:** Herramientas de planificación, colaboración y registro del progreso semanal.

**Herramientas de Hardware**

- **Raspberry Pi 4:** Unidad principal de procesamiento y comunicación con sensores.
- **Sensor de sonido FC-04:** Responsable de medir niveles de ruido en dB.
- **LED indicador:** Elemento de señalización visual cuando se detectan niveles de ruido superiores al umbral.
- **Grove button ON/OFF:** Permite al usuario encender o apagar el sistema manualmente.
- **Tarjeta MicroSD:** Almacenamiento del sistema operativo y del software del proyecto.

**Técnicas utilizadas**

- **Programación modular en Python:** Separación de las funciones del sistema para mejorar la mantenibilidad
- **Pruebas incrementales:** Evaluación por etapas del funcionamiento del sensor, LED, notificaciones y la lógica de control.
- **Metodología incremental:** Desarrollo por iteraciones, avanzando cada semana según bitácoras del proyecto.
- **Modelado en VR (Meta Quest 3):** Construcción de una representación virtual del prototipo para validar el diseño.

## Conclusión

Durante el desarrollo del proyecto el problema principal que se presentó fue la dificultad en la instalación y configuración del sistema operativo en la Raspberry Pi 4B, además de conflictos con el uso de la librería grove pi. Estas fueron corregidas de manera efectiva durante el transcurso del proyecto, permitiendo llegar a los procesos finales del avance. Hasta ahora es esencial implementar la lógica de programación para el trabajo en conjunto con los sensores grove pi los que tienen como función ser capaces de detectar niveles de ruido. Se continúa el avance en cada semana del semestre en la cual se implementará el funcionamiento de la lógica con los sensores, el diseño de la carcasa a la Raspberry, el manual de usuario y las documentaciones correspondientes a la última fase del proyecto.

## Referencias

Python (Lenguaje de programación)
Van Rossum, G., & Drake, F. L. Jr. (2009). Python 3 reference manual. CreateSpace. https://ir.cwi.nl/pub/5008 citebay.com

Arch Linux (Distribución del sistema operativo)
Arch Linux Wiki. (n.d.). ArchWiki: Reading. https://wiki.archlinux.org/title/Help%3AReading wiki.archlinux.org

Raspberry Pi OS (Sistema operativo)
Raspberry Pi Foundation. (n.d.). Raspberry Pi documentation. https://www.raspberrypi.com/documentation/raspberrypi.com