# UNIVERSIDAD DE TARAPACÁ

## FACULTAD DE INGENIERÍA

**Departamento de Ingeniería en Computación e Informática**

# Sistema de monitoreo y control de un acuario

# “Safe Nemo”


**Autor(es):**

- Dylan Flores
- Cristian Gutiérrez
- Joaquin Jelves
- Cristobal Hernández


**Asignatura:** Proyecto 2

**Académico:** Diego Aracena

Arica, 24 NOVIEMBRE 2025

## Historial de cambios

| Fecha | Versión | Descripción | Autores |
|---|---|---|---|
| 30/09/2025 | 1.0 | Versión preliminar del formato | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |
| 07/10/2025 | 1.1 | Se estructuró el formato inicial del informe | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |
| 14/10/2025 | 1.2 | Panorama General y Organización del Proyecto | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |
| 21/10/2025 | 1.3 | Estimación de Costos y Planificación de Riesgos | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |
| 11/11/2025 | 1.4 | Corrección del informe 1 y Requerimientos y Casos de Uso | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |
| 18/11/2025 | 1.5 | Diagrama UML, Secuencia y Actividades | Dylan flores<br>Cristobal Hernandez<br>Joaquin Jelves<br>Cristian Gutierrez |

Índice de contenido.

## Índice de tablas.



## Índice de Ilustraciones.

# 1. Panorama General

## 1.1. Resumen del Proyecto

La realización de este proyecto se fundamenta en las significativas dificultades inherentes al cuidado y mantenimiento de acuarios, una labor que representa un desafío considerable tanto para aficionados como para negocios que dependen de ecosistemas acuáticos saludables. Si bien la estabilidad del entorno es vital para la supervivencia de las especies, la realidad actual obliga a realizar tareas críticas de supervisión y ajuste de parámetros —tales como temperatura, pH, luz y nitratos— de manera manual, tediosa y con una alta dependencia del conocimiento técnico del usuario. Por tanto, este proyecto se justifica ante la notable carencia de sistemas accesibles y confiables que permitan garantizar un entorno estable, buscando automatizar procesos para simplificar radicalmente las exigencias del mantenimiento y reducir el riesgo de error humano.

Para abordar dicha problemática, se propone el desarrollo del prototipo "Safe a Nemo", una solución que se justifica técnica y operativamente mediante el uso del Internet de las Cosas (IoT). La elección de una Raspberry Pi 4 como núcleo de procesamiento es estratégica, ya que ofrece una plataforma potente y de bajo consumo capaz de centralizar la recopilación de datos de sensores de alta precisión en tiempo real. Esta capacidad de procesamiento en el borde (edge computing) es esencial para dotar al sistema de autonomía, permitiéndole no solo monitorear variaciones en temperatura o pH, sino ejecutar acciones de control inmediatas y enviar alertas preventivas. Así, la implementación de esta tecnología, sumada a una interfaz de usuario accesible, valida la viabilidad del sistema para transformar la gestión de acuarios en un proceso eficiente y seguro.

Los datos recogidos por esta red de sensores son procesados continuamente por la Raspberry Pi, la cual no solo registra los valores, sino que también ejecuta la lógica de control necesaria. Esta capacidad de procesamiento en el borde es esencial para la funcionalidad de "Safe a Nemo", permitiendo tomar decisiones automatizadas y rápidas. Por ejemplo, si la temperatura cae por debajo de un umbral preestablecido, el sistema puede activar automáticamente un calentador conectado, o si el pH se desvía peligrosamente, puede enviar una alerta inmediata al usuario. Además, el proyecto contempla la implementación de una interfaz de usuario accesible, probablemente a través de una aplicación móvil o un *dashboard* web, donde los propietarios podrán visualizar los datos históricos y en tiempo real de su acuario desde cualquier ubicación, recibiendo notificaciones y alertas instantáneas sobre cualquier anomalía.

### 1.1.1. Propósito

El propósito de este proyecto es brindar una solución tecnológica que facilite el cuidado y mantenimiento de acuarios, tanto para personas con escaso conocimiento en este ámbito como para establecimientos que requieran un sistema automatizado de monitoreo, tales como tiendas de mascotas, restaurantes u otros negocios. El sistema busca garantizar el bienestar de los peces mediante la regulación y supervisión automática de los parámetros vitales del agua, simplificando las tareas de mantenimiento y promoviendo un ecosistema acuático estable mediante su monitoreo.

### 1.1.2. Alcance

Nuestro proyecto está diseñado para tener un alcance variado, desde entusiastas de la marina hasta pequeños negocios. Se comenzará con el uso personal, ofreciendo soluciones para acuarios domésticos, como guías sencillas, monitoreo del agua y automatización. Luego, se expandirá a tiendas de mascotas (pet shops) para mejorar la exhibición y el cuidado de sus peces, y también a restaurantes que deseen acuarios atractivos en su ambiente. Se ofrecerá desde el diseño y la instalación hasta el mantenimiento. El objetivo es que todos puedan acceder a las últimas innovaciones en acuariofilia.

### 1.1.3. Objetivo General

Diseñar y hacer funcional un prototipo de sistema de monitoreo y automatización para acuarios, usando una Raspberry Pi 4, que regule solo y autónomamente los parámetro vitales del agua (temperatura, el pH y el nivel de agua y luz) y que permite la supervisión de esos datos en remoto por una interfaz que usa la gente, todo esto con el fin de garantizar un ecosistema acuático que es estable y que simplificará la labor de mantenimiento para el aficionados que están comenzando.

### 1.1.4. Objetivos Específicos

- Seleccionar y caracterizar los componentes de hardware, incluyendo los sensores para la medición de temperatura (DS18B20), pH y nivel de agua (interruptor de flotador), así como los actuadores para el control del calefactor, la bomba de agua y el alimentador automático.
- Diseñar y ensamblar el circuito electrónico que interconecte de forma segura la Raspberry Pi 4 con los sensores y módulos de control (relés y servomotor), asegurando la integridad de los componentes de bajo voltaje.
- Desarrollar el software de control y gestión en lenguaje Python para la Raspberry Pi 4, el cual integrará la lógica de operación y la interfaz de usuario, cumpliendo las siguientes funciones:
    - **Procesamiento de datos:** Leer, analizar e interpretar la información proveniente de los sensores.
    - **Control automatizado:** Comparar las mediciones con umbrales de seguridad predefinidos y activar los actuadores de manera automática.
    - **Interfaz de usuario (Panel Web):** Implementar una plataforma accesible de forma local o remota para la visualización de parámetros

en tiempo real, recepción de alertas y control manual de los actuadores.

- Construir e integrar la estructura física del prototipo, utilizando la maqueta 3D como guía para organizar los componentes de hardware de manera funcional y segura para entornos acuáticos.
- Realizar pruebas integrales del sistema en un entorno real para validar la precisión de las mediciones, la fiabilidad de las acciones automáticas y el correcto funcionamiento de la interfaz remota, documentando los resultados obtenidos.

### 1.1.5. Suposiciones y restricciones

- Suposiciones:
    - Se asume que los dispositivos actuadores (calefactor, bomba de agua y servomotor) operan dentro de sus parámetros nominales. Ante una falla de hardware, el sistema activará un protocolo de seguridad que notificará inmediatamente al usuario para prevenir daños en el ecosistema.
    - Se asume la disponibilidad constante de una red local o conexión a internet. En caso de pérdida de conectividad, el sistema continuará operando de forma autónoma mediante lógica local (*edge computing*) para mantener los umbrales críticos de supervivencia.
    - Se asume que el sistema de monitoreo no solo detecta cambios, sino que ejecutará acciones correctivas automáticas (como el encendido del calefactor ante bajas temperaturas) y generará alertas visuales y remotas en tiempo real.
- Restricciones:
    - **Conocimiento del usuario:** Existe una limitación en el manejo técnico por parte del usuario final. Para mitigar esto, el sistema se restringirá a una interfaz intuitiva que no requiera conocimientos de programación para su operación básica.
    - **Continuidad operativa:** El sistema es susceptible a fallos eléctricos o de red. Se establece como restricción que el hardware debe contar con un sistema de reinicio automático tras un corte de energía para recuperar el monitoreo lo antes posible.
    - **Ambiente corrosivo y humedad:** Debido a que el sistema opera cerca de un acuario, existe la restricción de proteger la electrónica. Todos los componentes de control deben estar aislados en una caja estanca (grado IP) para evitar cortocircuitos por evaporación o salpicaduras.
    - **Presupuesto y Hardware:** El prototipo está restringido al uso de una Raspberry Pi 4 y sensores de bajo costo. Esto limita la precisión extrema de laboratorio, pero asegura que la solución sea económicamente viable para el usuario objetivo.
    - **Seguridad de datos:** Al ser un dispositivo conectado a la red (IoT), el sistema tiene la restricción de operar bajo protocolos de autenticación básicos para evitar accesos no autorizados a los actuadores (como el alimentador o el calefactor).

### 1.1.6. Entregables del Proyecto

Los entregables del proyecto son:

1. Maqueta del sistema
2. Presentaciones
3. Informes
4. Redmine UTA (wiki, bitácoras y carta Gantt)
5. Video explicativo de la maqueta del proyecto en VR
6. Manual de usuario
7. Poster
8. Sistema de “Safe a Nemo”

### 1.1.7. Maqueta del Proyecto

Esta maqueta representa "Safe a Nemo", un sistema IoT impulsado por Raspberry Pi 4 que automatiza el control de parámetros vitales como temperatura y pH. La propuesta innova integrando el visor de realidad virtual Meta Quest para visualizar el estado del acuario, permitiendo un monitoreo inmersivo en realidad mixta para una gestión más eficiente e intuitiva.

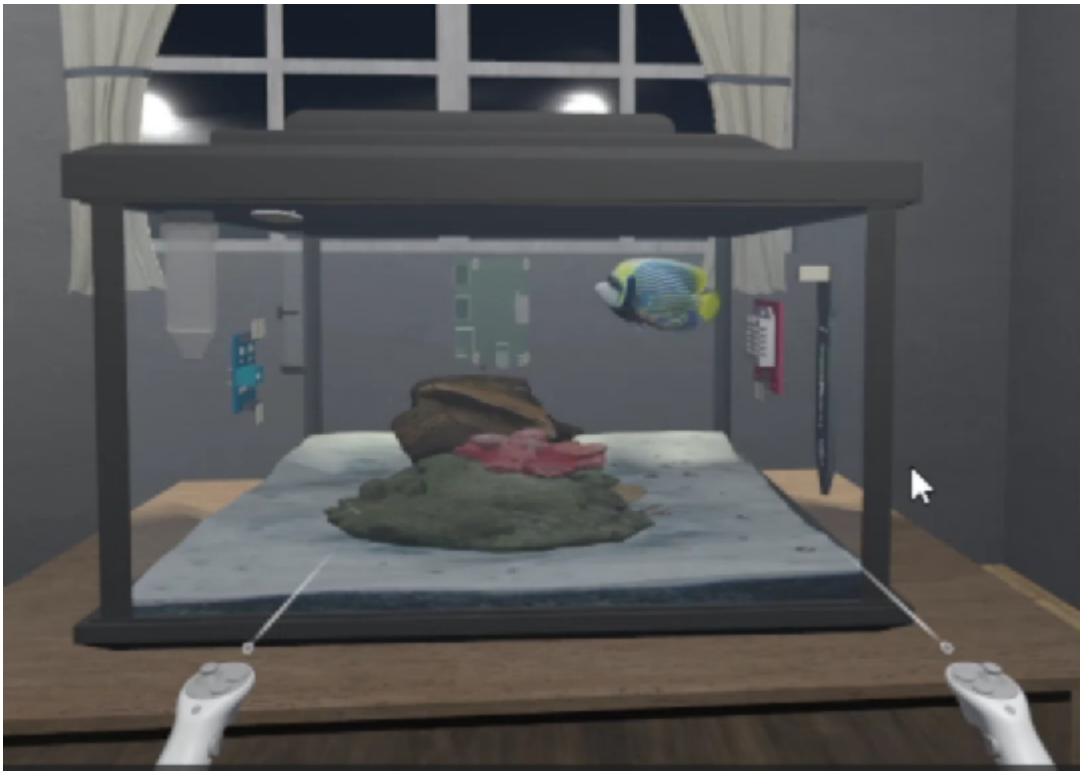


Ilustración 1: Maqueta Acuario

# 2. Organización del proyecto

## 2.1. Personal y entidades internas.

Para la organización se definió algunos roles para mejorar la coordinación:

**Jefe de grupo:** Persona encargada de la coordinación general del proyecto. Define los objetivos, asigna tareas, supervisa el cronograma y es el principal punto de contacto para la toma de decisiones. Es el líder.
**Programador:** Responsable de escribir, probar y mantener el código que se ejecutará en Raspberry Pi 4. Esto incluye la configuración del sistema operativo de la Raspberry Pi, la programación de la lógica principal del proyecto y la integración de librerías para interactuar con sensores o periféricos.
**Documentador:** Encargado de crear y mantener toda la documentación escrita del proyecto. Esto abarca desde el diseño inicial, las bitácoras, informes, presentaciones, administración de Redmine (diseño Wiki, subir archivos y carta Gantt) , hasta el manual de usuario.
**Técnico de hardware:** Responsable del montaje físico, la conexión de componentes (sensores, actuadores) a los pines de la Raspberry Pi, la gestión de la alimentación y la identificación/solución de problemas físicos o de interfaz electrónica. Se asegura de que la configuración física sea robusta y funcional.

## 2.2. Roles y responsabilidades

Los roles de cada integrantes son:

| Rol | Responsable |
|---|---|
| Jefe de Grupo | Cristian Gutierrez |
| Programador | Dylan Flores - Cristobal Hernández |
| Documentador | Cristian Gutierrez |
| Técnico de hardware | Joaquin Jelves |

Tabla 1: Roles y Responsables

## 2.3. Mecanismos de comunicación y organización

**WhatsApp:** Se usa para la organización y es la principal fuente de comunicación. Aquí se ponen de acuerdo con los horarios de las reuniones y se comparten cosas importantes al instante, como enlaces, archivos o referencias.
**Redmine:** Funciona como la oficina principal del proyecto. Aquí se guarda toda la documentación, se lleva el control de las tareas con el calendario y el "plan de trabajo" (Carta Gantt), y se presenta la información general del proyecto (como una enciclopedia interna o *wiki*).
**Google Drive:** Es el almacén de archivos del equipo. Ayuda a organizar todos los documentos, enlaces, videos y referencias del proyecto, haciendo que sea fácil acceder a ellos y trabajar en ellos al mismo tiempo.
**GitHub:** Control de versiones del código del proyecto. Es el mecanismo principal para que el Programador gestione, colabore y fusione cambios en el código de forma segura, evitando sobrescribir el trabajo de otros.

# 3. Planificación de los procesos de gestión

## 3.1. Planificación inicial del proyecto

En la parte de hardware, se utilizarán los siguientes productos:

- Computadores
- Smartphone
- Raspberry Pi 4
- Kit GrovePi o GPIO
- Tarjeta SD 32 Gb
- Sensores
- Accionadores

Mientras que para la parte de software:

- Visual Studio Code
- Unity Hub
- Blender
- Canva
- Meta Link
- Meta Link Developer
- Python

### 3.1.1. Planificación de estimaciones

- Costos de Hardware:

| Producto | Cantidad | Costo Unidad | Costo Total |
|---|---|---|---|
| Computador(personal) | 4 | $70.000 | $280.000 |
| Raspberry Pi 4 | 1 | $110.000 | $110.000 |
| Kit GrovePi o GPIO | 1 | $132.000 | $132.000 |
| Pantalla LCD | 1 | $8.000 | $8.000 |
| Sensor Ultrasónico | 1 | $2.500 | $2.500 |
| Sensor de Luz | 1 | $2.000 | $2.000 |
| Luz Led | 1 | $2.990 | $2.990 |
| Tarjeta SD 32 GB | 1 | $8.000 | $8.000 |
| Sensor Temperatura | 1 | $ 1.990 | $ 1.990 |
| Sensor de Ph | 1 | $ 24.990 | $ 24.990 |
| Calefactor | 1 | $6.990 | $6.990 |
| Monitor LG | 1 | $119.990 | $119.990 |

| | |
|---|---|
| **Total** | **$699.450** |

Tabla 2: Costos Hardware

- Costos de Software:

| Producto | Costo | Costo Total |
|---|---|---|
| Visual Studio Code | $0 | $0 |
| Canva | $0 | $0 |
| Documentos Google | $0 | $0 |
| Unity Hub | $0 | $0 |
| Blender | $0 | $0 |
| Meta Link | $0 | $0 |
| Meta Link Developer | $0 | $0 |
| Raspberry PI OS | $0 | $0 |
| Python | $0 | $0 |
| **Total** | | $0 |

Tabla 3: Costos Software

### 3.1.2. Planificación de Recursos Humanos

- Costos de Recursos Humanos:

| Rol | Cantidad por rol | Costo/Hora | Hora Mensual | Meses | Costo Total |
|---|---|---|---|---|---|
| Jefe de Grupo | 1 | $12.000 | 48 | 4 | $2.304.000 |
| Programador | 2 | $9.500 | 48 | 3 | $2.736.000 |
| Documentador | 1 | $7.000 | 24 | 4 | $672.000 |
| Técnico de Hardware | 1 | $8.000 | 48 | 2 | $768.000 |
| **Total por 4 meses** | | | | | **$6.480.000** |

Tabla 4: Costos Recursos Humanos

### 3.1.3. Costos totales

- Costo Totales

| Elemento | Costo Total |
|---|---|
| Hardware | $699.450 |
| Software | $0 |
| Recursos Humanos | $6.480.000 |
| **Total** | **$7.179.450** |

Tabla 5: Costos Totales

## 3.2. Distribución de tiempos

### 3.2.1. Carta Gantt

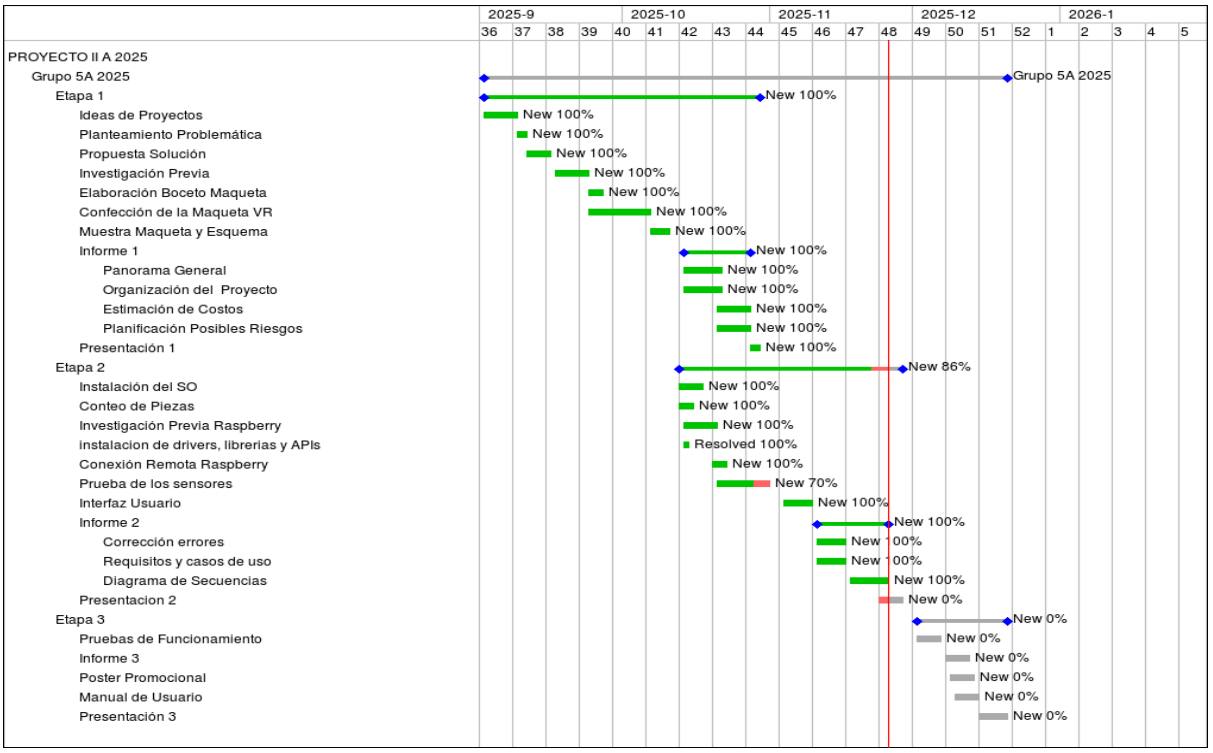



Ilustración 2: Carta Gantt

### 3.2.2. Asignación de tiempo

El proyecto está planificado para desarrollarse a lo largo de 15 semanas, con una distribución de trabajo semanal que incluye 6 horas de trabajo en clases y 6 horas de trabajo autónomo, lo que da un total estimado de 180 horas de trabajo. Estas 15 semanas se han dividido en 3 fases, las cuales se encuentran en la siguiente tabla.

| Fase | Semanas Transcurridas | Fecha |
|---|---|---|
| 1 | Desde la 1° semana hasta la 8° semana | Del 02 de septiembre hasta el 28 de octubre |
| 2 | Desde la 9° semana hasta la 12° semana | Del 29 de octubre hasta el 25 de noviembre |
| 3 | Desde la 13° semana hasta la 15° semana | Del 26 de noviembre hasta el 16 de diciembre |

Tabla 6: Asignación de tiempos

## 3.3. Planificación de la gestión de riesgos

Se elabora una tabla para establecer y analizar los posibles riesgos del proyecto y las medidas a tomar para solucionarlos, según su nivel de impacto. Los niveles de impacto son:

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

| Riesgos | Probabilidad de Ocurrencia | Nivel de Impacto | Acción Remedial |
|---|---|---|---|
| Pérdida de calibración de sensores | 60% | 2 | Programar una alerta en la interfaz de usuario que recuerde "Calibrar Sonda de pH" cada 15 días. |
| Corrupción de la Tarjeta SD de la Raspberry Pi | 50% | 1 | Programar copias de seguridad (backups) automáticas del código en un repositorio Git. |
| Fallo de la red Wi-Fi | 50% | 3 | Diseñar el sistema para que sea autónomo. El control vital (calefactor, nivel) debe seguir funcionando localmente en la Raspberry Pi aunque no tenga internet. |
| Indisponibilidad de un miembro clave | 50% | 3 | Fomentar la documentación interna y el trabajo en pares. Usar un repositorio centralizado (como GitHub) donde todo el código está comentado, para que otro programador pueda tomar el relevo si es necesario. |
| Consumo excesivo de recursos de la Raspberry Pi | 20% | 3 | Monitorear el uso de CPU y memoria de la Raspberry Pi durante las pruebas. |
| Desgaste acelerado por corrosión | 10% | 2 | Aislar. Asegurarse de que toda la electrónica (Raspberry Pi, relés) esté en una caja cerrada y *lejos* de las salpicaduras. |

Tabla 7: Gestión de Riesgos

# 4. Planificación de los procesos técnicos

## 4.1. Modelo de proceso

### 4.1.1. Requerimientos

Los requerimientos funcionales y no funcionales son esenciales para el desarrollo y diseño de sistemas, brindando una base importante para crear soluciones tecnológicas que satisfacen tanto las necesidades como las expectativas de sus usuarios.

- **Requerimientos Funcionales:**

| ID | Requisito Funcional | Descripción |
|---|---|---|
| RF-1 | Registro de temperatura del acuario | El sistema debe registrar los valores del sensor de temperatura |
| RF-2 | Registro de ph del acuario | El sistema debe registrar los valores del sensor de ph |
| RF-3 | Registro de nivel del agua del acuario | El sistema registrar los valores del nivel del agua a partir del sensor ultrasónico |
| RF-4 | Registra de nivel de luminosidad del acuario | El sistema debe registrar los valores de luminosidad a partir del sensor de luz |
| RF-5 | Mostrar la temperatura actual del acuario en la GUI | El sistema debe mostrar la temperatura por medio de la interfaz |
| RF-6 | Mostrar el ph actual del acuario en la GUI | El sistema debe mostrar el ph temperatura por medio de la interfaz |
| RF-7 | Mostrar el nivel de agua actual del acuario en la GUI | El sistema debe mostrar el del agua por medio de la interfaz |
| RF-8 | Mostrar el nivel de luminosidad actual del acuario en la GUI | El sistema debe mostrar el nivel de luminosidad por medio de la interfaz |
| RF-9 | Control de temperatura del acuario | El sistema debe activar el calefactor en caso de bajos niveles de T° |
| RF-10 | Control de luminosidad | El sistema debe activar las luces led cuando detecte bajos niveles de luminosidad |
| RF-11 | Alerta del nivel de agua del acuario | El sistema debe notificar al usuario cuando el nivel de agua no sea adecuado |
| RF-12 | Alerta del nivel de ph del acuario | El sistema debe notificar al usuario cuando el nivel de ph no sea apto |

| | | |
|---|---|---|
| RF-13 | Seleccionar la especie del pez | El usuario debe poder seleccionar la especie del pez que tendrá |
| RF-14 | Establecer parámetros por especie elegida | El sistema debe establecer parámetros para vida óptima de ph, temperatura, luminosidad de acuerdo a la especie elegida. |
| RF-15 | Establecer parámetros manualmente | El sistema debe establecer parámetros para vida óptima de ph, temperatura, luminosidad de forma manual. |
| RF-16 | Alimentar a los peces | El sistema debe alimentar los peces mediante el comedero |
| RF-17 | Alerta del nivel de comida del comedero | El sistema debe notificar al usuario cuando los niveles de comida sean bajos |

Tabla 8: Requerimientos funcionales

- **Requerimientos No Funcionales:**

| ID | Requisito No Funcional | Descripción |
|---|---|---|
| RNF-1 | Tiempo de respuesta (Rendimiento) | La aplicación debe cargar la pantalla principal y los datos de los sensores en tiempo real (temperatura, pH, etc.) en menos de 5 segundos bajo condiciones de red normales. |
| RNF-2 | Frecuencia de actualización de datos (Rendimiento) | El sistema debe actualizar los datos de los sensores al menos cada 1 minuto cuando la aplicación esté activa. |
| RNF-3 | Capacidad (Rendimiento) | El sistema debe ser capaz de manejar registros y alertas para al menos 1 acuario por usuario sin degradación del rendimiento. |
| RNF-4 | Autorización de acceso (Seguridad) | Solo el usuario propietario debe tener acceso a los datos y configuraciones de su acuario. |
| RNF-5 | Interfaz intuitiva (Usabilidad) | La aplicación debe tener una navegación clara y una visualización de datos fácil de interpretar |
| RNF-6 | Consistencia (Usabilidad) | El diseño y el flujo de trabajo deben ser consistentes en todas las pantallas de la aplicación. |

| | | |
|---|---|---|
| RNF-7 | Compatibilidad de dispositivos (Operacional) | La aplicación móvil debe ser compatible con al menos android 8.0 en adelante |
| RNF-8 | Mantenimiento (Operacional) | La aplicación debe ser fácil de actualizar y añadir nuevas características o parches de seguridad |

Tabla 9: Requerimientos no funcionales

- **Restricciones de diseño:**

| ID | Descripción |
|---|---|
| RST-01 | El sistema **debe** ser implementado usando una microcomputadora **Raspberry Pi 4**. |
| RST-02 | El lenguaje de programación para la lógica de control principal **debe** ser **Python 3**. |
| RST-03 | El sistema **debe** utilizar la placa de expansión **GrovePi+** para la conexión de los sensores Grove. |
| RST-04 | El sistema **debe** utilizar los sensores específicos definidos en el plan de proyecto (DS18B20, Sonda BNC de pH, Ultrasónico Grove). |
| RST-05 | El costo de adquisición del hardware no debe exceder el presupuesto total estimado de **$283.970 CLP**. |
| RST-06 | El prototipo funcional **debe** ser entregado acorde a las fechas estipuladas en la Carta Gantt del proyecto. |

Tabla 10: Restricciones de diseño

### 4.1.2. Descripción de la arquitectura

El proyecto se estructura bajo un modelo arquitectónico cliente-servidor.

- Cliente: Constituido por una aplicación móvil que faculta al usuario para la supervisión y gestión remota del sistema. Esta interfaz es la encargada de notificar alertas en tiempo real sobre las variables críticas del entorno.
- Servidor: Alojado en una Raspberry Pi, este componente ejecuta la lógica de control que interconecta los dispositivos de hardware (sensores y actuadores) con la interfaz de usuario.

Para garantizar la sincronización y transferencia de datos en tiempo real, es requisito indispensable que ambos subsistemas operen bajo la misma red Wi-Fi local.

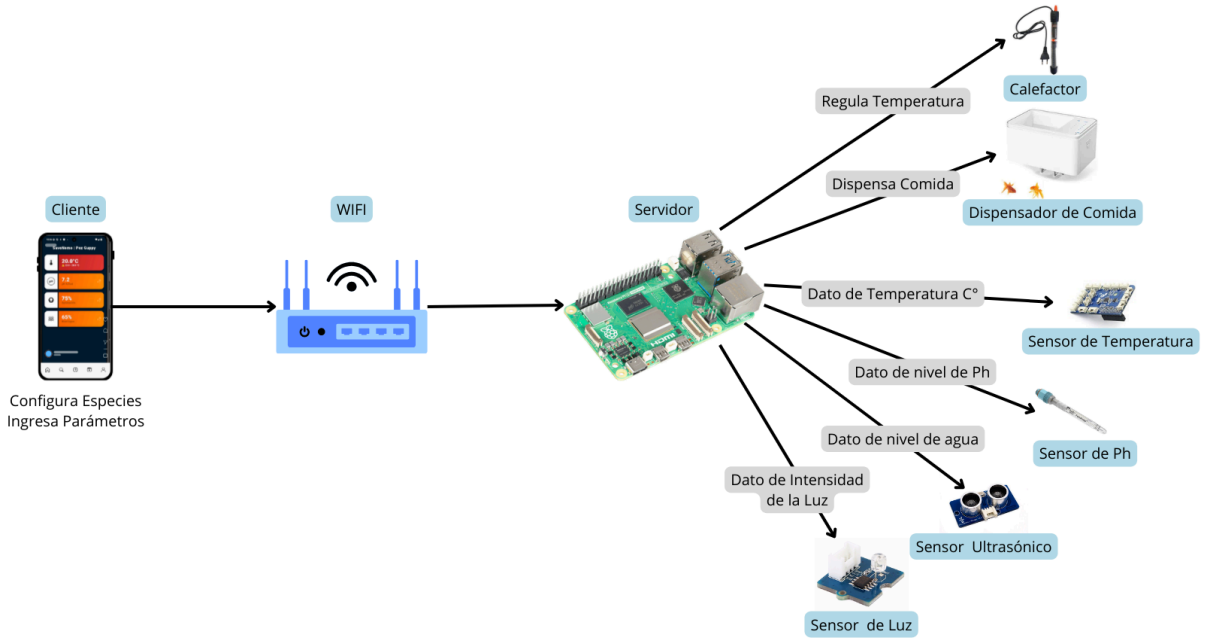
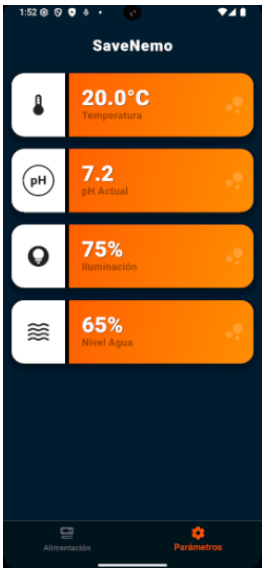



Ilustración 3: Descripción de arquitectura

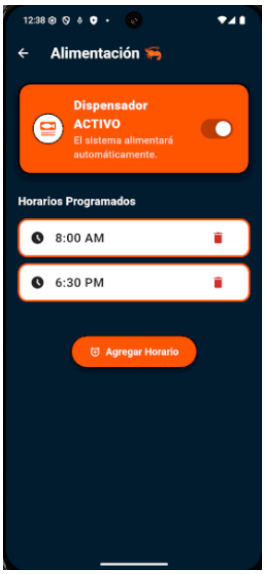
### 4.1.3. Diseño de interfaz de usuario




**Interfaz Monitoreo de parámetros:**

Se despliegan en pantalla las lecturas en tiempo real de las variables críticas captadas por los sensores.

Ilustración 4: Interfaz monitoreo de parámetros.




**Interfaz Configuración de parámetros:**

En esta sección, el usuario establece las condiciones objetivo para el hábitat del pez. Cuenta con dos modos:

- Al elegir un pez conocido, la aplicación completa automáticamente los valores de los parámetros.
- La opción "Personalizado" habilita la edición manual de todos los parámetros.

Ilustración 5: Interfaz configuración de parámetros.


**Interfaz Monitoreo de parámetros:**

Esta interfaz controla el funcionamiento del actuador encargado de la dispensación de comida. Al activar el sistema, se despliega una lista dinámica de horarios. El usuario puede agregar nuevos eventos temporales mediante un selector de hora (*Time Picker*) y eliminar eventos existentes.

Ilustración 6: Interfaz Programar alimentación .


**Interfaz Monitoreo de parámetros:**

Se ingresa la IP y el puerto para establecer la conexión remota.

Ilustración 7:  Interfaz Establecer Conexión .


**Secuencia de Pantallas:**

El diagrama de navegación de la aplicación 'Safe a Nemo' ilustra la ruta de interacción que sigue el usuario, permitiéndole la supervisión y gestión integral de las variables del acuario.

Ilustración 8: Secuencia Pantallas

### 4.1.4. Caso de uso general

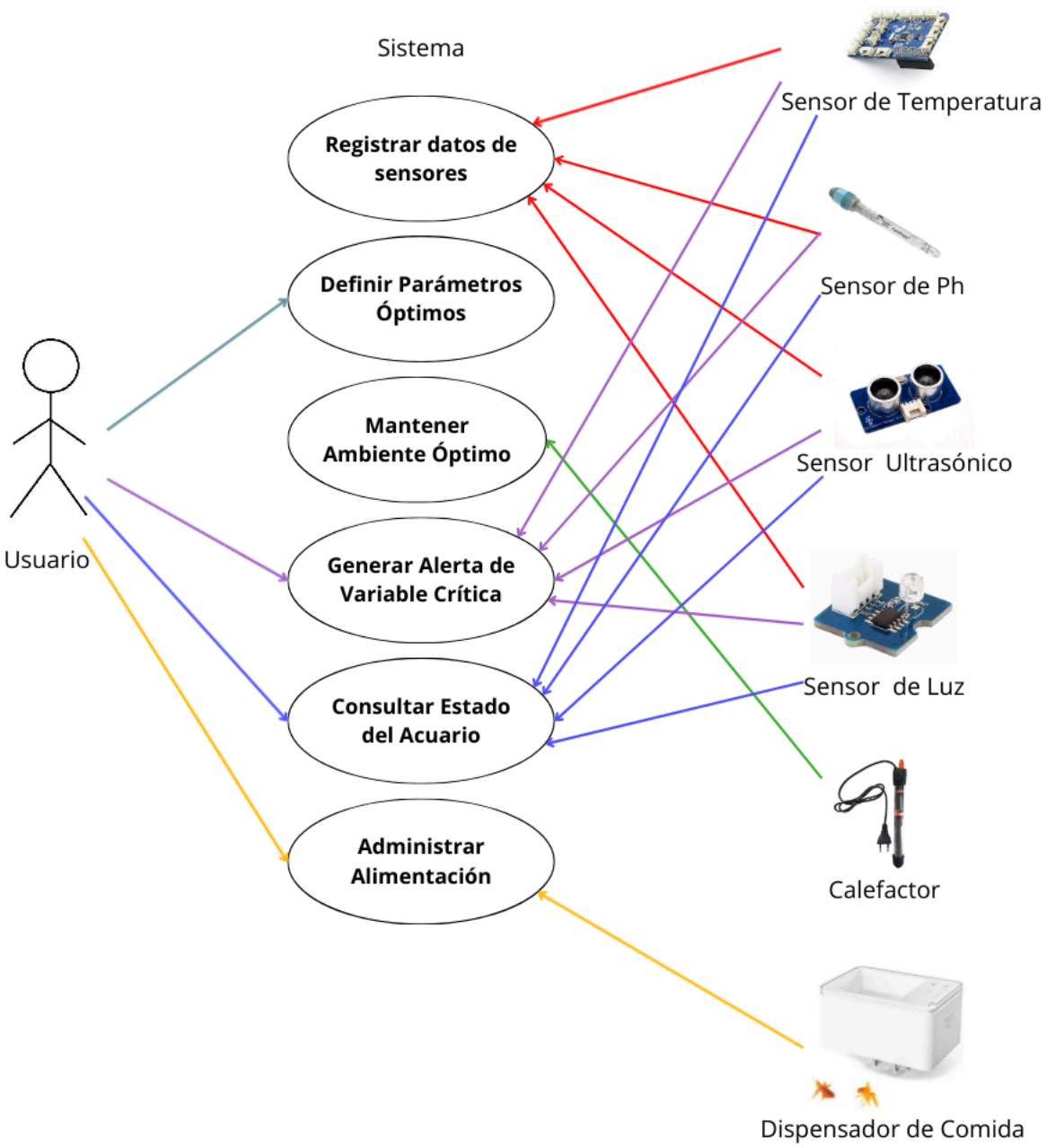



Ilustración 9: Caso de uso general

### 4.1.5. Casos de uso

<table>
<tr><td colspan="2">Nombre CU: Registrar datos de sensores (CU-01)</td></tr>
<tr><td colspan="2">Descripción: El sistema adquiere, procesa y almacena periódicamente las lecturas de los sensores de temperatura, pH, nivel de agua y luminosidad del acuario para mantener un historial y permitir la monitorización en tiempo real.</td></tr>
<tr><td colspan="2">Actor(es): Sensores (Temp, pH, Nivel, Luz)</td></tr>
<tr><td colspan="2">Precondición: El sistema debe estar encendido y los sensores deben estar conectados y operando</td></tr>
<tr><td colspan="2">RF Asociados: RF-1 (Registro de temperatura), RF-2 (Registro de pH), RF-3 (Registro de Nivel de Agua), RF-4 (Registro de Luminosidad)</td></tr>
<tr><td>Flujo Principal: Sensores<br><br>3. Los sensores envían los valores eléctricos/digitales actuales.</td><td>Flujo Principal: Sistema<br><br>1. El sistema inicia un ciclo de registro de datos programado (basado en RNF-2, al menos cada 1 minuto).<br><br>2. El sistema adquiere la lectura del sensor de temperatura (RF-1), sensor de pH (RF-2), sensor ultrasónico (RF-3), sensor de luz (RF-4).<br><br>4. El sistema registra los cuatro valores (Temperatura, pH, Nivel de Agua y Luminosidad) en la base local.<br><br>5. El sistema espera hasta el próximo ciclo de registro programado.</td></tr>
<tr><td>Flujo alternativo</td><td>Error en la Lectura de un Sensor:<br><br>2.1. Si un sensor devuelve un valor nulo o fuera de rango (indicando un posible error), el sistema registra el error, usa el último valor conocido y continúa con la lectura del siguiente sensor.</td></tr>
<tr><td colspan="2">Postcondiciones: Los valores de Temperatura, pH, Nivel de Agua y Luminosidad han sido almacenados en el historial del sistema con su respectiva marca de tiempo.</td></tr>
<tr><td colspan="2">Valor medible: Permite la creación de un historial de datos ambientales del acuario, lo que facilita el análisis de tendencias y la identificación de problemas a largo plazo en el ecosistema.</td></tr>
</table>

Tabla 11: CU-01:  Registrar datos de sensores

| **Nombre CU: Definir Parámetros Óptimos (CU-02)** | |
|---|---|
| **Descripción:** Permite al usuario establecer los rangos ideales (T°, pH, luminosidad) para su acuario, ya sea seleccionando una especie de pez (RF-14) o ingresándolos manualmente (RF-15). | |
| **Actor(es):** Usuario | |
| **Precondición:** El acuario debe estar conectado al sistema | |
| **RF Asociados:** RF-13, RF-14, RF-15 | |
| **Flujo Principal: Usuario** | **Flujo Principal: Sistema** |
| 1. El usuario navega a la sección de Configuración de Parámetros. | 2. El sistema muestra las opciones: a) Seleccionar Especie o b) Ingresar Manualmente. |
| 3. El usuario selecciona la opción a) Seleccionar Especie. | 4. El sistema muestra una lista de especies de peces disponibles (RF-13). |
| 5. El usuario selecciona la especie de pez. | 6. El sistema carga los parámetros óptimos predefinidos. |
| 7. El usuario confirma los parámetros mostrados. | 8. El sistema guarda los parámetros como valores de referencia. |
| **Flujo alternativo** (ingreso manual)**:**<br>3.1 El usuario selecciona la opción b) Ingresar Manualmente en el paso 2. | 4.1. El sistema muestra un formulario vacío para el ingreso de rangos. |
| 5.1. El usuario ingresa y guarda los valores. | 6.1. El sistema valida la lógica de los rangos y los guarda. (Continúa al paso 8). |
| **Postcondiciones:** Los parámetros de referencia del acuario quedan actualizados y se usan para el control automático y las alertas. | |
| **Valor medible:** Porcentaje de acuarios con parámetros configurados automáticamente vs. manualmente. | |

Tabla 12: CU-02:  Definir Parámetros Óptimos

| **Nombre CU: Mantener Ambiente Óptimo (CU-03)** | |
|---|---|
| **Descripción:** El sistema monitorea continuamente las lecturas de los sensores (Temperatura y Luminosidad) y activa los dispositivos de control (Calefactor y Luces) para asegurar que el acuario se mantenga dentro de los parámetros de vida óptima preestablecidos. (RF-9, RF-10, RF-14, RF-15) | |
| **Actor(es):** Calefactor (actuador), Luces LED (actuador) | |
| **Precondición:** Los parámetros de Temperatura y Luminosidad óptimos han sido definidos (CU-02). | |
| **RF Asociados:** RF-9 (Control de temperatura), RF-10 (Control de luminosidad) | |
| **Flujo Principal: Actuador**<br><br>4. El actuador se activa para corregir el ambiente. | **Flujo Principal: Sistema**<br><br>1. El sistema recibe los datos de los sensores (RF-1 Temperatura y RF-4 Luminosidad, activados por CUS-01).<br><br>2. El Sistema detecta valores fuera de rango<br><br>3.El Sistema envía señal de activación al Actuador pertinente.<br><br>5. El sistema registra cualquier cambio de estado del dispositivo y regresa al paso 1 para esperar el siguiente ciclo de datos. |
| **Flujo alternativo** | **Control Manual (CU-2):**<br><br>2.1. El Sistema detecta valores dentro de rango<br><br>3.1. El Sistema envía señal de desactivación al Actuador pertinente. |
| **Postcondiciones:** La temperatura y la luminosidad del acuario se mantienen activamente dentro de los rangos óptimos y los cambios son registrados. | |
| **Valor medible:** Tiempo de Estancia en Parámetros Óptimos (TEPO). Debe ser mayor o igual al 90% del tiempo de operación para considerarse exitoso. | |

Tabla 13: CU-03: Mantener Ambiente Óptimo

| **Nombre CU: Generar Alerta de Variable Crítica (CU-04)** | |
|---|---|
| **Descripción:** El sistema realiza un monitoreo automatizado con una frecuencia de muestreo de cada 15 minutos para el pH y de manera continua para el nivel de agua. Estos intervalos aseguran la estabilidad del ecosistema según estándares biológicos, notificando al usuario si los valores divergen del rango de seguridad definido (RF-11, RF-12). | |
| **Actor(es):** Sistema (Raspberry Pi 4) y Sensores (DS18B20 / pH / Flotador). | |
| **Precondición:** Los parámetros de pH y nivel de agua óptimos han sido configurados previamente en el sistema (CU-02). | |
| **RF Asociados:** RF-11 (Alerta nivel de agua), RF-12 (Alerta nivel de ph) | |
| **Flujo Principal: Usuario**<br><br>5. El usuario recibe la notificación | **Flujo Principal: Sistema**<br><br>1. El sistema activa el ciclo de lectura cada 15 minutos para pH y monitoreo constante para el nivel.(RF-2 y RF-3)<br><br>2. El sistema compara el valor actual con los umbrales de vida óptima establecidos por la literatura técnica.<br><br>3. Si el valor está fuera del rango aceptable (RF-11 o RF-12), el sistema registra el evento como Alerta Crítica.<br><br>4. El sistema genera y envía una notificación al usuario |
| **Flujo alternativo (No aplica)** | |
| **Postcondiciones:** Se envía al usuario una notificación de alerta crítica. | |
| **Valor medible:** Reducción del riesgo de mortalidad biológica al garantizar que ninguna desviación crítica de pH o nivel persista por más de 15 minutos sin intervención o conocimiento del usuario. | |

Tabla 14: CU-04:  Generar Alerta de Variable Crítica

| **Nombre CU: Consultar Estado del Acuario (CU-05)** | |
|---|---|
| **Descripción:** Permite al Usuario acceder a la aplicación para ver los valores más recientes de los sensores (T°, pH, Nivel de Agua y Luminosidad) a través de la interfaz gráfica (GUI). | |
| **Actor(es):** Usuario | |
| **Precondición:** Los sensores deben estar enviando datos al sistema. | |
| **RF Asociados:** RF-5, RF-6, RF-7, RF-8 | |
| **Flujo Principal: Usuario**<br><br>1. El usuario accede a la aplicación o navega a la pantalla principal de Monitoreo.<br><br>4. El usuario consulta la temperatura (RF-5), pH (RF-6), nivel de agua (RF-7) y luminosidad (RF-8). | **Flujo Principal: Sistema**<br><br>2. El sistema recupera los datos más recientes registrados por los sensores.<br><br>3. El sistema actualiza la Interfaz Gráfica (GUI) con los valores recuperados.<br><br>5. El sistema muestra los datos y continúa refrescándose periódicamente. |
| **Flujo alternativo (No aplica)** | |
| **Postcondiciones:** El sistema muestra la interfaz con los datos consultados, listos para que el usuario pueda observarlos. | |
| **Valor medible:** El usuario puede ver el estado actual del acuario y tomar decisiones informadas. | |

Tabla 15: CU-05:  Consultar Estado del Acuario

| **Nombre CU: Administrar Alimentación (CU-06)** | |
|---|---|
| **Descripción:** Permite al usuario iniciar la alimentación de los peces a través del comedero (ya sea programada o manual) y notifica al usuario cuando el nivel de comida en el depósito es bajo (RF-16, RF-17). | |
| **Actor(es):** Usuario, Sistema (Raspberry Pi/Controlador) | |
| **Precondición:** El comedero debe estar conectado y con comida en el depósito. | |
| **RF Asociados:** RF-16 (Alimentar a los peces), RF-17 (Alerta del nivel de comida) | |
| **Flujo Principal: Usuario**<br><br>1. El usuario navega a la sección de control de Alimentación.<br><br>2. El usuario selecciona la opción "Alimentar Ahora". | **Flujo Principal: Sistema**<br><br>3. El sistema verifica si es hora de una alimentación programada o si ha recibido una solicitud de alimentación manual.<br><br>4. El sistema envía una señal al comedero para dispensar la porción de comida (RF-16).<br><br>5. El sistema registra el evento de alimentación (hora y porción).<br><br>6. El sistema verifica el nivel de comida restante en el depósito.<br><br>7. Si el nivel de comida es bajo, el sistema genera y envía una notificación al usuario (RF-17). |
| **Flujo alternativo** | Flujo Alternativo (Fallo en la Dispensación)<br><br>4.1. El comedero no responde o reporta un error de dispensación.<br><br>5.1. El sistema registra el error, reintenta la dispensación y, si falla nuevamente, genera una alerta de fallo de comedero al usuario. |
| **Postcondiciones:** Los peces han sido alimentados y, si el nivel de comida es bajo, el usuario ha sido notificado para reponer el depósito. | |

| |
|---|
| **Valor medible:** Garantiza la nutrición programada de los peces (**autonomía** del sistema) y evita interrupciones en la alimentación debido a depósitos vacíos. |

Tabla 16: CU-06:  Administrar Alimentación

### 4.1.6. Diagrama de clases

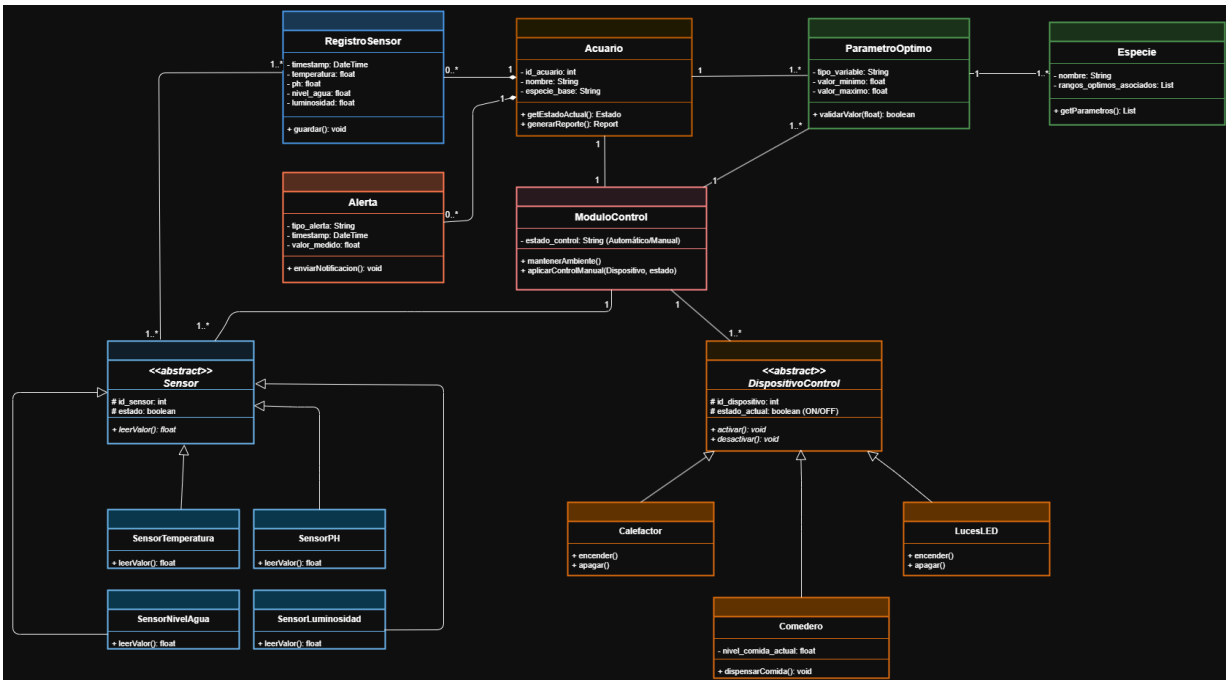



Ilustración 10: Diagrama UML

### 4.1.7. Diagramas de Actividades

- CU-01: Registrar datos de sensores

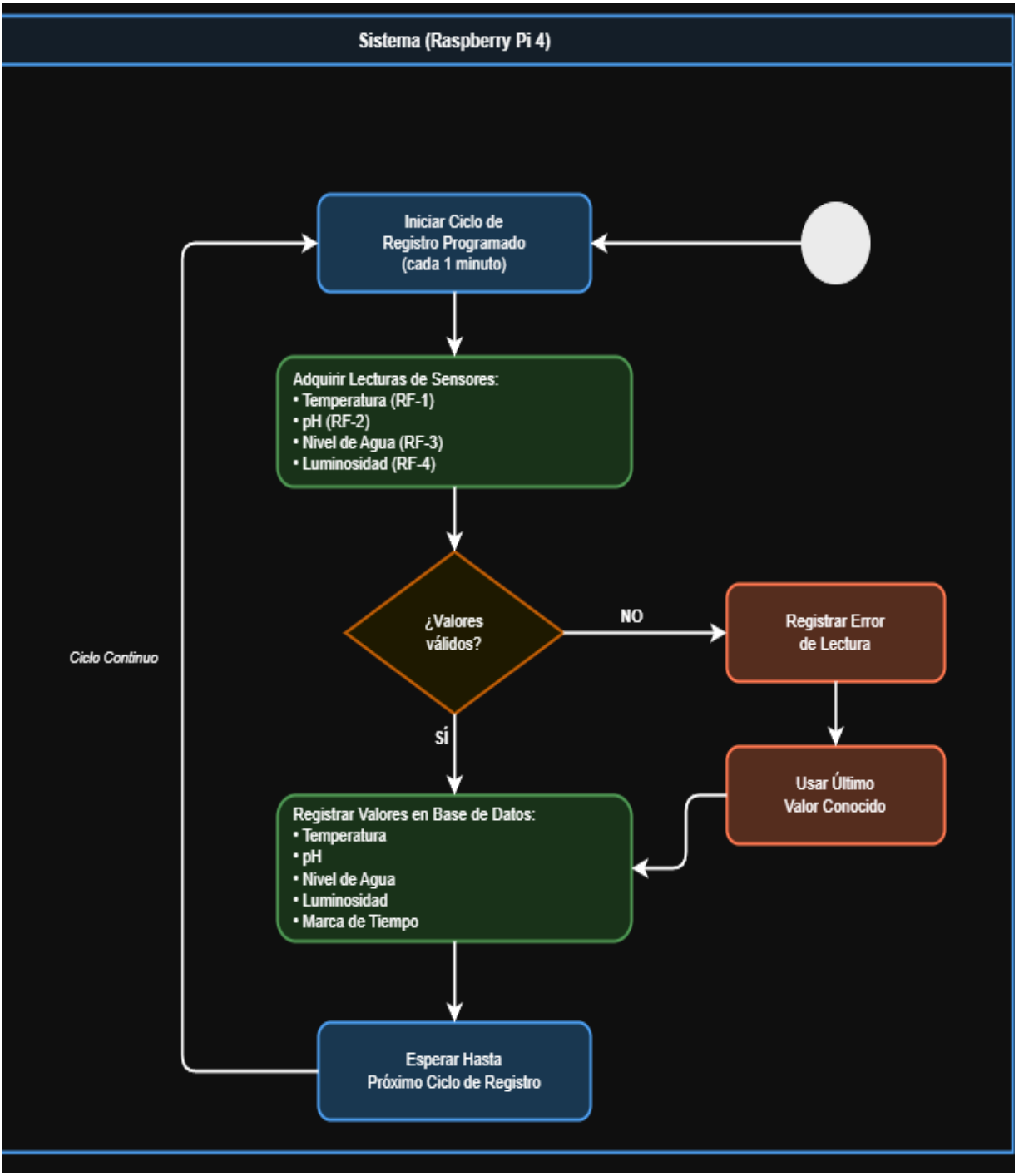



Ilustración 11: Diagrama actividades CU-01

- CU-04: Generar alerta de Variable Crítica

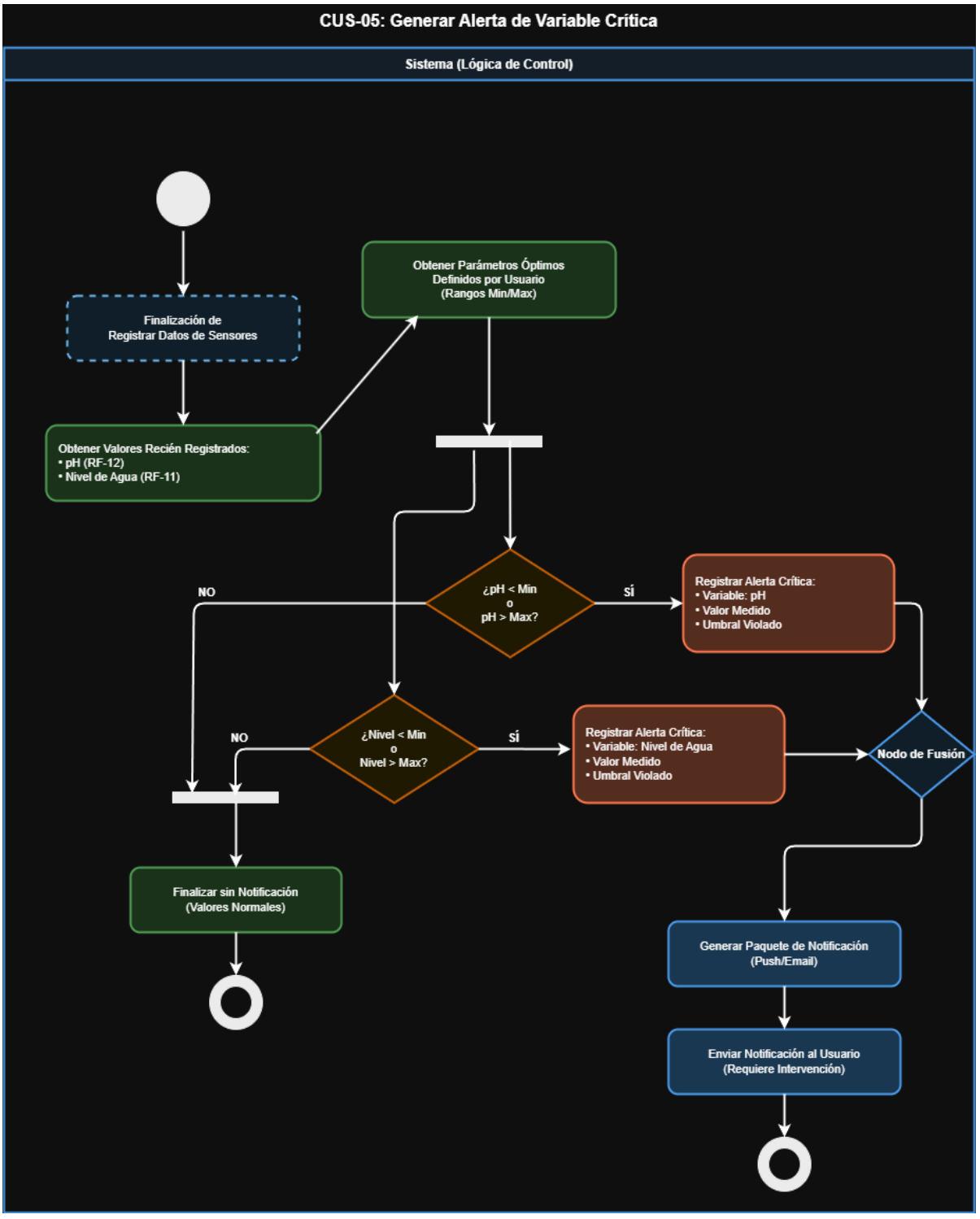



Ilustración 12: Diagrama actividades CU-04

- CU-05: Consultar Estado del Acuario

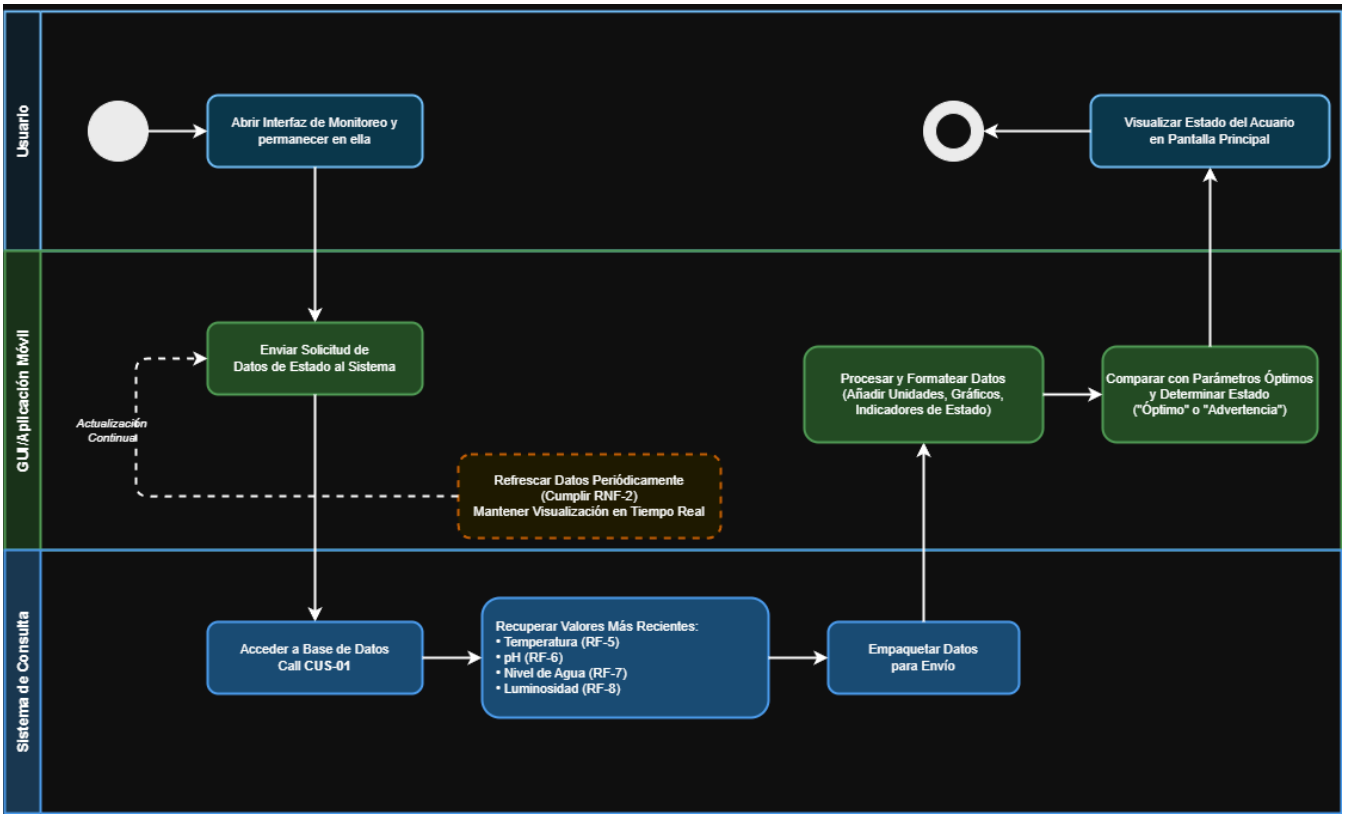



Ilustración 13: Diagrama actividades CU-05

### 4.1.8. Diagramas de Secuencia

- CU-02: Definir Parámetros óptimos

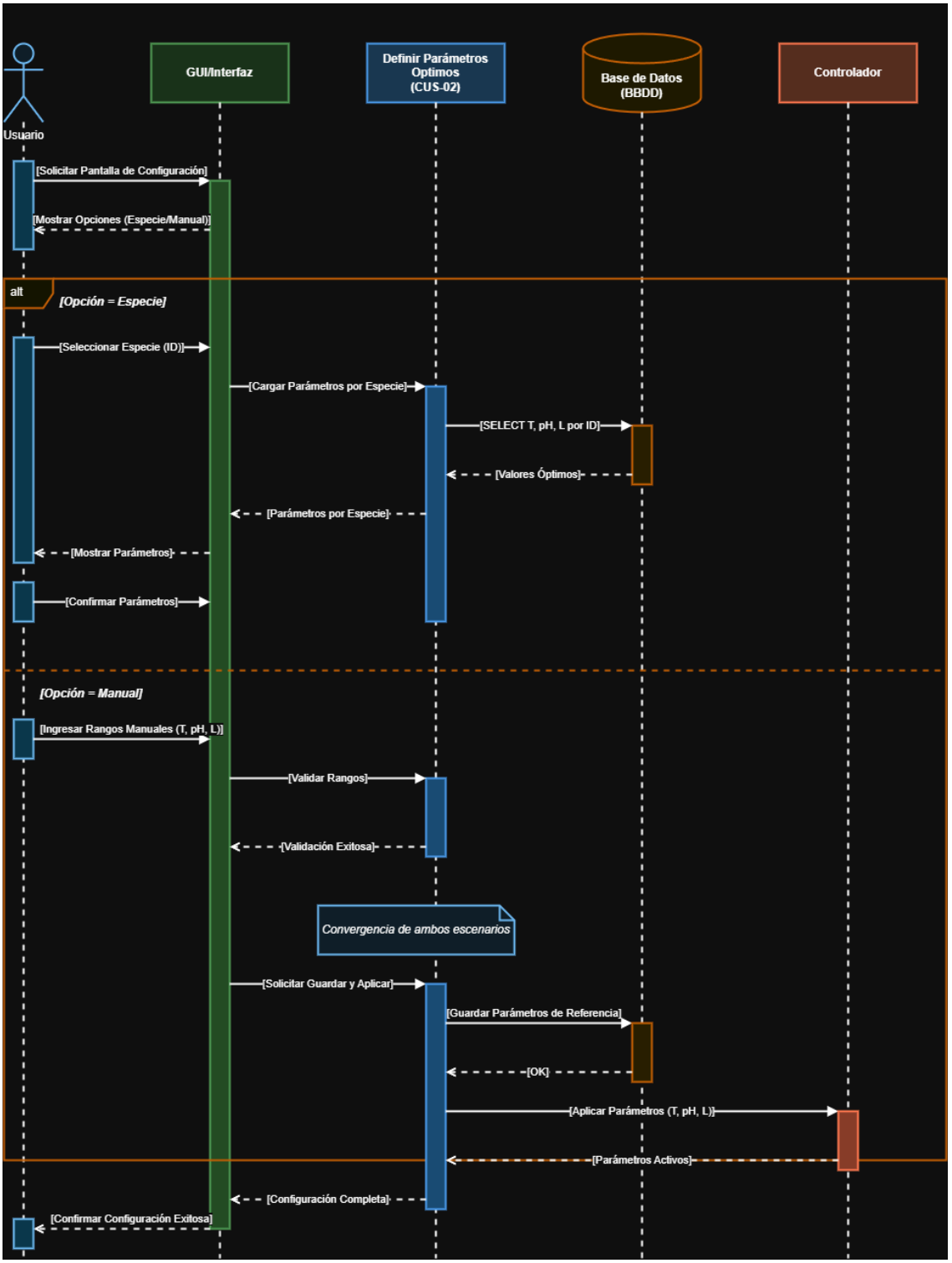



Ilustración 14: Diagrama actividades CU-02

- CU-03: Mantener Ambiente Óptimo

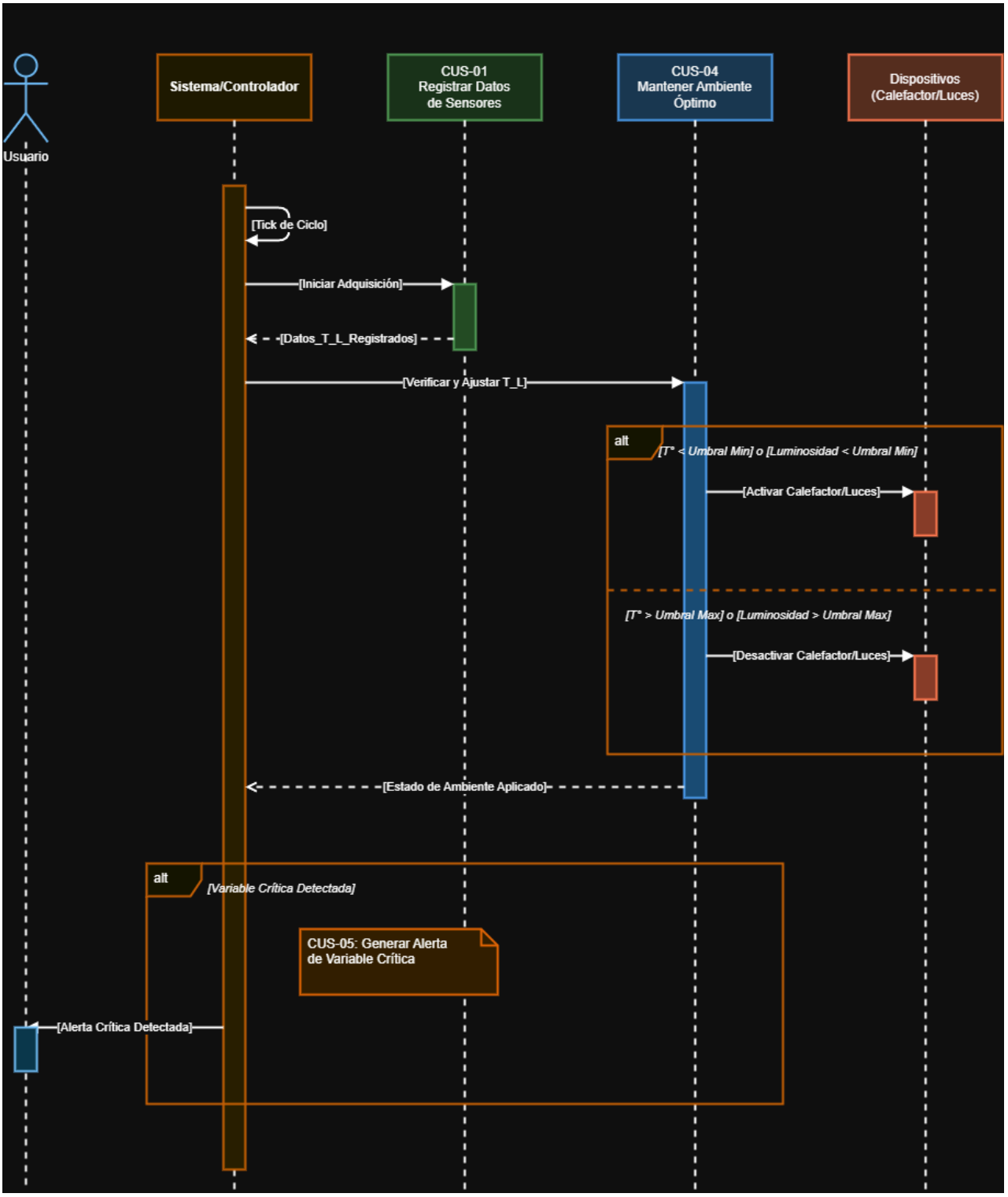



Ilustración 15: Diagrama Secuencia CU-03

- CU-03: Mantener Ambiente Óptimo

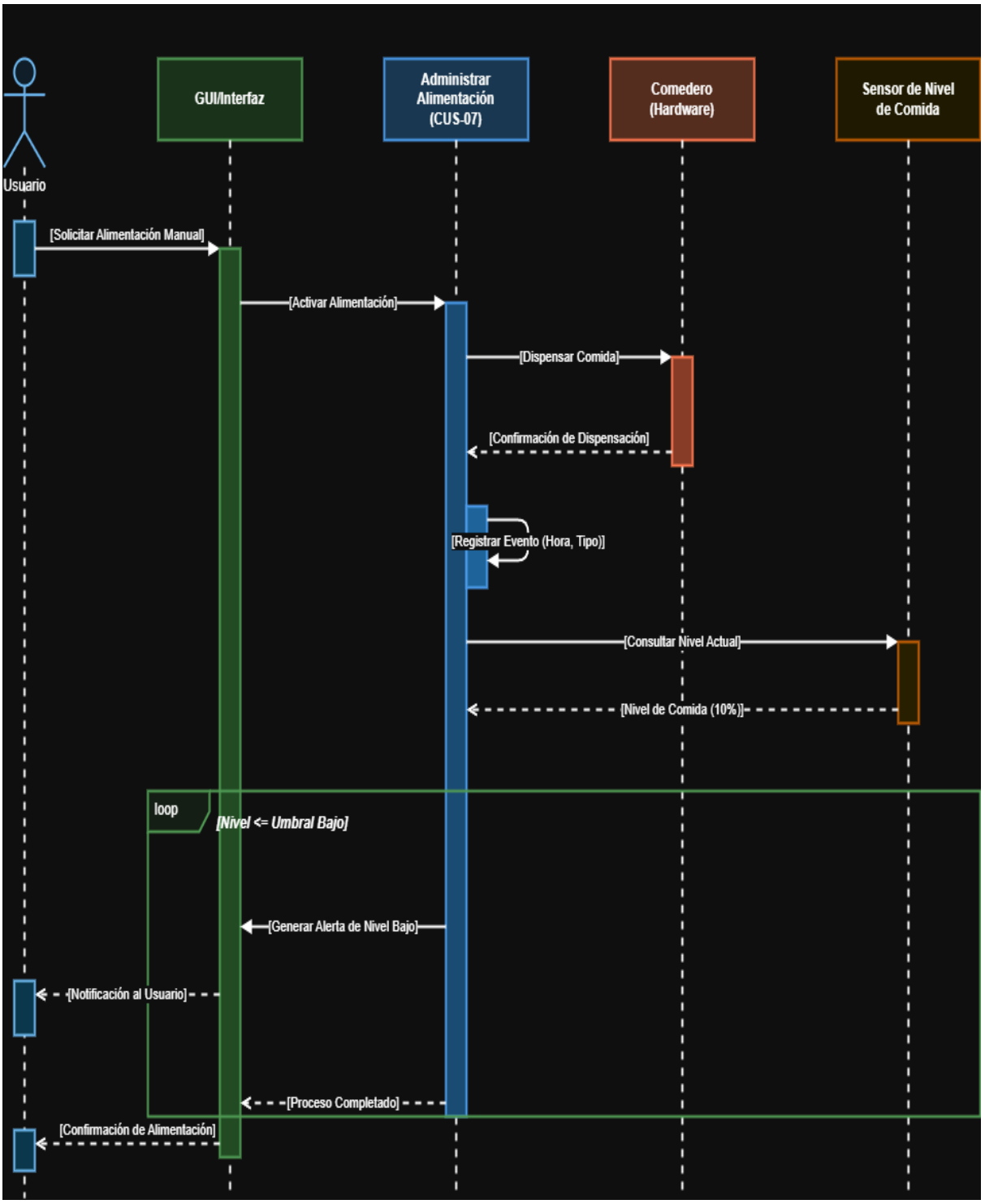



Ilustración 16: Diagrama Secuencia CU-06

## 4.2. Herramientas y técnicas

Para el desarrollo del sistema, se ha definido el siguiente stack tecnológico:

- **Visual Studio Code:** IDE principal utilizado para la edición de código y la administración general de los archivos del proyecto.
- **Python:** Lenguaje de programación base sobre el cual se construirá toda la lógica de control y automatización de la aplicación.
- **RPi.GPIO:** Librería específica de Python que facilita la comunicación con el hardware, permitiendo capturar datos de los sensores y controlar los actuadores a través de los pines GPIO.
-

# 5. Diseño del sistema

## 5.1. Modelo de Diseño (Arquitectura Lógica)

**Nota:** El sistema se ha implementado utilizando un paradigma híbrido. Aunque el código principal (*main.py*) es procedural por eficiencia en Raspberry Pi, lógicamente el sistema se estructura en los siguientes módulos o "clases lógicas":

- **Clase SensorManager:** Responsable de la adquisición de datos del entorno. Encapsula las lecturas de hardware heterogéneo.
  - *Atributos:* *SENSOR_TEMP_PATH* (W1), *PORT_PH* (A0), *PORT_SENSOR_LUZ* (A1), *PORT_ULTRASONIC* (D8).
  - *Métodos:* *leer_temperatura()* (DS18B20), *leer_ph()* (conversión lineal), *leer_nivel_agua()* (ultrasónico), *leer_luz()*.
- **Clase CloudManager (Firebase):** Gestiona la conectividad IoT y la sincronización de datos en tiempo real.
  - *Atributos:* *db* (Firestore), *credenciales*.
  - *Métodos:* *sincronizar_configuracion()*, *subir_estado()*, *enviar_notificacion_push()*.
- **Clase ActuatorController:** Gestiona los dispositivos de salida y las acciones físicas.
  - *Métodos:* *gestionar_calefactor()* (Relé), *gestionar_comida()* (Servo), *actualizar_lcd()* (Grove RGB).
- **Clase MainSystem:** El orquestador que ejecuta el ciclo infinito (*while True*), maneja las excepciones y coordina la comunicación entre sensores, nube y actuadores.

## 5.2. Modelo de Interacción (Flujo de Acciones)

El sistema opera bajo un ciclo de control continuo con integración en la nube, siguiendo esta secuencia lógica:

1. **Inicialización:** Carga de credenciales de Firebase, inicio del bus OneWire (temp) e I2C (GrovePi).
2. **Sincronización (Downlink):** El sistema consulta a Firestore para descargar la configuración más reciente (rangos de pH, horarios de comida).
3. **Adquisición de Datos:** Se leen secuencialmente: Temperatura (DS18B20), pH (Voltaje analógico), Nivel de agua (Distancia ultrasónica) y Luz.
4. **Lógica de Control:**
   - Si *Temp_Actual* < *Temp_Min* → Encender Calefactor (Relé).
   - Si *Hora_Actual* coincide con *Horarios_Comida* → Activar Servo (Alimentar).
5. **Gestión de Alertas:** Si algún parámetro sale del rango seguro, se genera una notificación Push inmediata al dispositivo móvil del usuario.
6. **Publicación (Uplink):** Se sube el estado actual de todos los sensores a la base de datos en la nube.

7. **Feedback Local:** Se actualiza la pantalla LCD con los valores y colores de estado (Verde=OK, Naranja=Calentando).

## 5.3. Descripción de la Arquitectura

El diseño sigue un patrón de IoT Layered Architecture (Arquitectura IoT por capas):

- **Capa Física (Hardware):** Sensores (DS18B20, Electrodo pH, Ultrasónico) y Actuadores (Servo, Relé) conectados al GrovePi+.
- **Capa de Procesamiento (Software):** Script en Python (*main.py*) ejecutándose en la Raspberry Pi, encargado de la lógica crítica y seguridad del pez (funciona aunque se corte internet).
- **Capa de Red (Internet):** Conexión WiFi gestionando peticiones HTTPS a Google Cloud/Firebase.
- **Capa de Aplicación (BD y App movil):** Firebase Firestore (Base de datos) y Firebase Cloud Messaging (Notificaciones).

# 6. Implementación

## 6.1. Plan de Integración

La estrategia de integración utilizada fue Incremental Ascendente (Bottom-Up), añadiendo la complejidad de la nube al final:

- **Fase 1 - Hardware Base:** Validación de sensores individuales (Lectura del archivo W1 para temperatura, lectura analógica para pH).
- **Fase 2 - Lógica Local:** Implementación del control del calefactor y el alimentador automático sin conexión a internet.
- **Fase 3 - Conectividad IoT:** Integración del SDK *firebase_admin*. Se programó la autenticación mediante *serviceAccountKey.json* y la sincronización de documentos JSON.
- **Fase 4 - Integración Total:** Fusión del bucle de control de hardware con las funciones asíncronas de la base de datos en el script *main.py*, añadiendo manejo de excepciones *try-except* para evitar caídas por fallos de red.

## 6.2. Descripción de los Módulos Implementados (Detalle Técnico)

A. Módulo de Temperatura (DS18B20):

A diferencia de sensores básicos, se utilizó el sensor digital impermeable DS18B20 que se comunica por el protocolo 1-Wire.

- *Implementación:* Lectura del archivo de sistema en */sys/bus/w1/devices/*, parseando la salida de texto para obtener la temperatura con precisión de 2 decimales.

**B. Módulo de pH y Calibración:**

- Implementación: Se utiliza el puerto analógico A0. La conversión de señal eléctrica a valor químico sigue la fórmula lineal implementada:
  $$pH = 7.0 + ((Voltaje - 2.50) \times 4.17)$$
  Donde $2.50V$ es el punto neutro y $4.17$ es el factor de sensibilidad ajustado experimentalmente.

**C. Módulo de Nivel y Seguridad:**

- *Implementación:* Uso de un sensor ultrasónico en el puerto D8. Se calcula el porcentaje de llenado en función de la distancia medida al espejo de agua (Altura total configurada: 40cm).

**D. Módulo de Gestión Remota (Firebase):**

- *Implementación:* El sistema no guarda datos localmente, sino que utiliza Firestore como backend. Esto permite que el usuario cambie la temperatura deseada desde una App y la Raspberry Pi lo reciba en el siguiente ciclo de sincronización (*sincronizar_configuracion*).

### 6.3. Reporte de Revisión

Se realizaron pruebas integrales al sistema "SaveNemo":

- **Prueba de Resiliencia de Red:** Se desconectó el cable de red durante la operación. El sistema detectó el fallo (excepción en *requests*), omitió la subida de datos pero continuó controlando el calefactor y la temperatura, garantizando la supervivencia del pez sin internet.
- **Prueba de Notificaciones:** Se forzó una bajada de pH simulada; el sistema envió exitosamente la notificación "¡ALERTA ACUARIO! PH Bajo" al dispositivo móvil en menos de 2 segundos.
- **Validación del Alimentador:** El servomotor demostró precisión en el movimiento (0° a 90° y regreso), evitando atascos de comida mediante una secuencia de agitación programada (*mover_servo_alimentar*).

# 7. Aspectos Generales

## 7.1. Avance de acuerdo a la Carta Gantt

Actualmente, el proyecto ha completado los hitos críticos de "Integración de Hardware IoT" y "Sincronización con la Nube". Nos encontramos en la fase de validación operativa y documentación final, habiendo cumplido satisfactoriamente con la implementación de la base de datos Firestore y las notificaciones push según el cronograma.

## 7.2. Problemas Encontrados

Durante la implementación, se enfrentaron dos desafíos principales de hardware y software:

- **Lecturas inconsistentes en el sensor de pH:** Inicialmente, el sensor entregaba valores de voltaje inestables o saturados debido a una falta de calibración física del potenciómetro de la interfaz y ruido eléctrico en la señal analógica.
- **Decodificación del Sensor de Temperatura (DS18B20):** A diferencia de sensores simples, el DS18B20 utiliza el protocolo 1-Wire que no entrega un dato directo, sino que genera un archivo de texto en el sistema operativo (*w1_slave*). El desafío fue leer y parsear este archivo correctamente en tiempo real sin bloquear el ciclo principal del programa.

## 7.3. Soluciones Propuestas

Para mitigar los problemas mencionados, se aplicaron las siguientes soluciones técnicas:

- **Para el pH:** Se realizó un ajuste físico del "Offset" en la placa controladora hasta obtener un voltaje basal cercano a 2.5V (neutro). Por software, se implementó una fórmula de conversión lineal ($y = mx + b$) ajustada con soluciones buffer para traducir el voltaje a niveles de pH precisos.
- **Para la Temperatura:** Se desarrolló una función robusta en Python (*leer_temperatura*) que accede al sistema de archivos de la Raspberry Pi, busca la cadena de texto específica (*t=*) y realiza la conversión matemática (división por

1000). Además, se añadió un bloque *try-except* que asigna un valor seguro por defecto (25.0°C) si el sensor se desconecta, evitando que el programa colapse.

## 7.4. Conclusiones

El desarrollo de "Salvando a Nemo" ha demostrado con éxito la integración de tecnologías embebidas (Raspberry Pi/Grove) con servicios en la nube (Firebase). Se logró transicionar de un sistema de control local a uno IoT (Internet of Things), capaz de monitorear variables críticas y reaccionar (alimentar/calentar) de forma autónoma o remota. La arquitectura implementada garantiza la supervivencia de la mascota incluso si la conexión a internet falla, gracias a la lógica de redundancia programada en el controlador local.

## 7.5. Trabajo Futuro

- **Análisis Predictivo de Datos:** Utilizar el historial acumulado en Firebase para generar gráficos de tendencias y predecir cuándo el agua se ensucia antes de que ocurra.
- **Visión Computarizada:** Integrar una cámara PiCam al sistema actual para realizar streaming de video en tiempo real a la aplicación y utilizar procesamiento de imágenes para detectar visualmente anomalías en el pez o turbidez en el agua.

# 8. Conclusión

El desarrollo e implementación del proyecto "Safe a Nemo" ha permitido validar con éxito la aplicación de tecnologías del Internet de las Cosas (IoT) en el cuidado y preservación de ecosistemas acuáticos domésticos. A través de la integración de una placa Raspberry Pi 4, sensores ambientales y actuadores mecánicos, se ha logrado transformar la tarea manual y propensa a errores del mantenimiento de un acuario en un proceso automatizado, preciso y confiable.

Desde una perspectiva técnica, la elección del stack tecnológico fue determinante para el éxito del sistema. La utilización de Visual Studio Code como entorno de desarrollo unificado y Python como lenguaje principal proporcionó la versatilidad necesaria para manejar lógica compleja de programación. Específicamente, la implementación de la librería RPi.GPIO actuó como el puente crítico entre el software y el hardware, permitiendo que el código interpretara fielmente las señales eléctricas de los sensores (entradas) y ejecutara órdenes físicas sobre el calefactor, las luces y el comedero (salidas) con una latencia mínima.

El modelado del sistema, basado en una arquitectura donde los sensores y actuadores operan como actores activos, demostró ser altamente eficiente. Esta estructura permite definir flujos de trabajo claros donde el sistema no es un ente pasivo, sino que reacciona dinámicamente ante los cambios del entorno. La capacidad del sistema para autorregularse (manteniendo la temperatura y luminosidad en rangos óptimos sin intervención humana) y para gestionar la alimentación de forma autónoma cumple con el objetivo fundamental del proyecto: garantizar la supervivencia de las especies y reducir la carga operativa del usuario.

Asimismo, el sistema de alertas críticas para variables sensibles como el pH y el nivel de agua añade una capa de seguridad indispensable. Al notificar al usuario en tiempo real sobre anomalías que el sistema no puede corregir por sí mismo, se asegura una intervención oportuna, mitigando riesgos que en un acuario tradicional pasarían desapercibidos hasta ser fatales.

En definitiva, "Safe a Nemo" demuestra que es posible crear un ecosistema autosustentable y monitoreado, donde la tecnología actúa como garante del bienestar biológico, ofreciendo al usuario final una herramienta robusta para la toma de decisiones informadas y el control total de su acuario.

# 9. Referencias


[1] Aquaforest. (2023, Junio 19). Water parameters in tank - How to use water tests? Aquaforest. https://aquaforest.eu/en/knowledge-base/testing-your-aquarium-water-guide-to-essential-water-parameter/

[2] Flores Mollo, S., & Aracena Pizarro, D. (2018, August 06). Sistema de monitoreo remoto de acuicultura en estanques para la crianza de camarones. Ingeniare. Revista chilena de ingeniería. https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052018000500055&lng=es&nrm=iso&tlng=es

[3] Guía Definitiva Acuario para Principiantes 2023 - AQ, Aquarium Solutions. (n.d.). AQ-arium. https://www.aq-arium.com/aqmarine/guia-definitiva-acuario-para-principiantes-2023/

[4] Kit Módulo Sensor De pH + Sonda De Electrodo BNC Arduino - Envío Gratis A Todo Chile. (2020, November 9). MechatronicStore. https://www.mechatronicstore.cl/kit-modulo-sensor-de-ph-sonda-de-electrodo-bnc-arduino/

[5] Raspberry Pi Documentation. (n.d.). Raspberry Pi. https://www.raspberrypi.com/documentation/

[6] Seeed Studio. (2023, Marzo 13). Grove Ecosystem Introduction. Seeed Studio Wiki. https://wiki.seeedstudio.com/Grove_System/

[7] Sensor De Temperatura Ds18b20 Tipo Termocúpula - Envío Gratis A Todo Chile. (n.d.). MechatronicStore. https://www.mechatronicstore.cl/sensor-de-temperatura-ds18b20-tipo-termocupla-arduino-pic/

[8] Sueldo de Documentador en Chile. (n.d.). Indeed. https://cl.indeed.com/career/archivista-y-capturista/salaries

[9] Sueldo de jefe de proyecto en Chile. (n.d.). Indeed Chile. https://cl.indeed.com/career/jefe-de-proyecto/salaries

[10] Sueldo de programador en Chile. (n.d.). Indeed Chile. https://cl.indeed.com/career/programador-web/salaries

[11] Sueldo de Técnico Electrónico en Chile. (n.d.). Indeed. https://cl.indeed.com/career/t%C3%A9cnico-electronico/salaries

[12] Meta. (n.d.). Meta Quest Developer Center: Mixed Reality & Passthrough. https://developer.oculus.com/documentation/